MASTER EN SCIENCES INFORMATIQUES

L'autonomie de la personne ayant une déficience mentale : un logiciel d'aide à la gestion des achats alimentaires

Topet, Léopold; Vonèche, Xavier

Award date:
1991

Awarding institution:
Universite de Namur

Année académique 1990-1991

L'autonomie de la personne ayant une déficience mentale : un logiciel d'aide à la gestion des achats alimentaires.

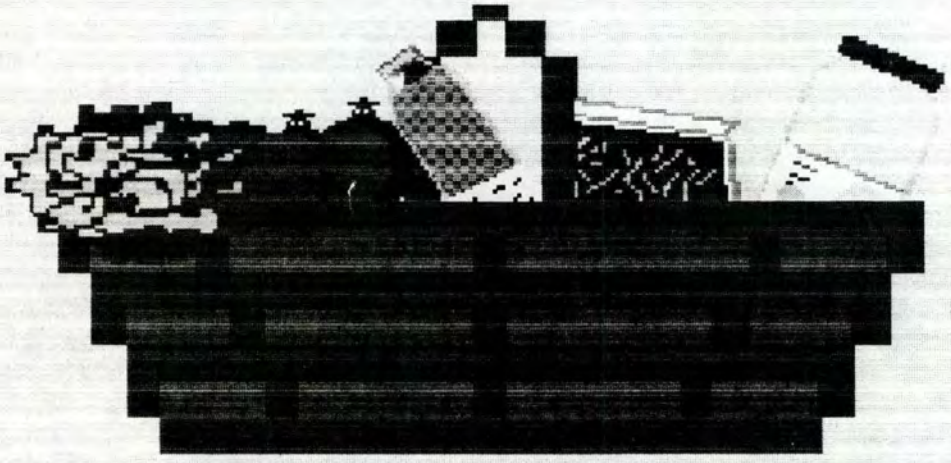
MANUEL d'UTILISATION.

Léopold Topet **Xavier Vonèche**

Promoteur : Madame le professeur Monique Noirhomme-Fraiture.
(Institut d'Informatique)
Co-promoteur : Monsieur le professeur Michel Mercier.
(Département de psychologie de la Faculté de Médecine)

Mémoire présenté en vue de l'obtention du titre de Licencié et Maître en Informatique.

Je fais mes courses



Promoteurs : M. Noirhomme (Institut d'Informatique)
M. Mercier (Département de Psychologie)

Auteurs : L. Topet (étudiant Institut d'Informatique)
X. Vonèche (étudiant Institut d'Informatique)

Mémoire présenté en vue de l'obtention du titre de Licencié et Maître en Informatique. FUNDP, 1990-1991.

Partie I

Présentation du logiciel "Je fais mes courses".

I.1 INTRODUCTION.

Le logiciel "Je fais mes courses" a été conçu et développé en collaboration avec l'Institut d'Informatique des Facultés de Namur, et le centre Psinha (PSychologie, INformatique et HAndicap, département de psychologie des Facultés de Namur), sur une idée de Martine Lamy, logopède à la Clairière à Bruxelles.

Ce document débutera par une brève présentation du logiciel : les objectifs à remplir, les fonctionnalités à offertes, la population visée, les pré-requis nécessaires à son utilisation et ses limites, notamment.

I.2 L'IDEE DE DEPART.

Dans certains appartements supervisés et maisons communautaires, où vivent des personnes ayant une déficience mentale, l'établissement d'une liste d'achats alimentaires est un travail généralement réalisé conjointement par les habitants de ces lieux et par des éducateurs. C'est une tâche qui consiste d'abord à choisir des plats pour les repas complets de la semaine, et ensuite à faire le tour des armoires et frigos dans le but de déterminer les produits à acheter ainsi que les quantités dans lesquelles les acheter, afin

d'une part de pouvoir préparer les menus et d'autre part de disposer de produits dits de base (pour les repas tartines, ...). C'est un travail qu'une personne ayant une déficience mentale a en général beaucoup de difficultés à réaliser seule. Une liste d'achats est alors établie, sous une forme compréhensible par les intéressés, et ce sont ensuite ces derniers, avec ou sans aide d'un éducateur, qui effectuent, dans un grand magasin, les achats spécifiés.

En réalité, ce sont cependant surtout les éducateurs qui s'investissent dans ce travail, car, bien souvent surchargés, ils ne disposent pas de suffisamment de temps pour discuter, avec les personnes dont ils s'occupent, du choix des plats, pour essayer de leur apprendre à calculer eux-mêmes les quantités restantes des produits, ...

On constate ainsi que, souvent, ce travail est peu enrichissant pour les différentes parties, de plus il est long et répétitif (hebdomadaire en général).

C'est ainsi qu'un projet de développement de logiciel a été proposé au centre Psinha, logiciel qui présente aux utilisateurs un certain nombre de plats, parmi lesquels ils doivent alors effectuer des choix. Le logiciel doit également établir une liste des achats nécessaires pour la confection des plats choisis, liste qui soit bien sûr sous une forme utilisable par une personne ayant une déficience mentale.

I.3 LES FONCTIONNALITES DU LOGICIEL.

I.3.1 ENUMERATION DES FONCTIONNALITES.

Le logiciel[1] "Je fais mes courses" permet :

-1 : de constituer des menus pour certaines dates (+ enregistrement des menus);
-2 : de modifier les menus constitués (+ enregistrement des modifications);
-3 : de consulter des menus constitués;
-4 : d'obtenir l'impression des menus constitués;

[1] Sont présentées ici uniquement les fonctionnalités développées, d'autres sont prévues qui ne sont pas encore offertes aux utilisateurs. Se référer au texte du mémoire.

-5 : de consulter de la liste des produits entrant dans la composition des plats d'un certain menu;
-6 : d'obtenir l'impression de cette liste;

-7 : d'obtenir l'établissement d'une liste d'achats pour cette période, qui reprend les produits alimentaires à acheter et les quantités à acheter de ces produits pour la préparation des menus constitués pour une période qu'il doit spécifier;
-8 : d'obtenir l'impression de cette liste.

La forme de la liste doit être telle qu'une personne ayant une déficience mentale soit à même de pouvoir l'utiliser pour réaliser, de la façon la plus autonome possible, c'est-à-dire sans devoir s'adresser à un vendeur ou tout autre personne, les achats prescrits par la liste, de même qu'elle doit être comprise par toute personne à laquelle, le cas échéant, elle demanderait de l'aide. De même, les menus et listes de produits imprimés doivent être utilisés par cette même catégorie de personnes.

I.3.2 DEROULEMENT DU PROGRAMME.

Avant de présenter les écrans et la façon d'utiliser le logiciel, nous allons expliquer les grands principes du déroulement du programme, afin d'obtenir une vue d'ensemble de ce dernier.

Deux parties, deux "logiciels" sont à distinguer :

- 1 : la partie Menu;
- 2 : la partie Liste d'achats.

Au début de toute session, vous devez introduire votre nom, ceci afin de contrôler ultérieurement que les disquettes que vous utiliserez sont bien les vôtres.

I.3.2.1 La partie Menu :

Elle englobe les fonctionnalités 1 à 6 vues ci-dessus. Les fonctionnalités 5 et 6 sont liées aux quatre premières, et ne sont accessibles que via ces dernières.

-1 : La création d'un menu.

La création d'un menu consiste à spécifier trois caractéristiques :

- la date pour laquelle le menu est prévu;
- le nombre de personnes pour lequel le menu est à préparer (maximum 6);
- un ensemble de couples (thème[1]-plat) constituant le menu.

Il y a deux scénarios possibles de création de menus :

-1 : dans le premier, l'ordinateur vous impose, une à une, les dates des jours pour lesquels vous pouvez créer des menus. Pour chacune de ces dates, vous pouvez accepter ou refuser de créer un menu. Si vous acceptez, vous devez alors spécifier le nombre de personnes, les thèmes et les plats. Si vous refusez, l'ordinateur vous impose la date suivante. Et ainsi de suite, jusqu'à ce que vous signifiez que vous souhaitez arrêter de travailler ou jusqu'à ce que tous les jours soient épuisés (voir ci-dessous).

-2 : dans le second scénario, c'est vous-même qui choisissez une date parmi un ensemble de dates proposées par l'ordinateur. Il ne s'agit donc plus ici de devoir indiquer, pour chaque date, si oui ou non, vous désirez créer un menu.

Les dates proposées sont toujours des dates non antérieures à la date du jour courant et ne sont pas trop éloignées de cette dernière. Nous verrons plus tard, au point concernant la paramétrisation, que c'est vous qui pouvez déterminer l'éloignement maximum.

---

[1] Un thème est un "moment" d'un repas : l'entrée, le plat principal, ...

Les nombres de personnes, les thèmes et les plats, sont à choisir parmi ceux proposés par l'ordinateur.

En ce qui concerne les thèmes, il peut en exister des décomposables. Typiquement, le thème Plat principal peut se décomposer en les thèmes Viande, Légume, Accompagnement. C'est à vous (paramétrisation) de choisir si les thèmes décomposables sont proposés ou au contraire si ce sont leurs fils qui sont proposés. L'utilisation de thèmes décomposables permet de vous aider dans les mariages de plats.

En cours de travail sur un menu, il est possible de supprimer des thèmes qui ont été spécifiés pour ce dernier. Une fois un menu créé, vous pouvez demander son impression, ainsi que la consultation et l'impression de la liste des ingrédients entrant dans la composition des plats le constituant.

-2 : La modification d'un menu.

La modification d'un menu consiste à ajouter/supprimer à/d'un menu auparavant créé, des couples (thème-plat), et/ou à modifier le nombre de personnes de ce menu, ainsi qu'à nouveau la possibilité de consulter/-imprimer la liste des ingrédients des plats du menu modifié et celle d'imprimer ce menu. Vous avez bien sûr le loisir de consulter et d'imprimer votre menu.

-3 : La consultation d'un menu.

La consultation d'un menu consiste à demander l'affichage à l'écran du contenu d'un menu créé, ainsi qu'il vous est possible de demander l'impression de ce menu, de même, à nouveau, que la consultation/-impression de la liste des ingrédients du menu.

-4 : L'impression d'un menu.

L'impression d'un menu consiste à demander l'impression du contenu d'un menu créé.

I.3.2.2 La partie Liste d'achats :

Elle englobe les fonctionnalités 7 et 8, la huitième n'étant accessible que via la septième.

L'établissement d'une liste d'achats consiste tout d'abord, pour l'utilisateur, à spécifier une période à couvrir par cette liste. Cela se fait par choix des deux dates délimitant cette période, parmi les dates proposées par l'ordinateur. Dans le cas du premier scénario de création de menus la période choisie doit avoir été entièrement traitée[1].

L'ordinateur calcule ensuite la liste des produits nécessaires et les quantités nécessaires de ces produits pour la période pour satisfaire aux menus de la période. Pour chaque produit nécessaire, l'ordinateur vous demande s'il vous en reste au moins autant que nécessaire; si ce n'est pas le cas, vous devez spécifier ce qu'il vous en reste.

L'ordinateur établit alors la liste d'achats de la période, comprenant les produits à acheter et les quantités à acheter des ces produits, et il imprime cette liste.

Vous pouvez demander l'établissement et l'impression d'une liste pour une période ne prenant cours qu'un jour ultérieur au jour courant.

Pour savoir si vous disposez encore d'autant de produit que nécessaire, on distinguera en réalité deux types de produits : ceux pour lesquels on employera une gestion de stock dite du type de la ligne rouge, et les autres. Pour ceux de la première catégorie, il n'est jamais précisé la quantité exacte nécessaire, il vous est simplement demandé, le cas échéant, si vous disposez d'assez du produit, auquel cas, vous devez déterminer vous-même ce que signifie "assez". C'est un système que l'on pourra utiliser avec les épices notamment, produit pour lequel il est difficile de mesurer une quantité. Vous pouvez dessiner dans ce cas une ligne rouge sur les pots et où "assez" veut alors dire que le niveau du produit arrive au dessus de la ligne rouge. Ce système de la ligne rouge est un "système de

---

[1] Une période de jours est dite traitée si, pour tous les jours qui la composent, l'utilisateur a dit (partie Menu) si oui ou non il désirait créer un menu.

mesure" parmi d'autres. C'est à vous de voir, en fonction des produits concernés, quel système est le plus approprié.

Pour les produits de la seconde catégorie, les quantités nécessaires sont exprimées en nombre d'unités de conditionnement d'utilisation de ces derniers. Celui de la pomme, par exemple, est la pomme elle-même, celui de la purée en flocons, est le sachet.

Quant aux quantités à acheter, quelle que soit la catégorie à laquelle le produit concerné appartient, elles sont exprimées en nombre d'unités de conditionnement de vente. Le conditionnement de vente de la pomme est encore la pomme, tandis que celui de la purée en flocons est la boîte, de quatre sachets par exemple.

Dans la seconde partie de ce document, à l'aide des écrans, nous verrons comment les plats, les produits, ... sont représentés, ainsi que la façon dont les quantités restantes, ... doivent être spécifiés.

I.4 LES OBJECTIFS ATTENDUS DU LOGICIEL.

I.4.1 DU COTE DE L'EDUCATEUR.

Nous avons eu l'occasion au point précédent de présenter l'objectif de base du logiciel : libérer l'éducateur de l'accomplissement d'une tâche peu enrichissante.

I.4.2 DU COTE DES PERSONNES AYANT UNE DEFICIENCE MENTALE.

Si l'on se place maintenant du côté des personnes ayant une déficience mentale, à quels objectifs doit répondre ce logiciel ? Tout d'abord ce logiciel devrait leur permettre d'accomplir seules un travail pour lequel elles étaient assistées et donc d'obtenir une certaine autonomie dans ce domaine. Un second objectif visé avec ce logiciel et qui découle de ce qui vient d'être dit, est de permettre aux personnes concernées de choisir des plats et de ne plus tout le temps manger la même chose. La façon d'agir des personnes ayant une déficience mentale est très marquée par les habitudes (les habitudes rassurent), et il est difficile de modifier ces

dernières. Cette constatation est aussi valable pour le choix des plats où l'on constate que de semaine en semaine les mêmes choix sont opérés et ce en dépit des conseils des éducateurs. On peut alors espérer qu'avec "Je fais mes courses", différents plats étant proposés et l'ordinateur étant peut-être plus disponible que les éducateurs pour ce travail, les utilisateurs, après un temps d'adaptation au logiciel, vont varier leurs menus.

I.5 POPULATION VISEE.

Comme nous le savons, "Je fais mes courses" s'adresse aux personnes ayant une déficience mentale et plus particulièrement, parmi elles, celles ayant un niveau léger à modéré de déficience. Pour les personnes ayant une déficience de niveau élevé, ce logiciel ne servirait à rien. Sans préjuger de l'utilisation du logiciel dans d'autres circonstances, en ce qui concerne la situation des personnes par rapport à une institution spécialisée, plusieurs catégories sont visées :

-1 : comme nous l'avons vu ci-dessus, les personnes vivant en appartement supervisé (ou maisons communautaires) et éprouvant des problèmes pour l'établissement d'une liste d'achats alimentaires.

-2 : les personnes vivant en dehors de toute institution mais qui pour certains travaux recourent à une aide extérieure (accompagnant) et notamment pour l'établissement d'une liste d'achats.

-3 : les personnes placées dans une institution et auxquelles le personnel éducatif de l'établissement souhaiterait faire acquérir un peu d'autonomie dans le domaine qui nous intéresse afin, moyennant l'acquisition d'autonomie dans d'autres domaines, de leur permettre de peut-être un jour quitter l'institution, ou d'aller peut-être vivre dans un appartement supervisé.

I.6 PRE-REQUIS NECESSAIRES A L'UTILISATION DU LOGICIEL.

Les pré-requis peuvent ne pas se distinguer de ce qui a été dit ci-dessus en ce sens qu'ils ne font que préciser la population ciblée; cependant nous différencierons ces deux points parce que le point précédent désigne un état de fait alors que les pré-requis désignent des capacités qu'il est éventuellement possible d'acquérir par certains apprentissages.

Il y a trois choses à bien distinguer :

- 1 : les pré-requis nécessaires à l'utilisation du logiciel.
- 2 : les pré-requis nécessaires à l'utilisation des productions du logiciel qui sont les menus, liste d'ingrédients composant un menu et liste d'achats.
- 3 : les pré-requis nécessaires à la réalisation des activités engendrées par l'utilisation de ces productions.

I.6.1 PRE-REQUIS NECESSAIRES A L'UTILISATION DU LOGICIEL.

-1 : à un niveau tout-à-fait technique, savoir démarrer un ordinateur, savoir utiliser une imprimante, savoir introduire correctement une disquette et savoir utiliser la souris d'un ordinateur.
-2 : avoir des notions de temps : notamment la semaine, les jours de la semaine, et la distance d'un jour par rapport au jour courant.
-3 : avoir des connaissances dans le mariage des plats (la proposition de thèmes composés permet d'éviter ce pré-requis).
-4 : avoir des connaissances dans les produits alimentaires (savoir ce qu'est du lait, du jambon, ...).
-5 : savoir compter (des petites quantités : nombre de boites, ...).
-6 : avoir un système de représentation des objets et des actions à accomplir sur ces objets identique à celui de l'interface (se référer à la seconde partie de ce document pour voir ce que sont ces représentations).

I.6.2 PRE-REQUIS NECESSAIRES A L'UTILISATION DES PRODUCTIONS DU LOGICIEL.

-1 : avoir des connaissances dans les produits alimentaires (savoir ce qu'est du lait, du jambon, ...).
-2 : savoir compter (des petites quantités).
-3 : avoir un système de représentation des objets véhiculés par les productions identiques à celui de ces dernières (se référer à la seconde partie de ce document pour voir ce que sont ces représentations).

I.6.3 PRE-REQUIS NECESSAIRES A LA REALISATION DES ACTIVITES ENGENDREES PAR L'UTILISATION DE CES PRODUCTIONS.

-1 : pouvoir se déplacer jusqu'à un grand magasin et savoir se diriger dans les rayons.
-2 : avoir des habitudes d'achat.
-3 : avoir des habitudes de consommation et être ainsi à même de pouvoir gérer le contenu de ses armoires et répartir les produits disponibles entre les différents repas de la semaine.
-4 : pouvoir contrôler les dates de péremption.
-5 : pouvoir contrôler que les échanges d'argent avec les caissières sont corrects.

Une personne peut très bien posséder les pré-requis nécessaires à l'utilisation du logiciel et de ses productions sans pouvoir pour autant accomplir les activités engendrées par ces dernières (ne pas pouvoir faire seule ses achats, ne pas pouvoir cuisiner seule,...), auquel cas elle pourra recourir à l'aide d'une tierce personne pour la réalisation de ces activités. Aucune action en rapport avec ce logiciel, que ce soit au niveau de l'interface ou autre, n'aura de conséquence dessus.

I.7 QUI DETERMINE LES THEMES, PLATS ET PRODUITS PROPOSES A L'UTILISATEUR ?

Actuellement, tout cela est fourni d'office avec le logiciel. Les changements qu'il serait nécessaire d'apporter, en vue d'une adaptation à un utilisateur en particulier, doivent être réalisés par une personne désignée pour cela au Psinha.

En annexe, vous trouverez les thèmes, plats et produits proposés à l'heure actuelle.

I.8 LIMITES DU LOGICIEL.

Ci-après, sont énumérées les principales limites du logiciel :

-1 : ce logiciel n'exerce aucun contrôle quant à ce qui est choisi, quant à la fréquence du choix de chaque plat, quant au mariage des plats choisis pour un menu (si ce n'est que l'on peut proposer le thème plat principal pour lequel les plats proposés sont des repas complets (viande, légume, accompagnement) dont les différents constituants ont été déterminés par une tierce personne).

-2 : aucun contrôle n'est exercé quant au stock restant des produits nécessaires qui est déclaré par l'utilisateur, si ce n'est que les quantités proposées parmi lesquelles l'utilisateur peut choisir, ont une valeur minimale et maximale. Donc si ce dernier utilise "mal" le logiciel, il y risque de sur-stock et/ou de sous-stock de produits alimentaires dans les réserves de l'utilisateur.

-3 : aucun contrôle du budget alimentaire

On pourrait presque résumer ces contraintes en une seule qui est : le logiciel ne règle pas tous les problèmes de choix et d'achat de produits alimentaires.

I.9 LA PARAMETRISATION.

L'éducateur peut paramétriser le logiciel en vue de l'adapter davantage à un utilisateur. Il peut déterminer :

-1 : si le son est ou n'est pas utilisé lors du dialogue avec la machine;
-2 : le scénario (1 ou 2) de création de menus;
-3 : si l'ordinateur tutoye ou vouvoye l'utilisateur dans ses messages;
-4 : si les thèmes et plats proposés sont les décomposables ou non;
-5 : l'éloignement maximum entre le jour courant et tout jour à traiter (maximum 9 jours);
-6 : le nombre maximum de jours qu'il est possible de traiter en une session (maximum 7 jours).

Partie II

Manuel d'utilisation.

II.1 LE MATERIEL REQUIS.

Le logiciel est conçu pour s'exécuter sur un ordinateur Amiga *500* (Commodore) pourvu, au minimum, de deux lecteurs de disquette (l'interne, monté sur la machine, et un externe) et d'une mémoire vive de 1 mégabytes. Un disque dur n'est pas nécessaire. Une imprimante est bien entendu requise.

II.2 PRINCIPES GENERAUX D'UTILISATION DU LOGICIEL.

II.2.1 LES DISPOSITIFS D'ENTREE ET DE SORTIE.

Deux dispositifs d'entrée sont prévus :

-1 : la pression sur les touches du clavier pour l'introduction du nom en début de session.
-2 : le simple cliking (presser-lacher, une fois) sur la touche gauche de la souris pour toute autre opération.

Trois dispositifs de sortie sont prévus :

-1 : l'affichage à l'écran de certaines informations.
-2 : la prononciation de certaines informations.
-3 : l'impression de certaines informations.

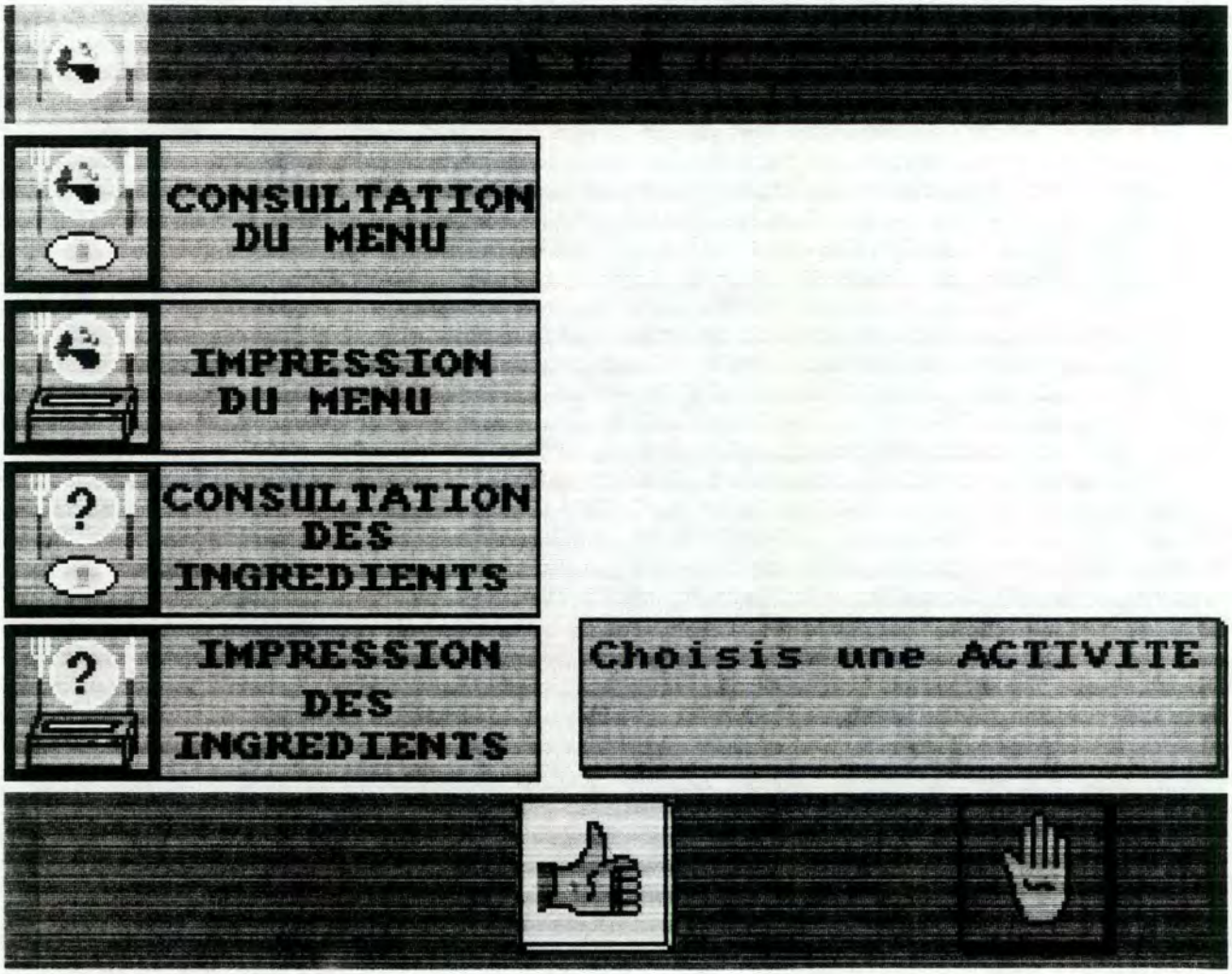
II.2.2 LES DIFFERENTS TYPES D'ECRANS ET DE FENETRES.

On peut distinguer deux grands types d'écrans et deux grands types de fenêtres utilisés pour le dialogue entre vous et l'ordinateur.

Pour chacun de ces types, sur base d'un exemple tiré du logiciel, nous expliquerons son mode de fonctionnement. Il vous sera ensuite très facile d'extrapoler ce que nous aurons vu aux autres écrans et fenêtres du même type. Vous reconnaîtrez facilement les écrans ou fenêtres d'un même type, car ils sont quasi-identiques pour l'essentiel. Chaque écran ou fenêtre pouvant néanmoins avoir ses caractéristiques propres, nous verrons celles-ci lorsque nous étudierons chaque écran et fenêtre en particulier.

Il y a des écrans et des fenêtres qui ne font partie d'aucun des quatre types que nous allons voir dans ce point, nous parlerons de leur fonctionnement en temps opportun.

II.2.2.1 Les écrans pour le choix d'une fonctionnalité.



Après l'affichage d'un tel écran, son but est prononcé : "Choisis une activité". A tout instant, vous pouvez faire répéter cette prononciation en cliquant sur le bouton où le but est écrit.

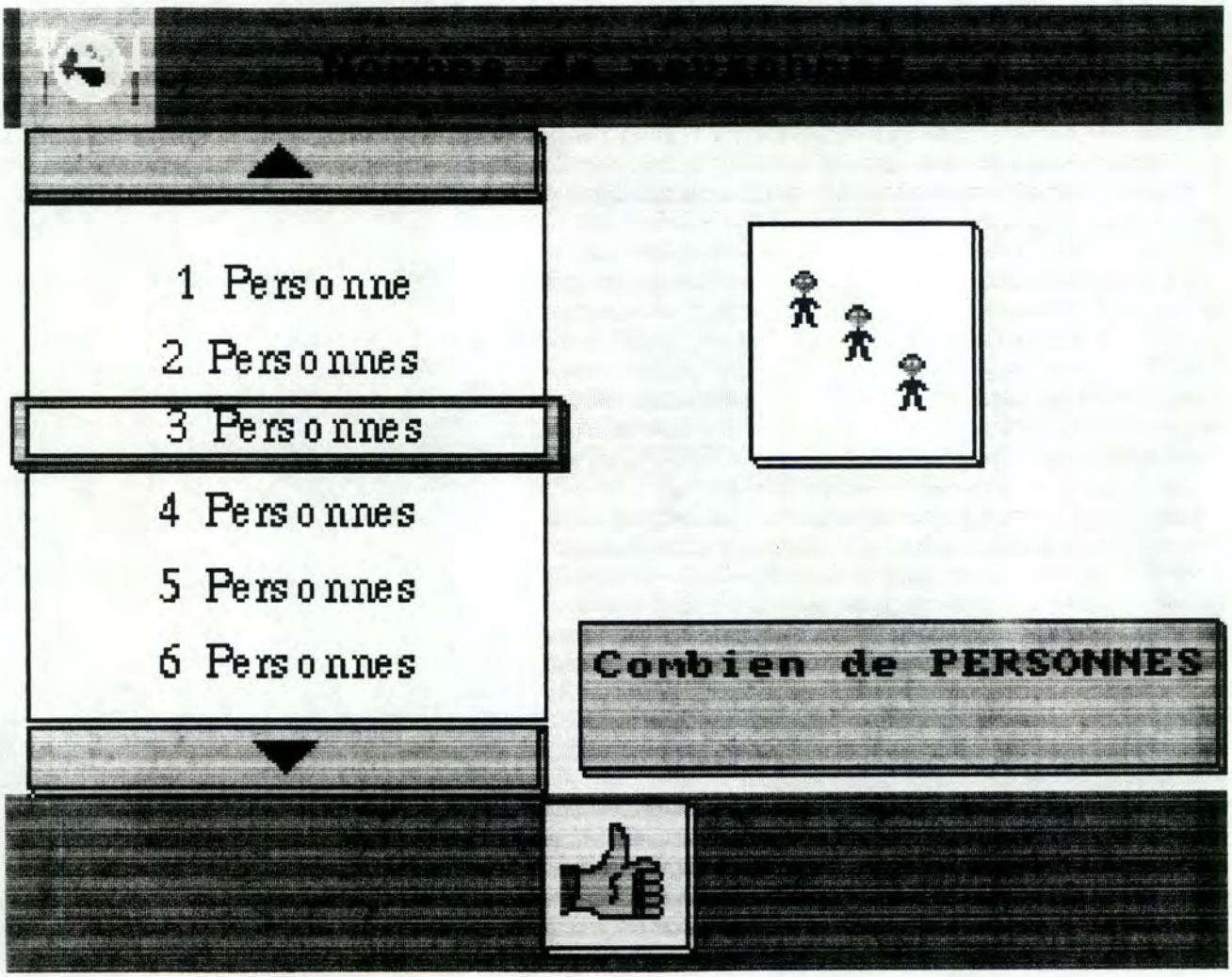
Pour choisir une fonctionnalité, cliquez sur son bouton. Ce dernier est alors mis en évidence et le nom de la fonctionnalité est prononcé. En cliquant à nouveau sur ce bouton, le nom est encore prononcé.

Suite à votre premier choix, le bouton  apparaît. Vous devez confirmer votre choix de fonctionnalité en cliquant sur ce bouton.

Tant que votre choix d'une fonctionnalité n'est pas confirmé, il vous est possible d'en choisir une autre, ce qui a pour effet de d'annuler automatiquement la sélection précédente.

II.2.2.2 Les écrans avec dérouleur.

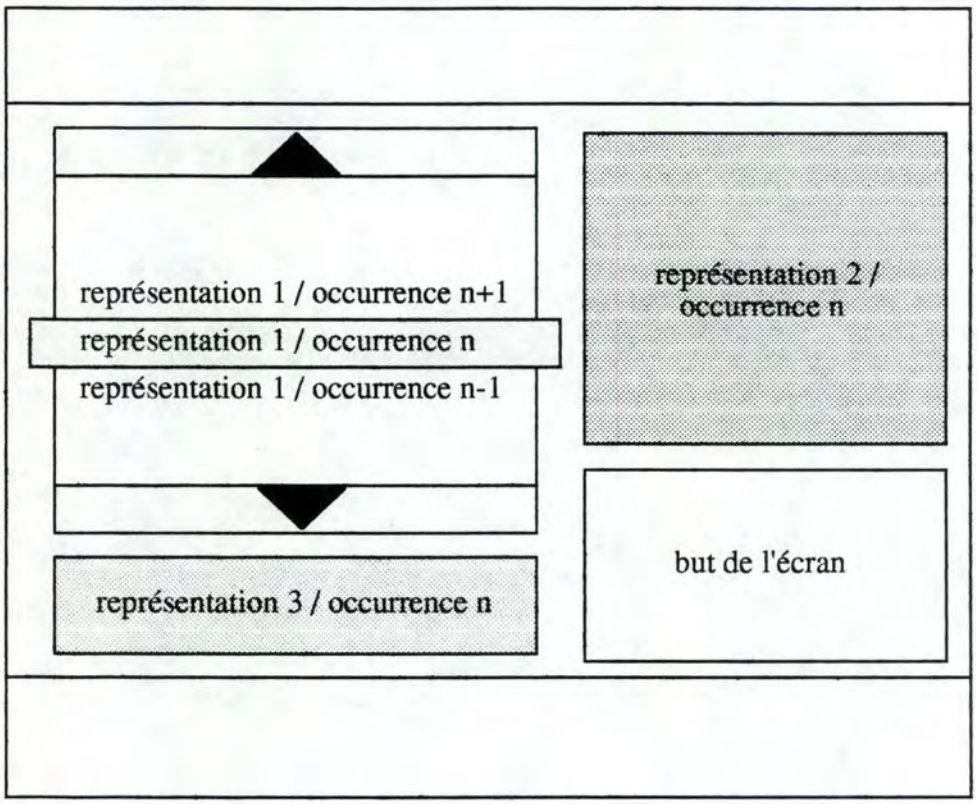
En cours de travail, vous devrez souvent utiliser des dérouleurs. Ce sera le cas pour choisir une date, un nombre de personnes, un thème, un plat, une quantité, pour la consultation d'un menu et d'une liste de produits entrant dans la composition d'un menu.



Après l'affichage d'un tel écran, son but est prononcé ("Choisis un thème", "Voici ton menu", ... en fonction du travail à réaliser sur l'écran).

Il est à tout moment possible de faire prononcer à nouveau ce but, en cliquant sur le bouton sur lequel il est écrit.

Les principes d'utilisation sont les mêmes pour tous les dérouleurs. Voyons-les sur base d'une représentation logique d'un écran.



Tout dérouleur comprend une flèche haut, une flèche bas et une fenêtre.

Les objets représentés dans un dérouleur sont tous de la même nature : des plats, des produits, ...

L'occurrence de l'objet dont la représentation est dans la fenêtre, est l'occurrence courante.

Les objets ont en général plusieurs représentations visuelles possibles, ceci afin d'aider au mieux les utilisateurs dans leur travail. Un plat, par exemple, est représenté par son nom et par un dessin. Nous verrons plus tard les différents types de représentation des objets.

Un dérouleur ne contient qu'un type de représentation : la représentation textuelle.

Les autres représentation visuelles de l'occurrence courante, uniquement, figurent dans les zones représentation2 et représentation3 (il y a maximum trois types de représentations possibles par objet).

Selon le travail à effectuer sur l'écran, il se peut qu'au départ, la fenêtre du dérouleur n'apparaisse pas et qu'il n'y ait donc aucune occurrence courante.

Pour rendre une première occurrence courante, il suffit de cliquer sur la flèche haut ou sur la flèche bas.

Lorsqu'une occurrence devient courante, son nom est prononcé et ses éventuelles représentations, autres que la représentation textuelle, sont affichées dans les zones représentation2 et représentation3.

Il est à tout moment possible de faire à nouveau prononcer le nom de l'occurrence courante, en cliquant sur n'importe laquelle de ses représentations.

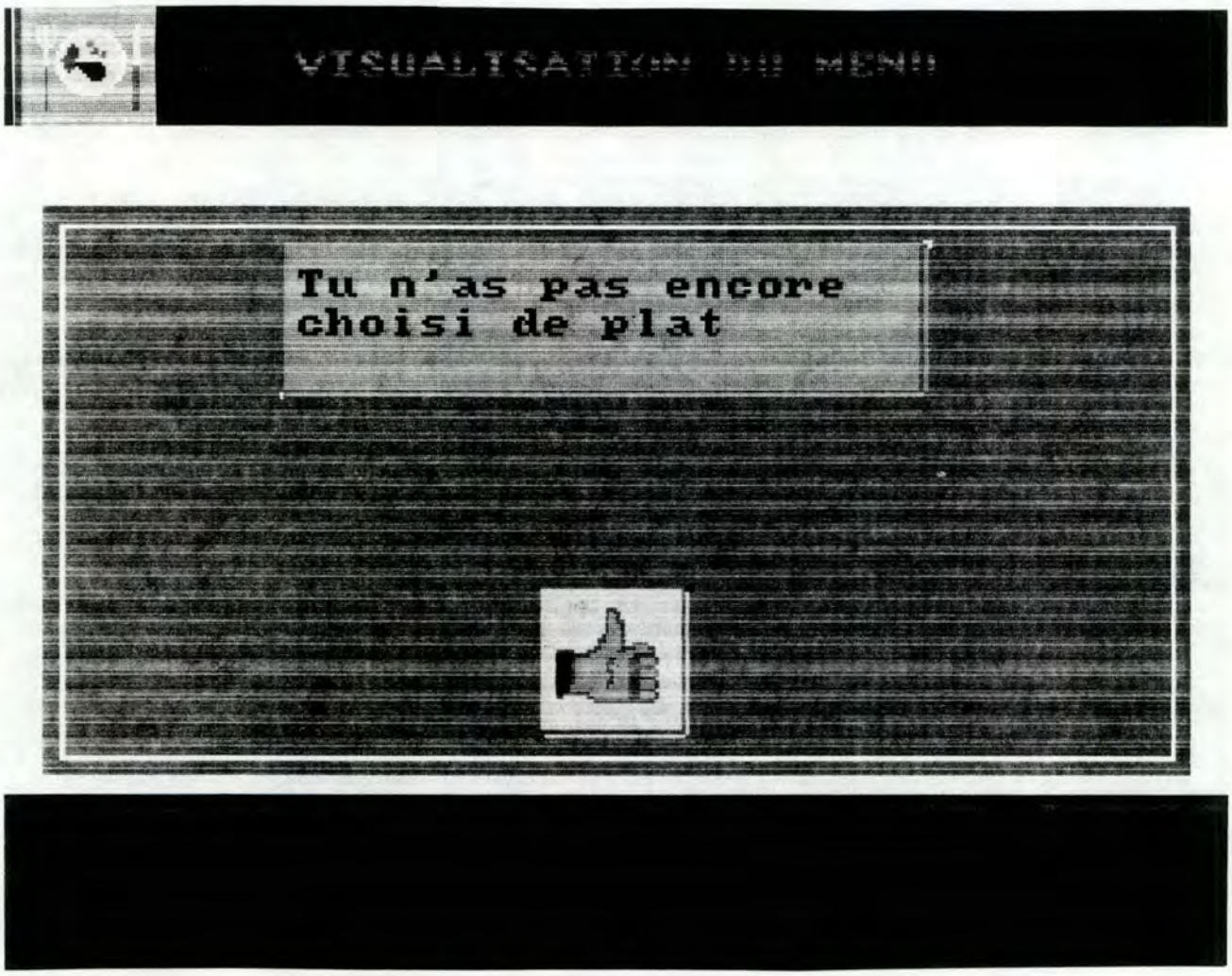
Cliquer sur la flèche haut a pour effet de rendre courante l'occurrence n+1 (ce qui implique un changement du contenu de la fenêtre et des zones représentation2 et représentation3 et la prononciation de la représentation sonore de l'occurrence n+1). Cliquer sur la flèche bas produit le même effet mais concernant l'occurrence n-1.

L'occurrence courante est la seule qui puisse être l'objet d'une commande :

- lorsqu'il s'agit d'un écran pour une consultation d'un menu ou d'une liste de produits pour un menu, la seule commande possible est la demande de prononciation du nom de l'occurrence, par cliking sur une de ces représentations.
- lorsqu'il s'agit d'un écran pour un choix (d'un thème, d'une date, ...), deux commandes sont possibles : la demande de la prononciation du nom ou la sélection de l'occurrence en cliquant sur  .

II.2.2.3 Les fenêtres pour messages du système.

En cours de travail, des messages du système peuvent apparaître en suraffiché, dans une fenêtre, sur l'écran courant, sur fond noir, vous informant d'un fait ou vous demandant d'accomplir une action



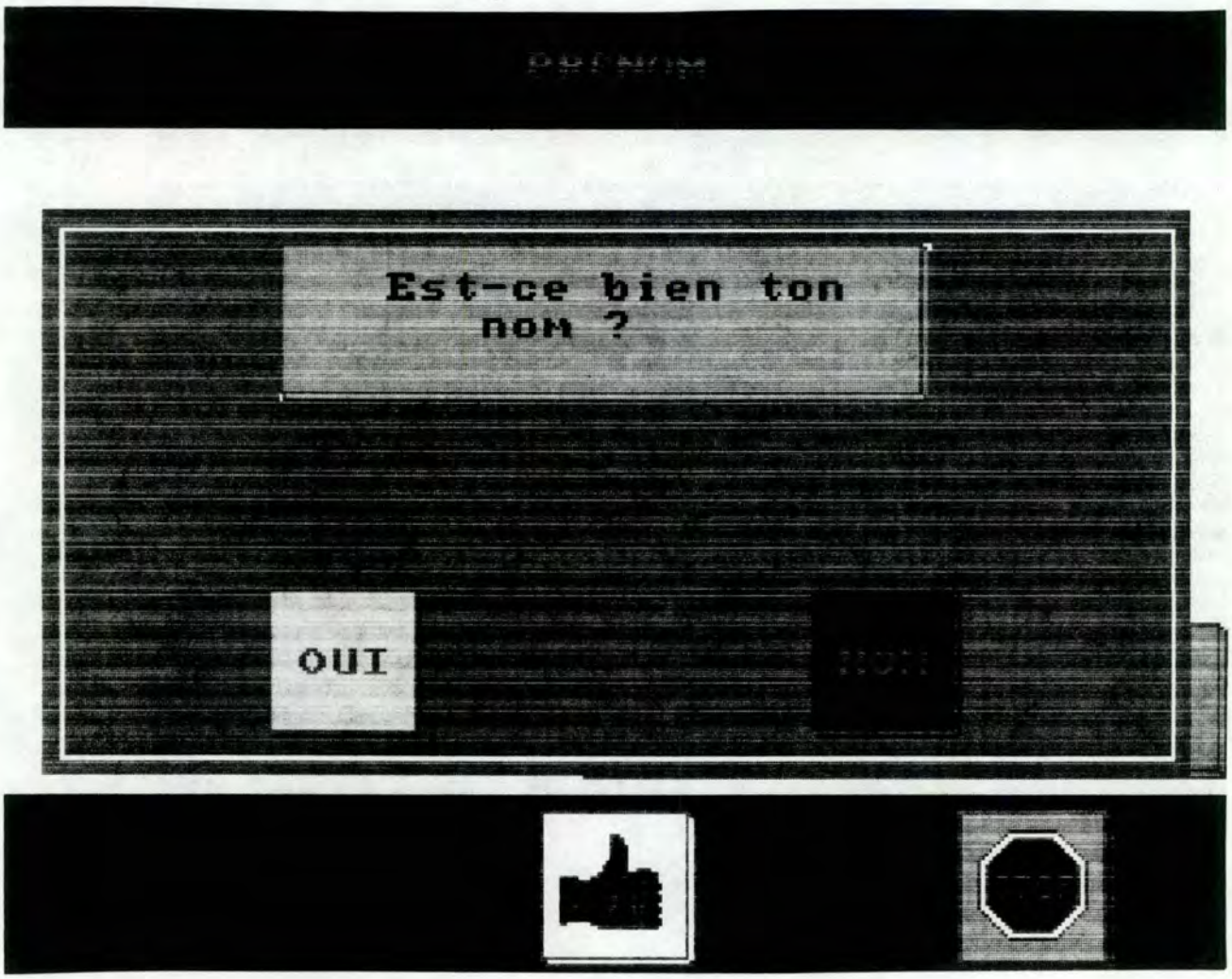
Lorsqu'un tel message vous parvient, son contenu est toujours prononcé, et vous pouvez demander la répétition de cette prononciation en cliquant n'importe où dans le fenêtre.

Vous devez ensuite, le cas échéant, accomplir ce que le message vous demande et enfin, de toute façon, indiquer au système que vous avez pris connaissance du message, et ce, en cliquant sur >>> . Cela a pour effet

de faire disparaître la fenêtre, et de vous laisser reprendre votre travail où vous l'aviez laissé avant d'être interrompu. Parfois, selon les circonstances, ce sera l'inverse, vous devrez signifier que vous avez pris connaissance du message, et vous devrez ensuite accomplir ce que le message demande.

II.2.2.4 Les fenêtres pour confirmation d'un fait.

En cours de travail, des messages du système peuvent apparaître en suraffiché, dans une fenêtre, sur l'écran courant, sur fond noir, vous demandant de confirmer ou d'infirmer un fait toujours clairement précisé.



Lorsqu'un tel message vous parvient, son contenu est toujours prononcé, et vous pouvez demander la répétition de cette prononciation en cliquant sur le bouton sur lequel le message est écrit.

Vous devez ensuite répondre au message en cliquant sur  ou  , à votre convenance. Votre réponse a pour effet de faire disparaître le message, et de vous laissez poursuivre votre travail, en fonction de votre réponse bien entendu.

II.3 DEMARRAGE DU PROGRAMME.

-1 : Mettez l'ordinateur et le moniteur sous tension, et, si vous avez l'intention de demander des impressions au cours de la séance de travail, veillez également à ce que l'imprimante soit aussi sous tension, qu'elle soit bien connectée à l'ordinateur qu'elle soit correctement initialisée (référez vous pour ce faire au manuel d'utilisation de cette dernière), et qu'elle soit approvisionnée en papier.

-2 : Introduisez la disquette bleue ("PROGRAMME") dans le lecteur interne, et la disquette jaune ("BD") dans le lecteur externe. Attendez, quelques dizaines de secondes, que s'affiche l'Ecran 0 de présentation du logiciel, et encore quelques secondes que s'affiche l'Ecran 1 d'introduction du nom. Nous verrons ces écrans ci-dessous.

II.4 LES ECRANS.

Au cours de ce point, nous allons voir le plan du dialogue, c'est-à-dire la succession des écrans qui aura lieu lors du dialogue entre vous et l'ordinateur, ainsi que toutes les possibilités d'action que vous avez sur chaque écran.

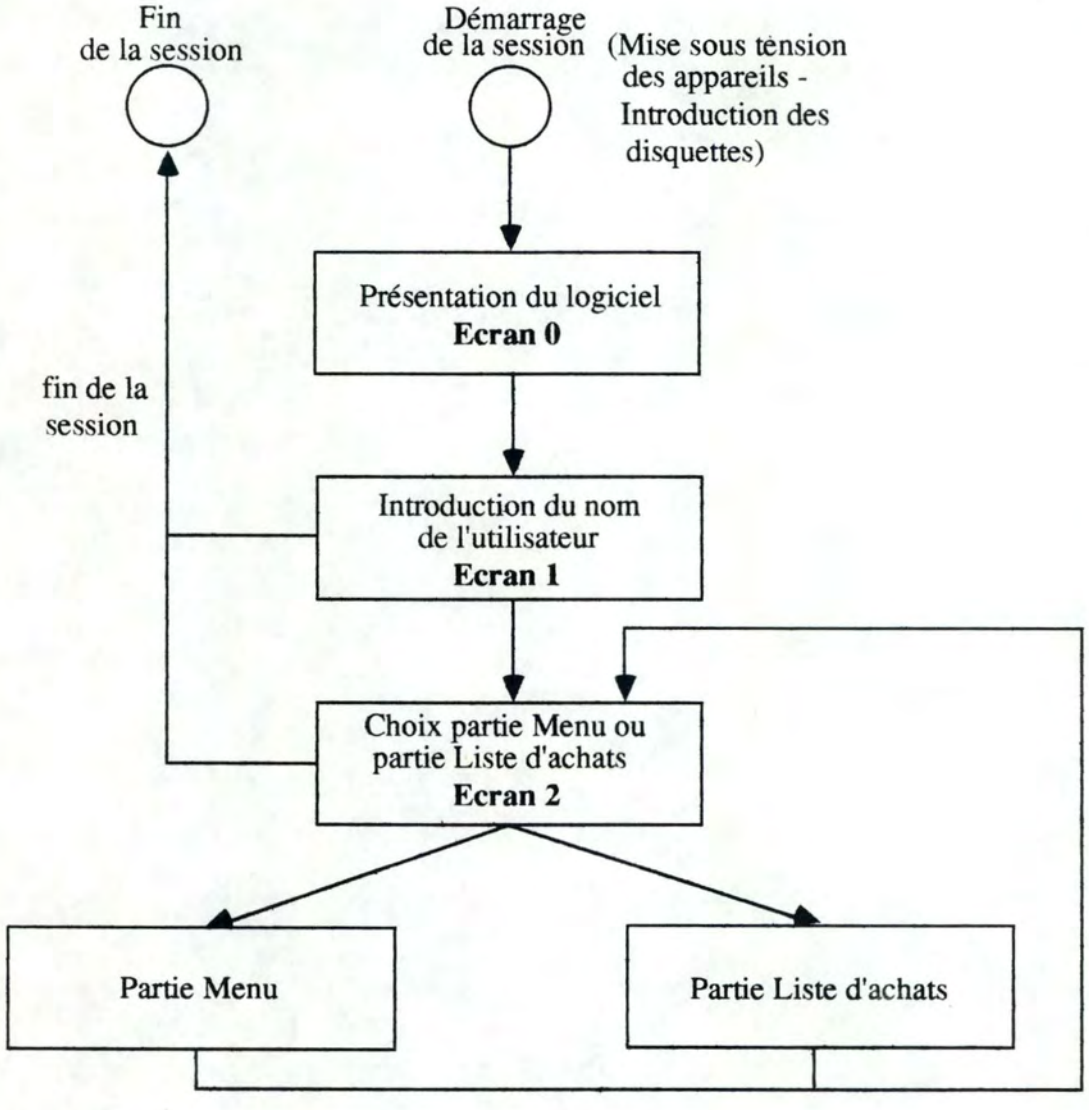
II.4.1 LE PLAN DU DIALOGUE.

Le plan du dialogue vous est présenté au moyen de plusieurs graphiques. Ces graphiques sont simples et vous permettront de vous situer rapidement dans votre travail. Dans ces graphiques, ne sont repris que les écrans qui apparaissent lors d'un dialogue entre vous et l'ordinateur. Les fenêtres, qui se superposent aux écrans, seront vues en temps opportun.

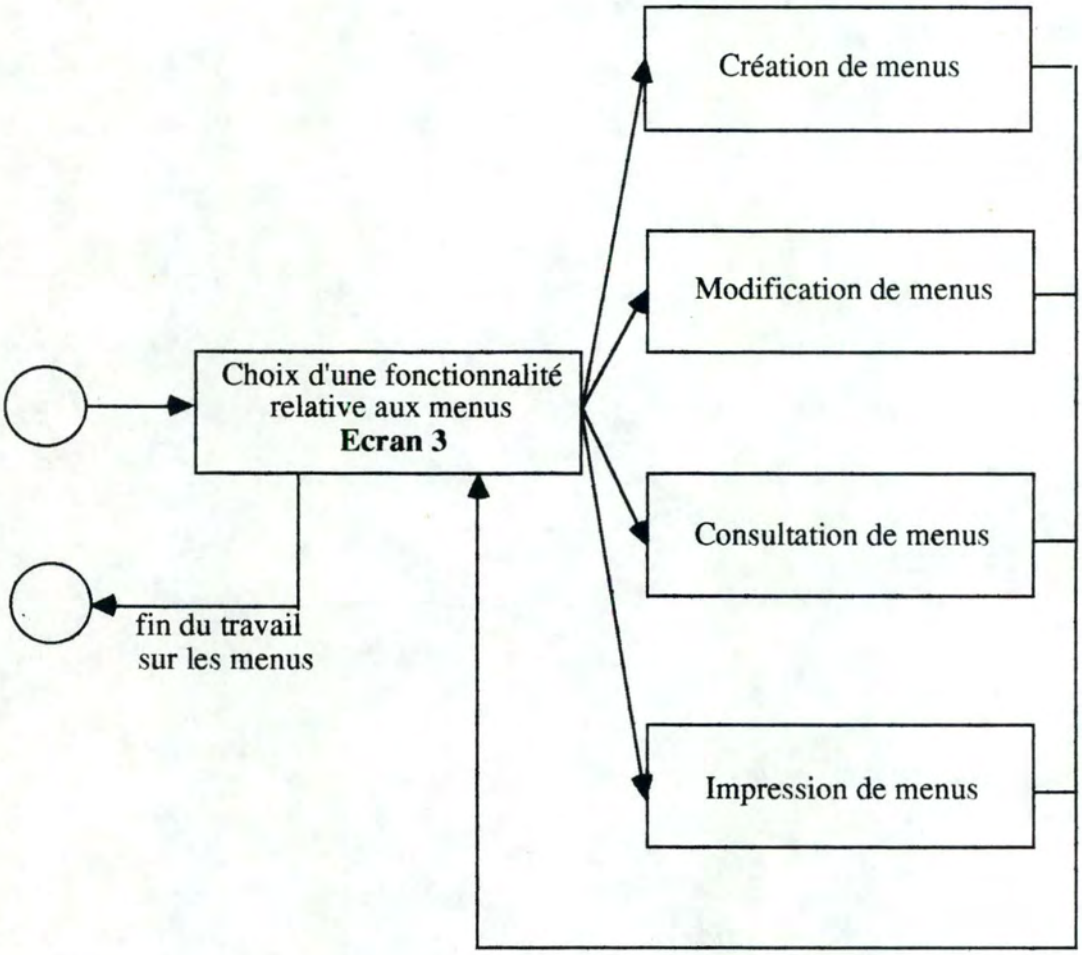
Les conventions suivantes sont adoptées pour les graphiques :

- les cercles symbolisent l'entrée et la sortie d'un graphique.
- les rectangles symbolisent un écran ou une succession d'écrans. Ils symbolisent un écran lorsqu'un numéro d'écran y est précisé; vous vous référez alors à l'explication de cet écran, que nous verrons ultérieurement. Lorsqu'aucun numéro d'écran n'est précisé, cela signifie qu'il y une succession d'écrans pour laquelle un graphique distinct existe; référez-vous à ce graphique.
- les flèches symbolisent le passage automatique d'un écran à un autre. Lorsque nous étudierons chaque écran en particulier, nous verrons comment quitter chacun de ceux-ci, le départ d'un écran impliquant le passage automatique à l'écran suivant dans le graphique.

II.4.1.1 PLAN GLOBAL.

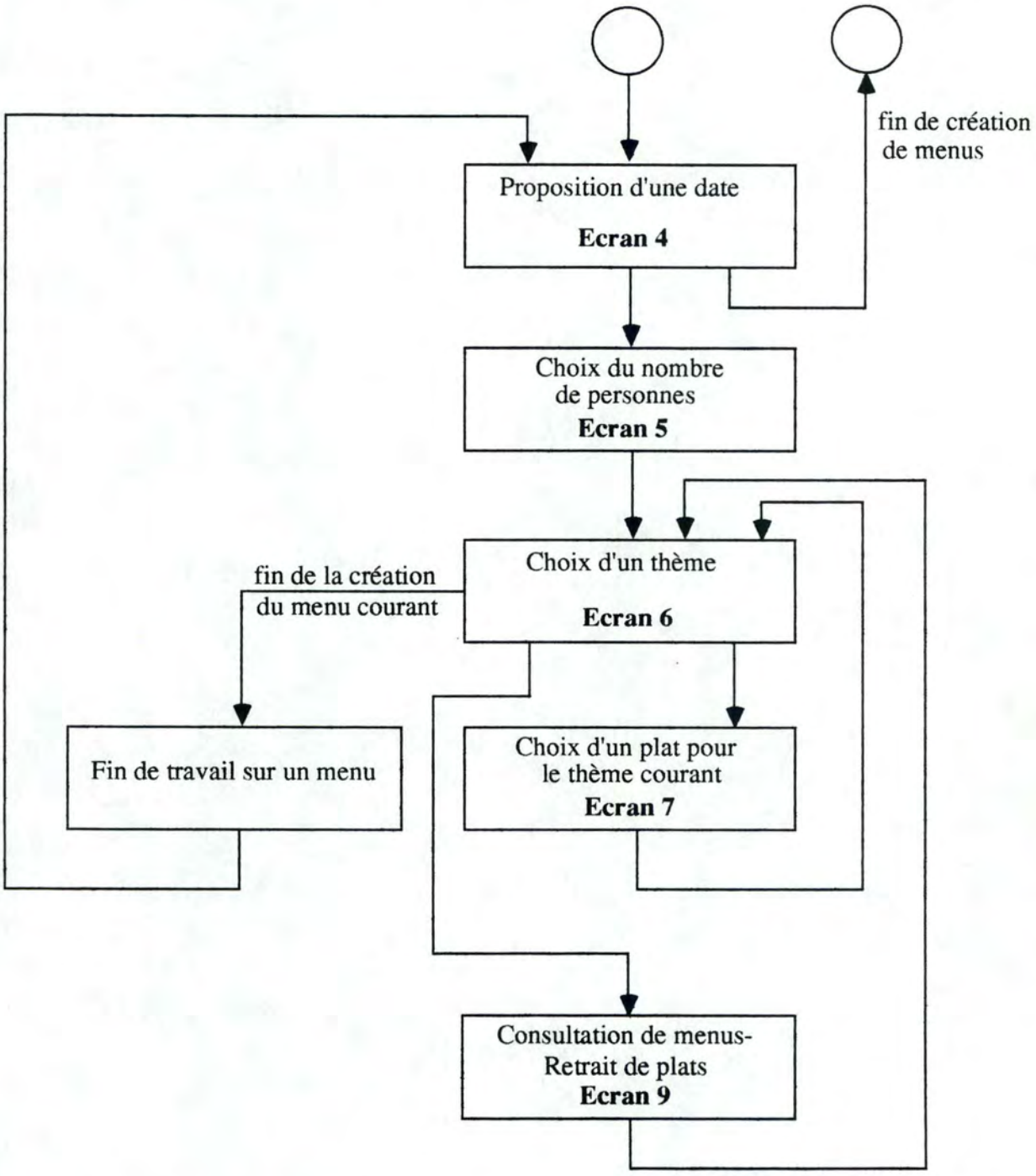


II.4.1.2 PARTIE MENU.



II.4.1.3 CREATION DE MENUS.

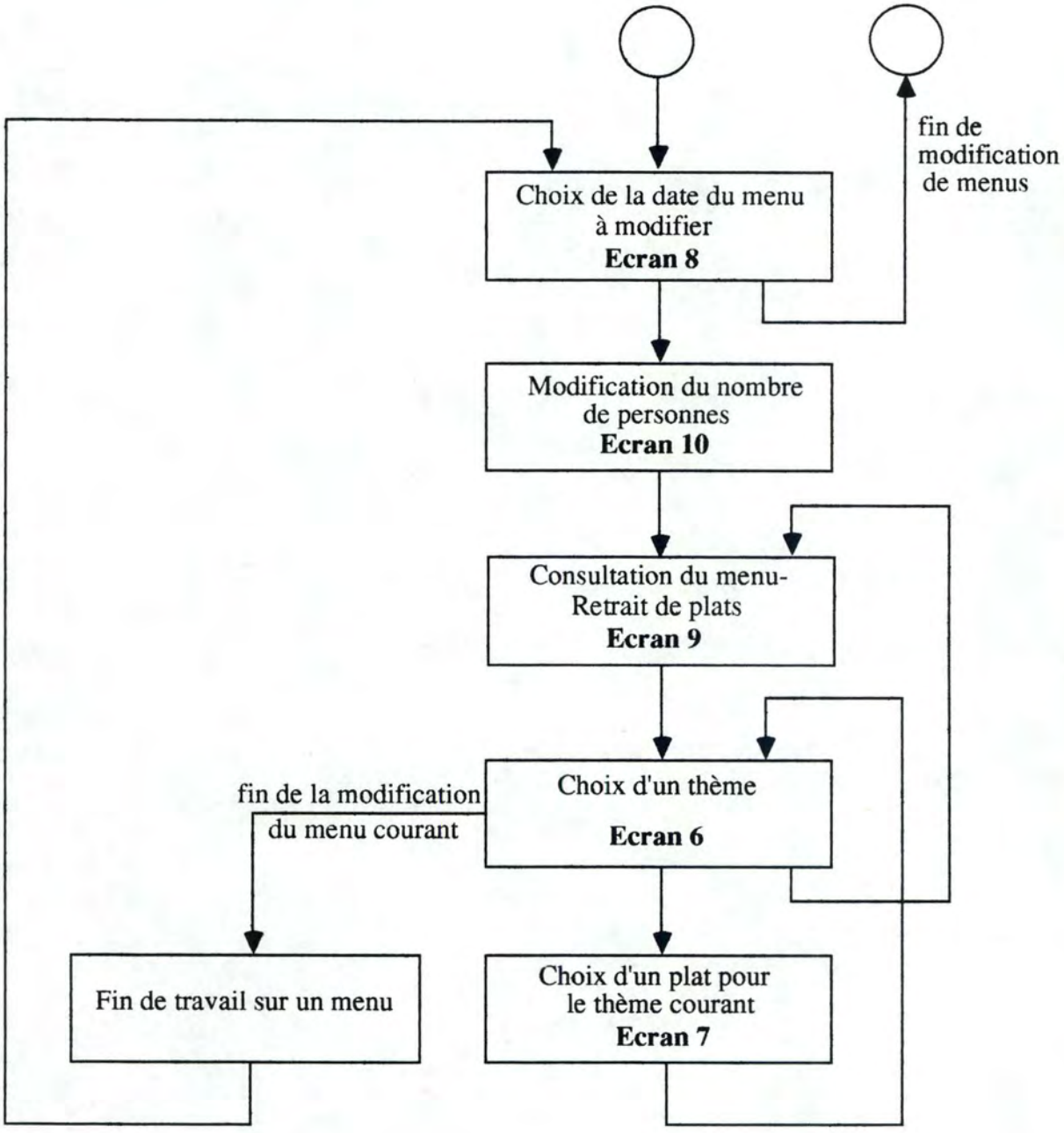
-a : Premier scénario de création de menus.



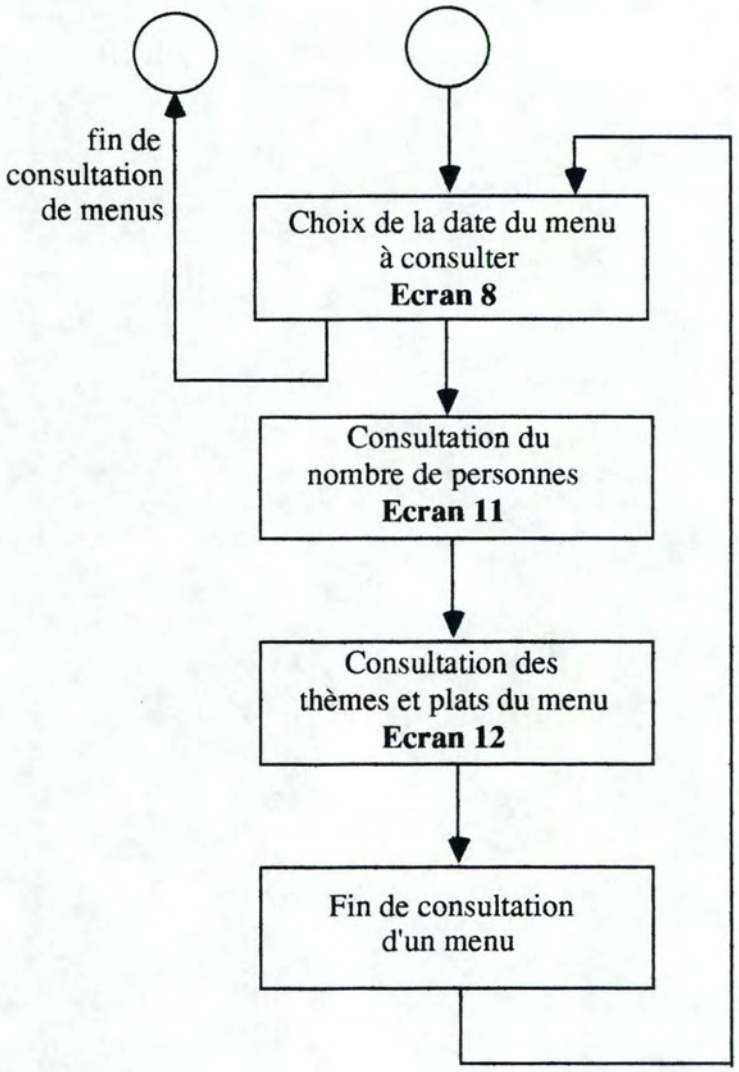
-b : Second scénario de création de menus.

Idem premier scénario sauf voir **Ecran 4bis** au lieu de Ecran 4.

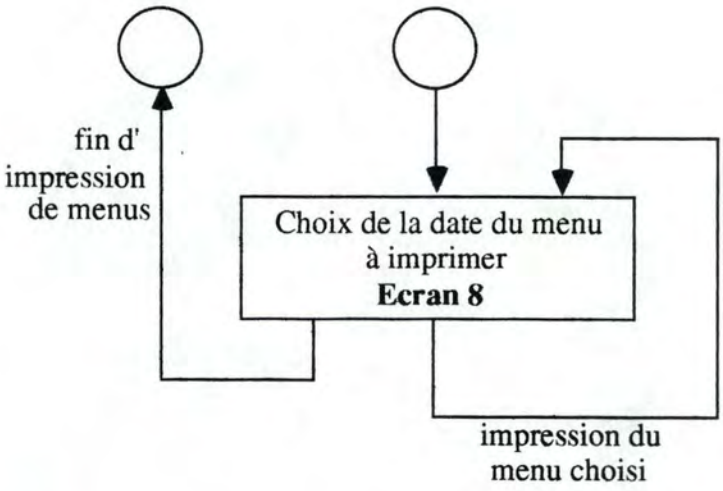
II.4.1.4 MODIFICATION DE MENUS.



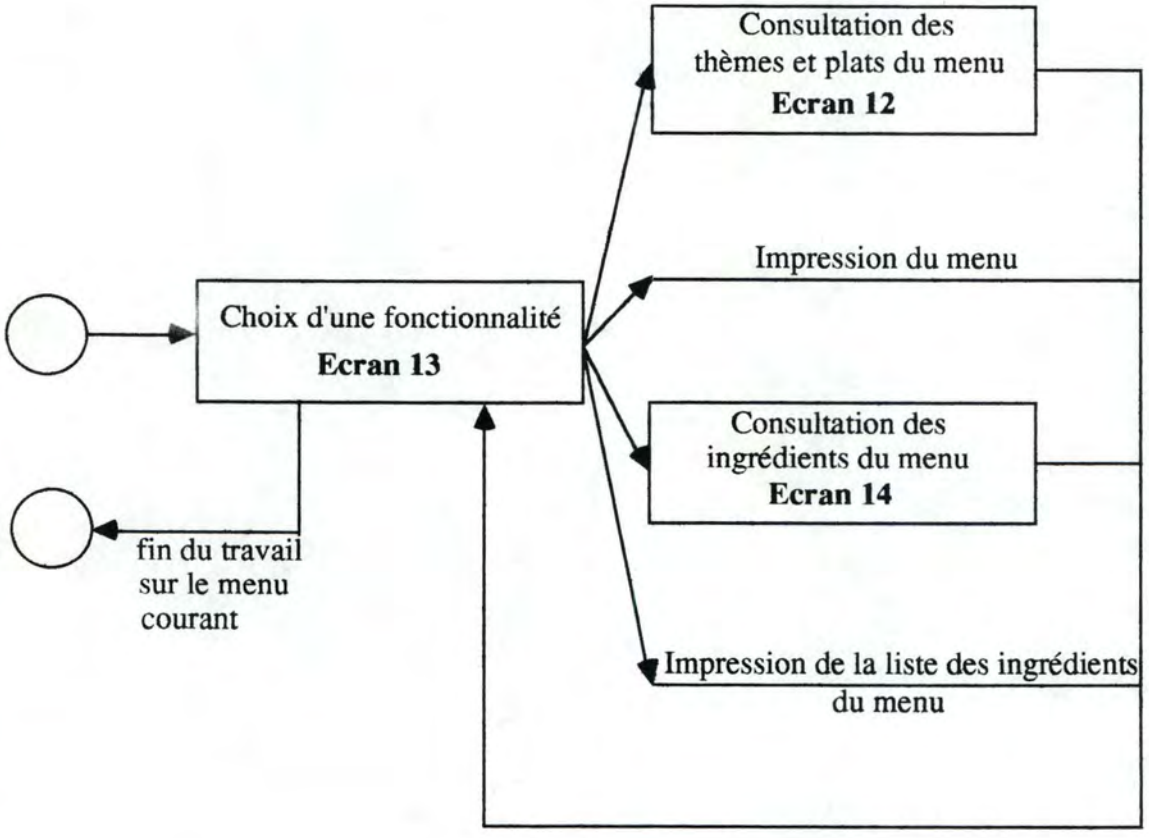
II.4.1.5 CONSULTATION DE MENUS.



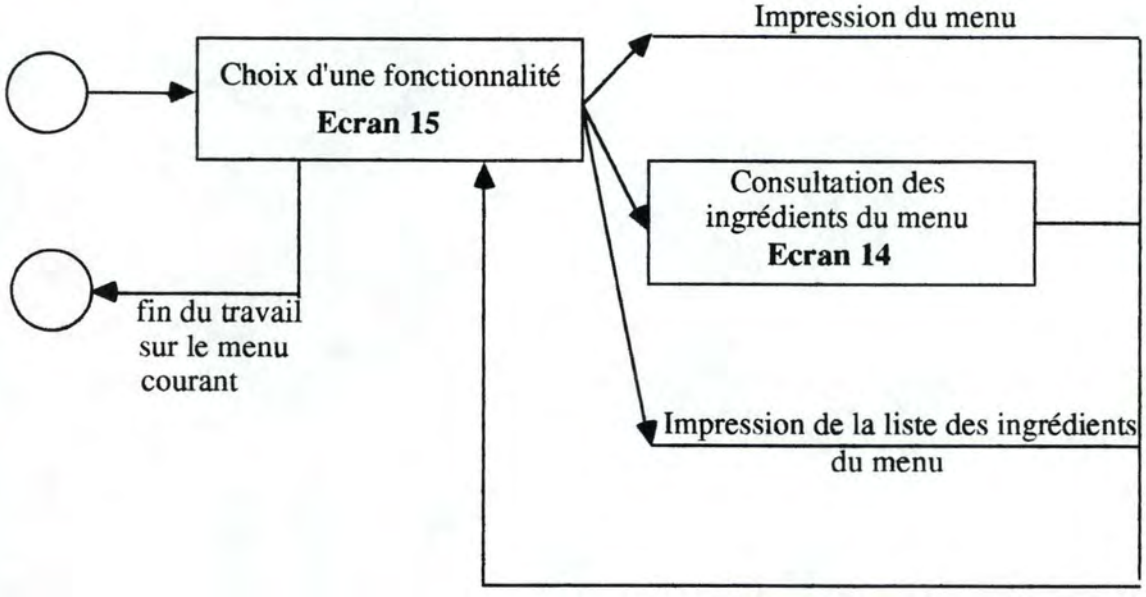
II.4.1.6 IMPRESSION DE MENUS.



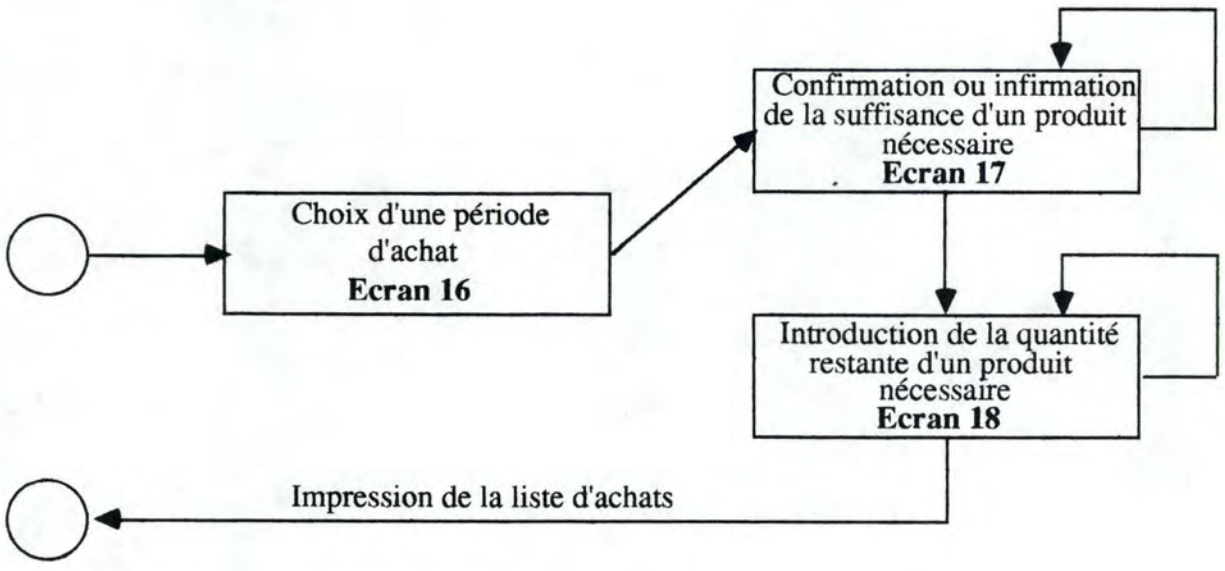
II.4.1.7 FIN DE TRAVAIL SUR UN MENU.



II.4.1.8 FIN DE CONSULTATION D'UN MENU.



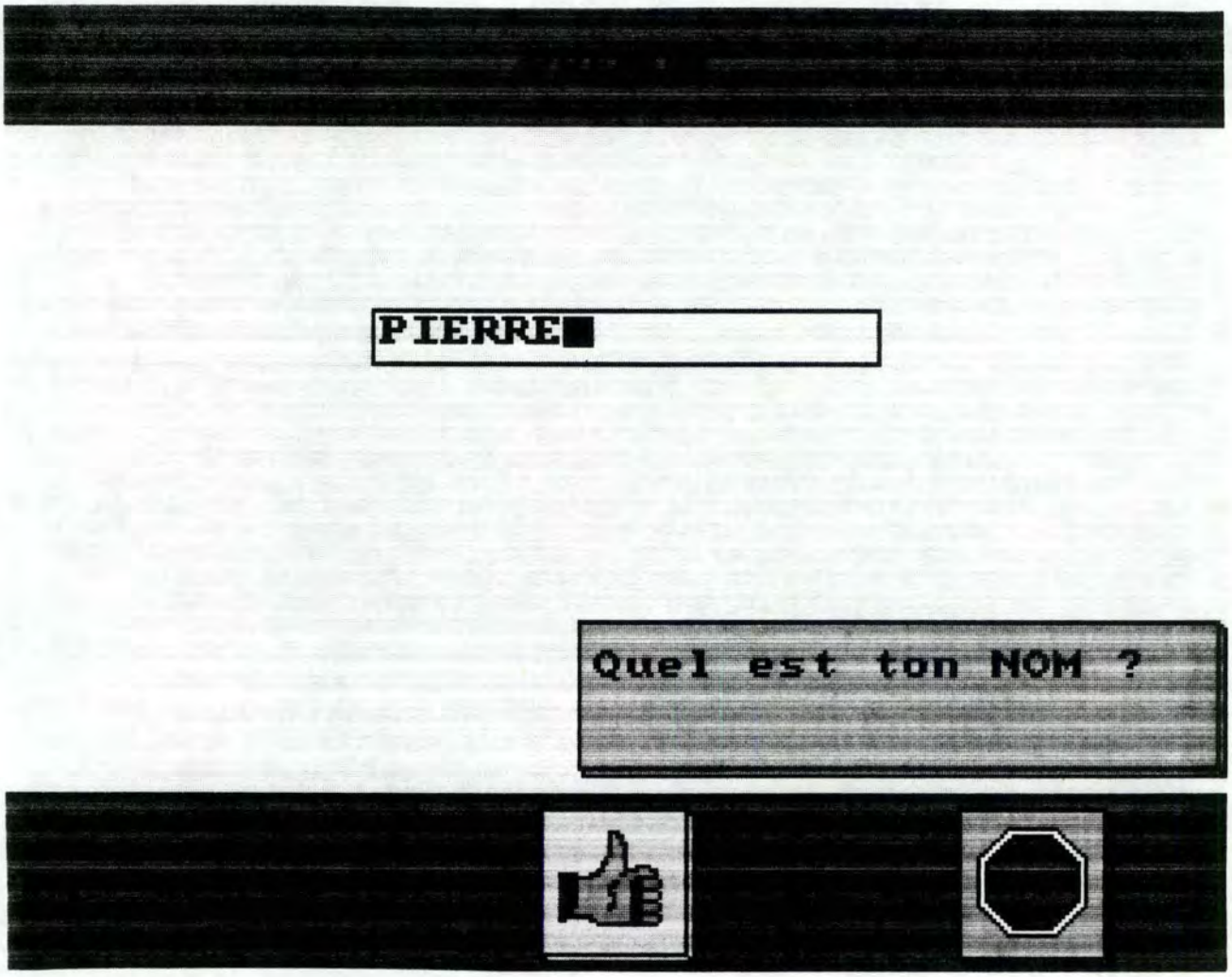
II.4.1.9 PARTIE LISTE D'ACHATS.



II.4.2 LES ECRANS.

II.4.2.1 ECRAN 1.

L'écran 1 vous invite à déclarer votre nom. Lorsque cet écran s'affiche, il vous propose le nom du propriétaire de la disquette "BD". Si c'est bien la vôtre, cliquez directement sur  , sinon, effacez le nom à l'aide de la touche <←>, et introduisez le vôtre. Vous devez réaliser cela à l'aide des touches du clavier, l'introduction se faisant indifféremment en majuscule ou en minuscule.



Si vous introduisez vous-même un nom différent de celui qui vous est proposé, vous devez bien sûr, avant tout, introduire la disquette "BD"

correspondant qui correspond à ce nom, faute de quoi vous ne pourrez pas utiliser le logiciel. Si le nom que vous avez introduit ne correspond pas à celui connu de la disquette "BD", vous recevrez un message du système (fenêtre 1).

Si vous êtes d'accord avec le nom qui vous est proposé, ou lorsque vous l'avez modifié, vous cliquez sur  , et une confirmation vous est demandée (fenêtre 13).

Vous pouvez quitter le programme, en cliquant sur 

, auquel cas une confirmation vous sera demandée (fenêtre 2).

Une fois votre nom correctement introduit, vous passez automatiquement à l'écran 2.

II.4.2.2 ECRAN 2.

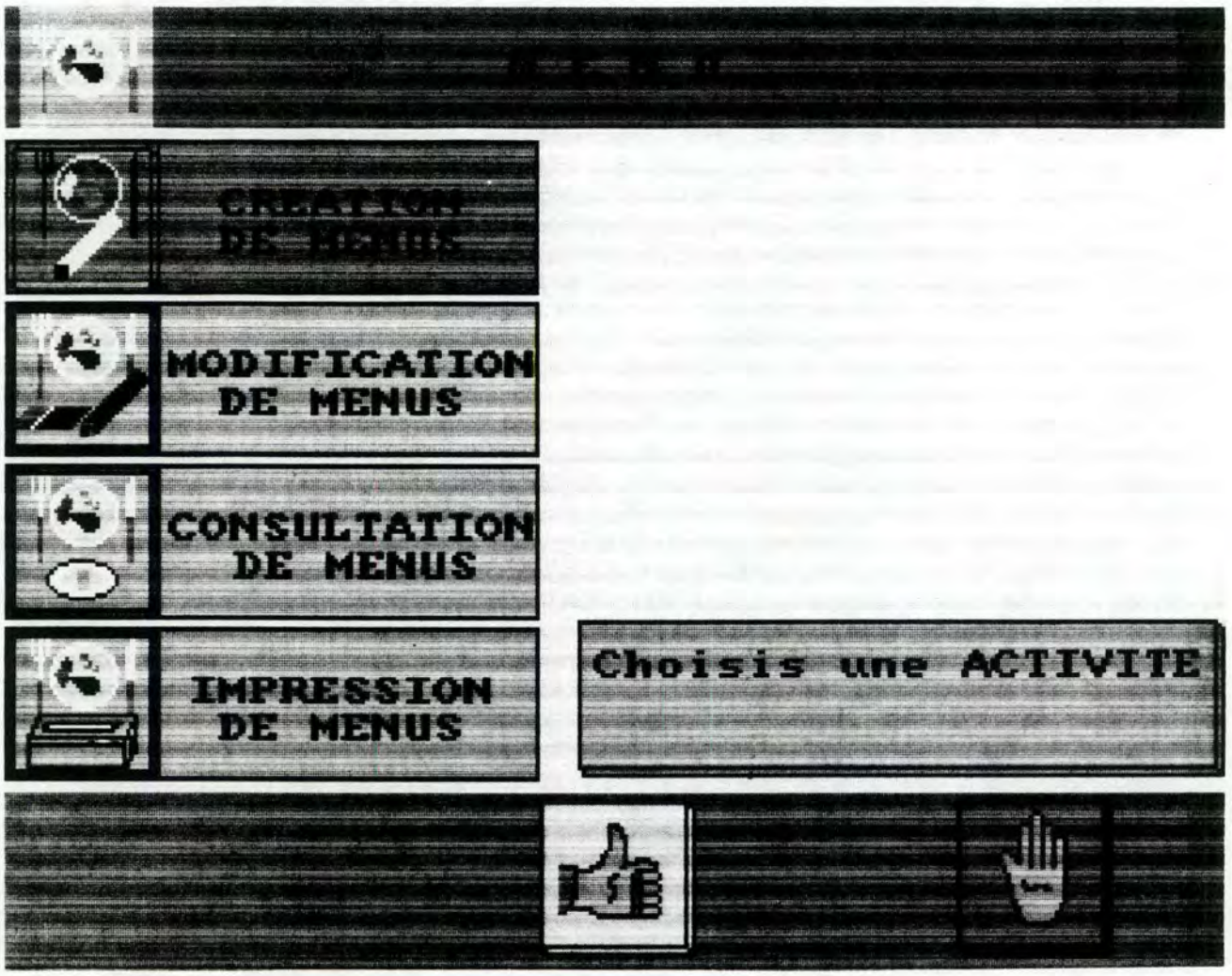
Sur l'écran 2, vous choisissez entre travailler sur les menus et travailler sur les listes d'achats.

Le bouton  vous permet de quitter le programme; une confirma

tion vous sera demandée (fenêtre 2).

II.4.2.3 ECRAN 3.

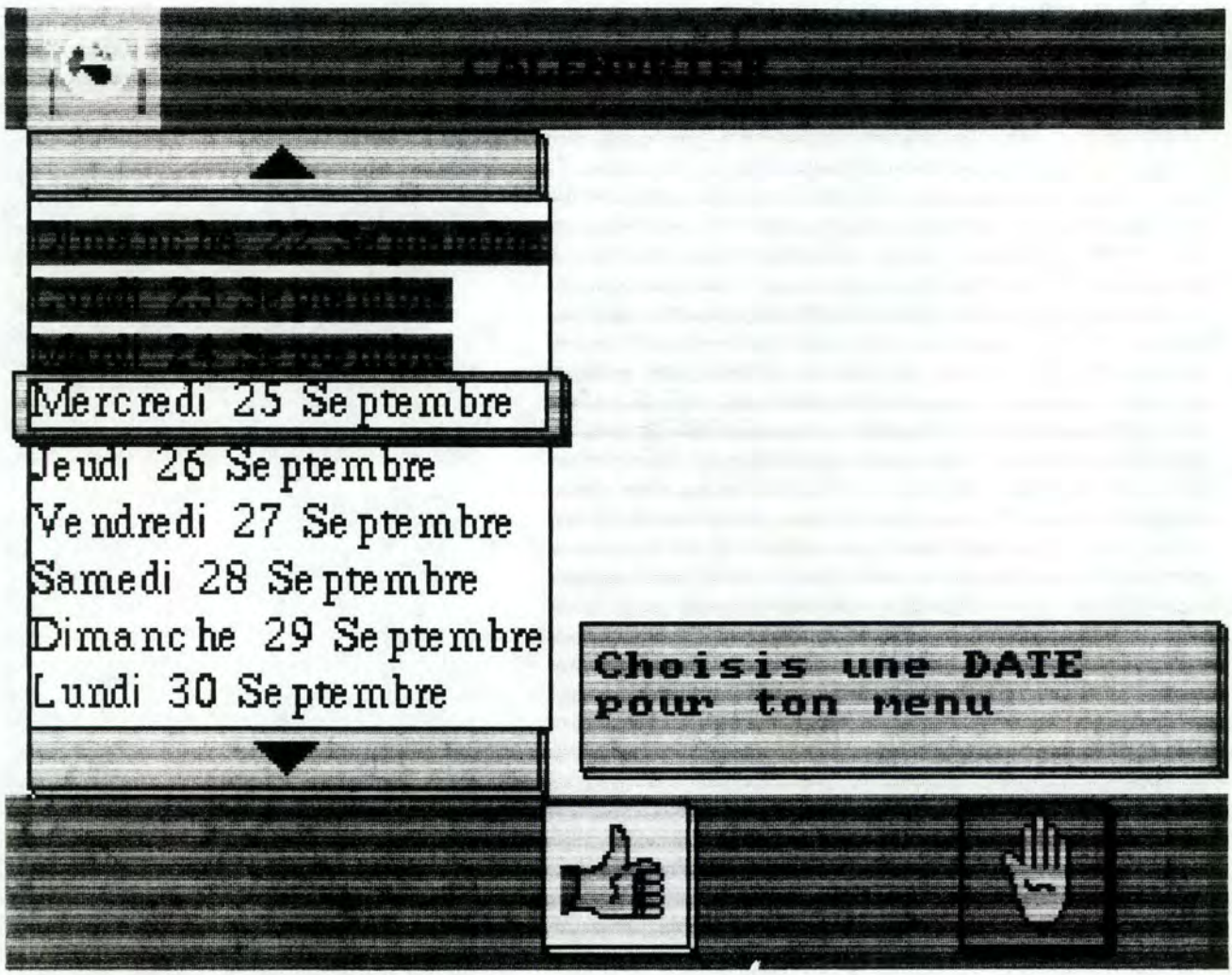
Sur l'écran 3, vous devez tout d'abord choisir une fonctionnalité relative aux menus.



Le bouton  vous permet de quitter l'écran, une confirmation vous est toutefois demandée (fenêtre 3).

II.4.2.4 ECRAN 4.

L'écran 4 vous propose, dans un dérouleur, un ensemble de dates, dont la première est la date du jour courant.



Les dates mises en évidence sont les dates de jours déjà traités, c'est-à-dire pour lesquels vous avez déjà créé un menu ou pour lesquels vous avez déjà signifié que vous ne désiriez pas créer de menu.

Le dérouleur ne dispose pas de flèches, vous ne pouvez pas le manipuler, puisqu'avec ce premier scénario, vous n'avez pas le choix des dates. La date dans la fenêtre est la date du jour que vous devez traiter maintenant.

Répondez, en cliquant sur le bouton approprié, si oui ou non vous désirez créer un menu pour le jour dont la date est dans la fenêtre.

Si vous avez répondu "OUI", alors, vous passez à l'écran suivant, sinon, vous restez sur le même écran, mais la date dans la fenêtre est la date suivante.

S'il n'y a plus de jour qu'il soit actuellement possible de traiter, un message vous parvient (fenêtre 3).

Vous pouvez abandonner ce travail de création de menus en cliquant sur  , auquel cas il vous sera demandé de confirmer (fenêtre 4). Vous retournez alors à l'écran précédent.

II.4.2.5 ECRAN 4BIS.

Sur l'écran 4bis, associé au second scénario de création de menus, vous devez choisir vous-même la date du menu à créer, en manipulant le dérouleur.

Les dates de jours déjà traités sont mises en évidence, et vous ne pouvez plus les choisir, la fenêtre ne s'y arrête jamais.

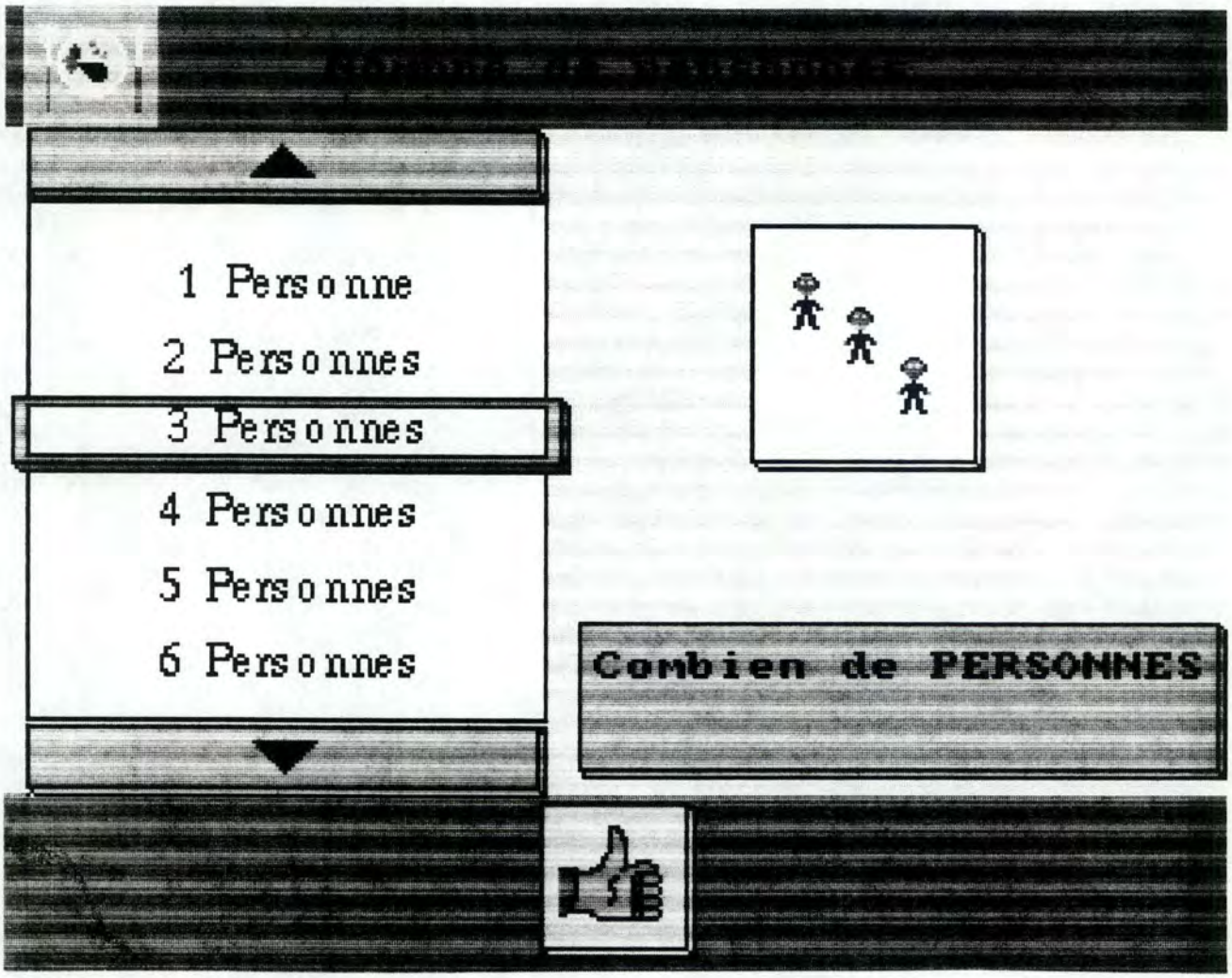
II.4.2.6 ECRAN 5.

Sur l'écran 5, vous devez choisir un nombre de personnes pour lequel le menu sera préparé.

Ce nombre est compris entre 1 et 6.

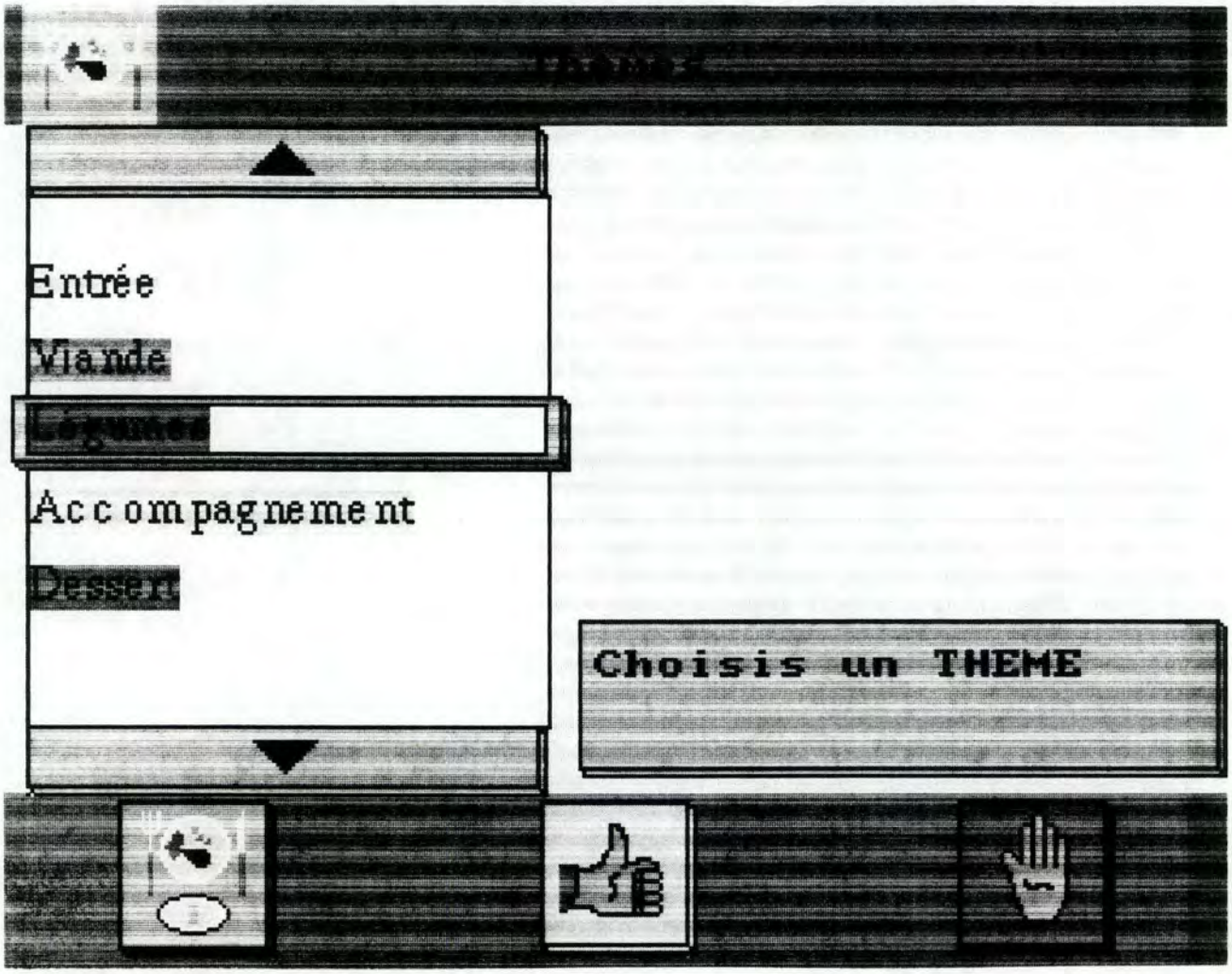
Remarquez que les deux représentations de ces nombres sont les chiffres arabes et les dessins de petits bonhommes.

Votre sélection faite, vous passez à l'écran suivant.



II.4.2.7 ECRAN 6.

Sur l'écran 6, vous choisissez un thème pour le menu.



Les deux représentations pour les thèmes sont le texte (nom du thème) et la couleur (chaque thème a une couleur qui lui est associée).

Remarquez que les seuls thèmes qui vous sont présentés sont ceux qui ne sont pas encore contenus dans le menu sur lequel vous travaillez, ce qui vous empêche de choisir plusieurs fois un même thème pour un même menu. Si votre menu comprend déjà tous les thèmes, vous ne pouvez donc plus en choisir un autre; dans ce cas, vous recevez un message (fenêtre 7).

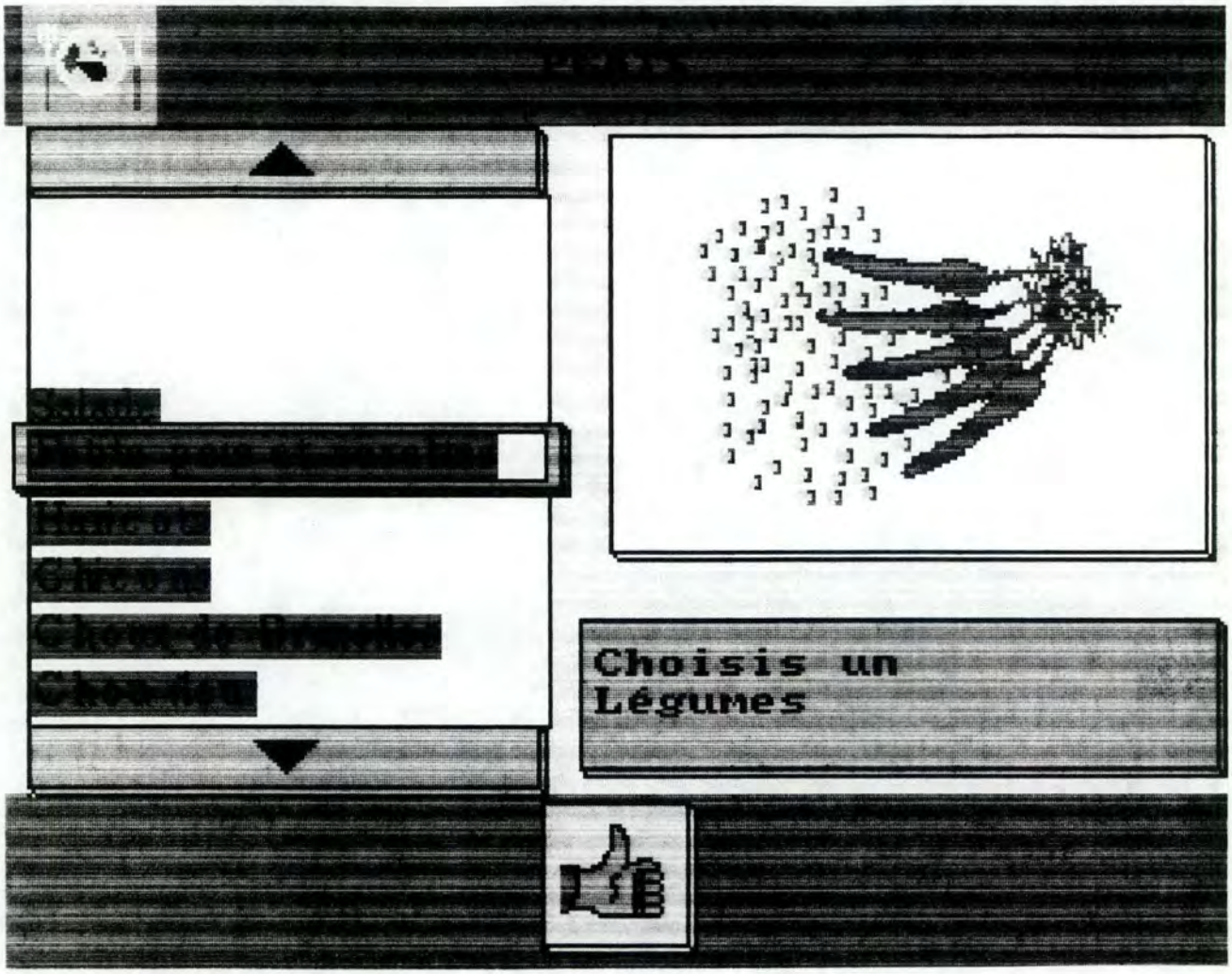
Votre thème choisi, vous passez à l'écran suivant pour le choix d'un plat.

Si vous avez terminez de travailler sur votre menu, vous l'indiquez en cliquant sur  . Si le menu ne comprend aucun plat, il vous est alors demandé de confirmer ce fait (fenêtre 5), sinon vous passez à l'écran suivant.

Cliquer sur le bouton  vous permet de visualiser l'état actuel de constitution de votre menu (écran 9). Si votre menu ne comprend aucun plat, vous recevez un message (fenêtre 6).

II.4.2.8 ECRAN 7.

Sur l'écran 7, vous devez choisir un plat pour le thème que vous venez de sélectionner.

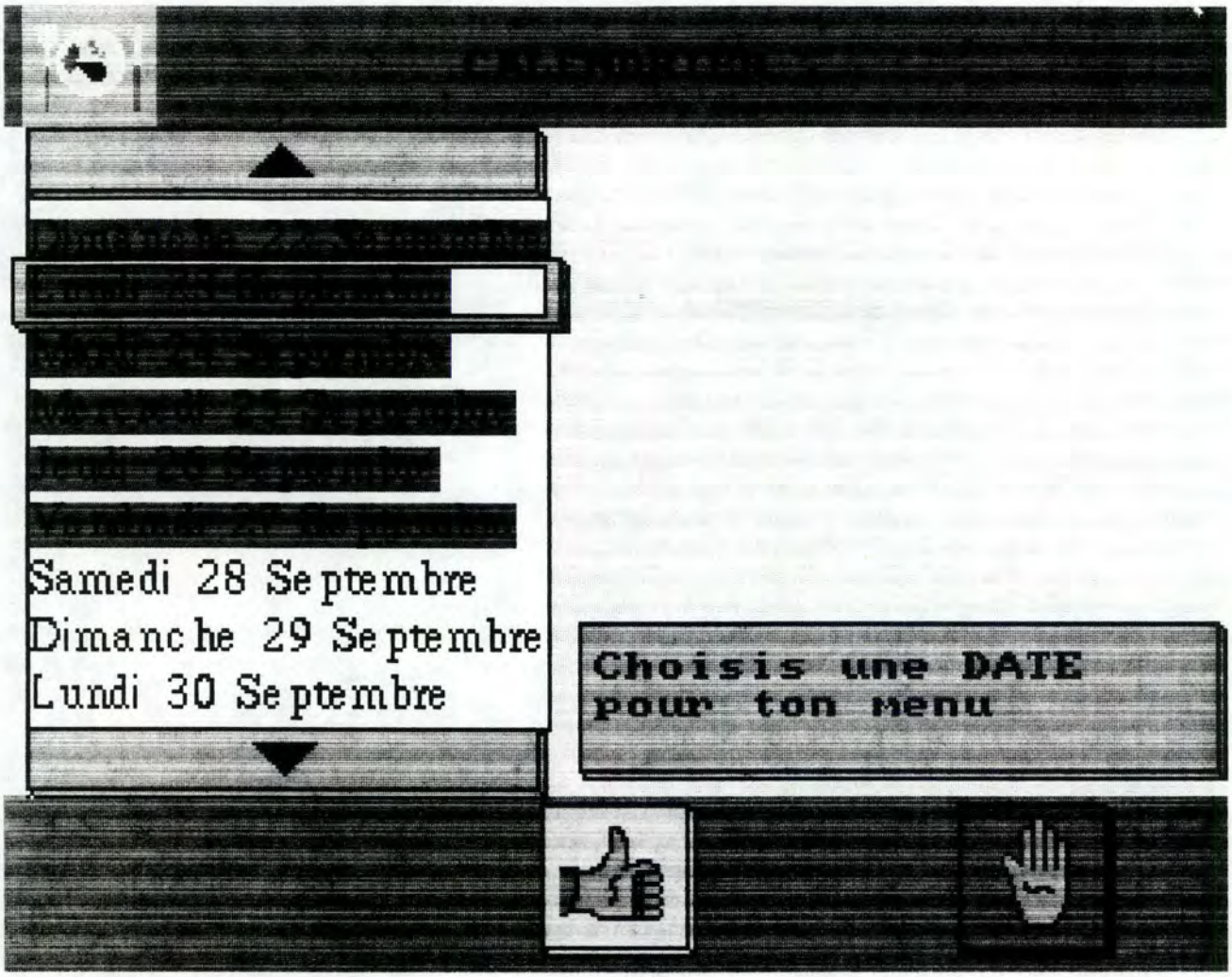


Les plats ont deux représentations : le texte (nom du plat), et le dessin (chaque plat a un dessin qui lui est associé). Remarquez que la couleur du thème pour lequel vous devez choisir un plat, est présente dans le dérouleur.

Une fois un plat choisi, vous retournez à l'écran 6, pour le choix d'un autre thème.

II.4.2.9 ECRAN 8.

Sur l'écran 8, vous choisissez la date d'un menu déjà créé, pour le consulter, le modifier ou l'imprimer.



Les dates de jours déjà traités sont mises en évidence, vous ne pouvez réaliser votre choix que parmi elles, la fenêtre du dérouleur ne s'arrête d'ailleurs que sur elles.

II.4.2.10 ECRAN 9.

Sur l'écran 9, vous pouvez visualiser les thèmes et plats constituant le menu courant.

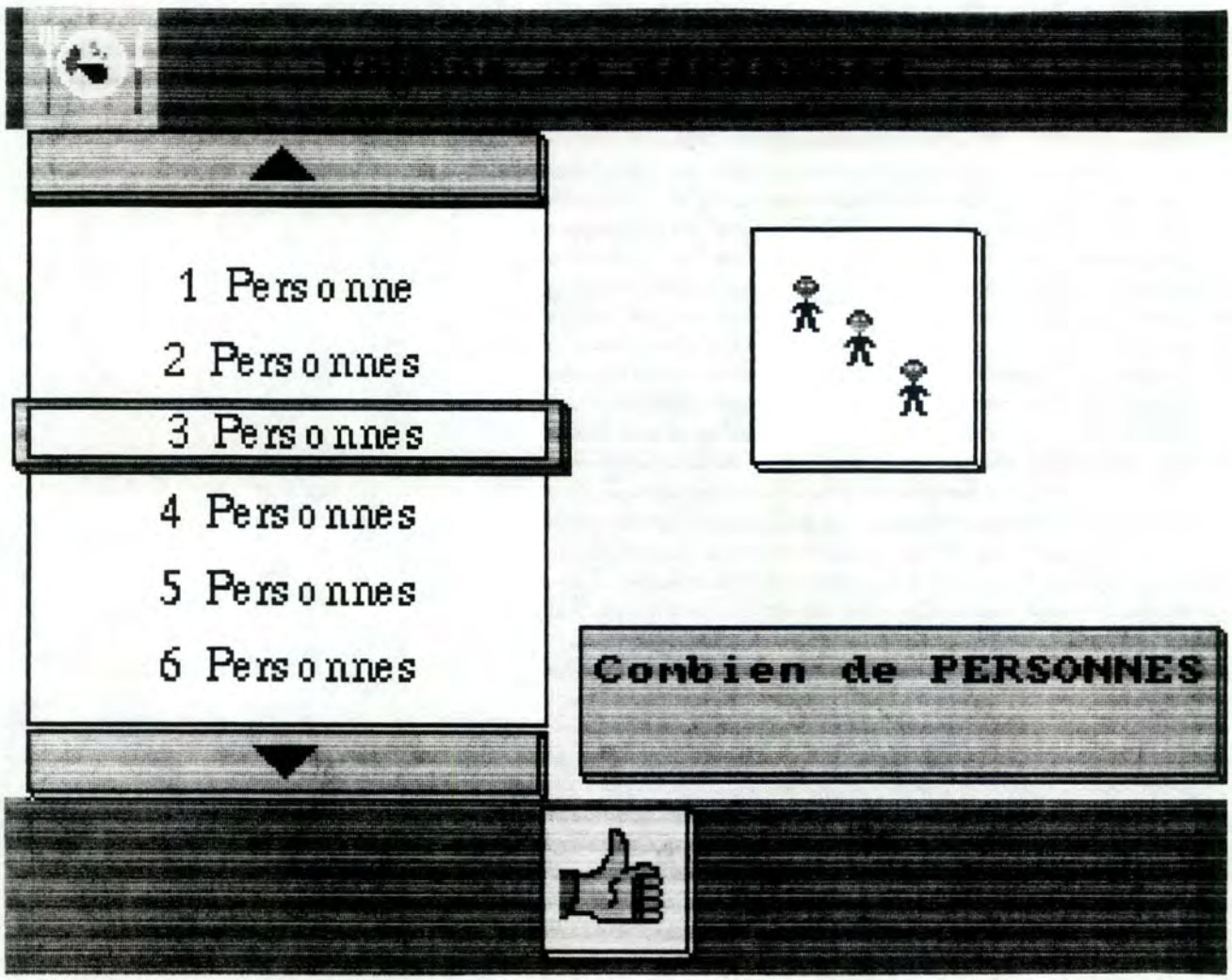
Remarquez que le nom des plats est écrit sur un fond qui a la couleur du thème auquel le plat est associé dans le menu.

Cliquer sur le bouton  permet de retirer du menu le plat courant du dérouleur. Si, après retrait d'un plat, votre menu n'en contient plus aucun, vous recevez un message (fenêtre 10).

Cliquer sur le bouton  vous permet de quitter cet écran.

II.4.2.11 ECRAN 10.

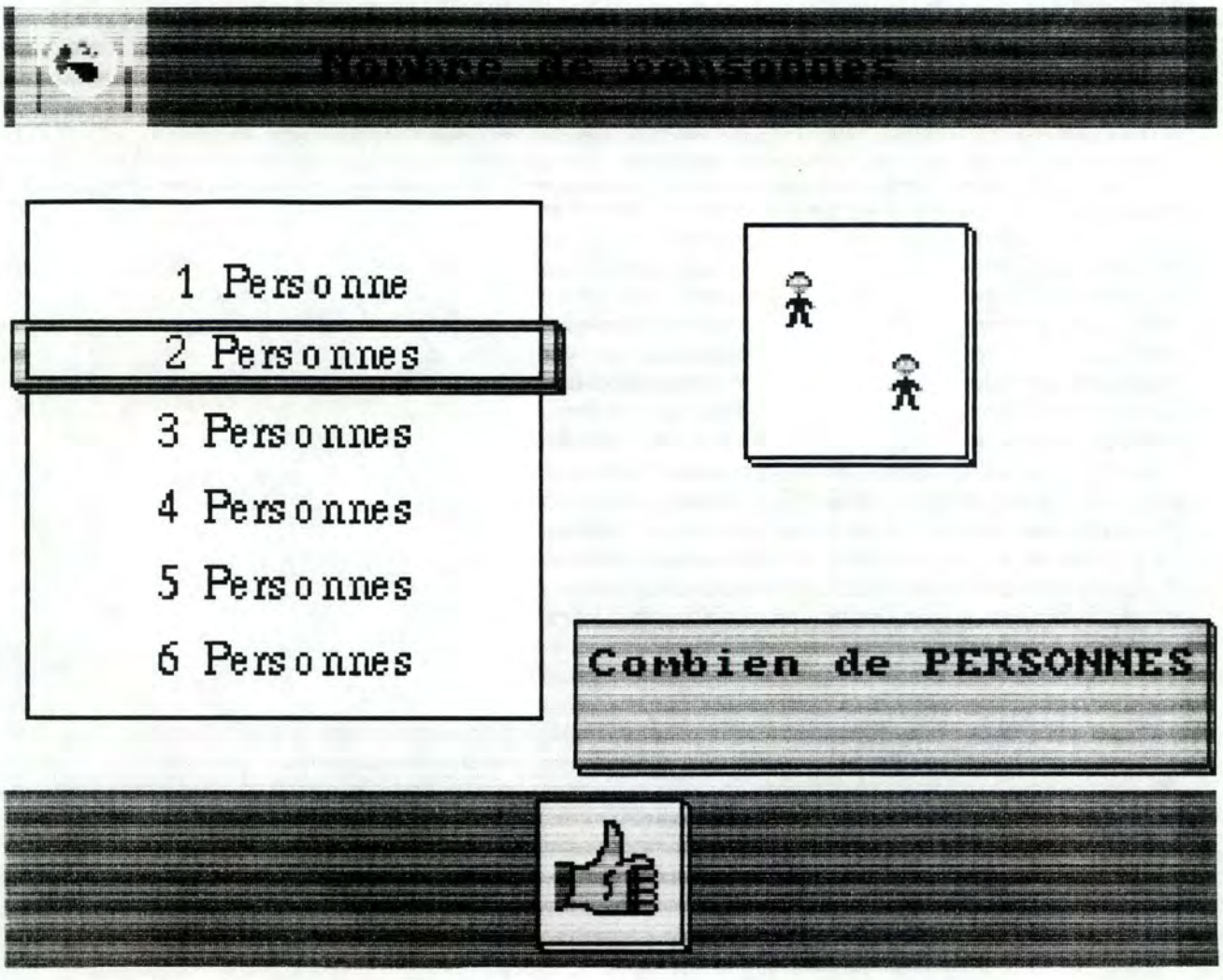
Lorsque l'écran 10 s'affiche, le nombre de personnes actuel du menu courant est dans la fenêtre du dérouleur. Vous pouvez modifier ce nombre si vous le souhaitez.



Une fois votre sélection faite, vous passez à l'écran suivant.

II.4.2.12 ECRAN 11.

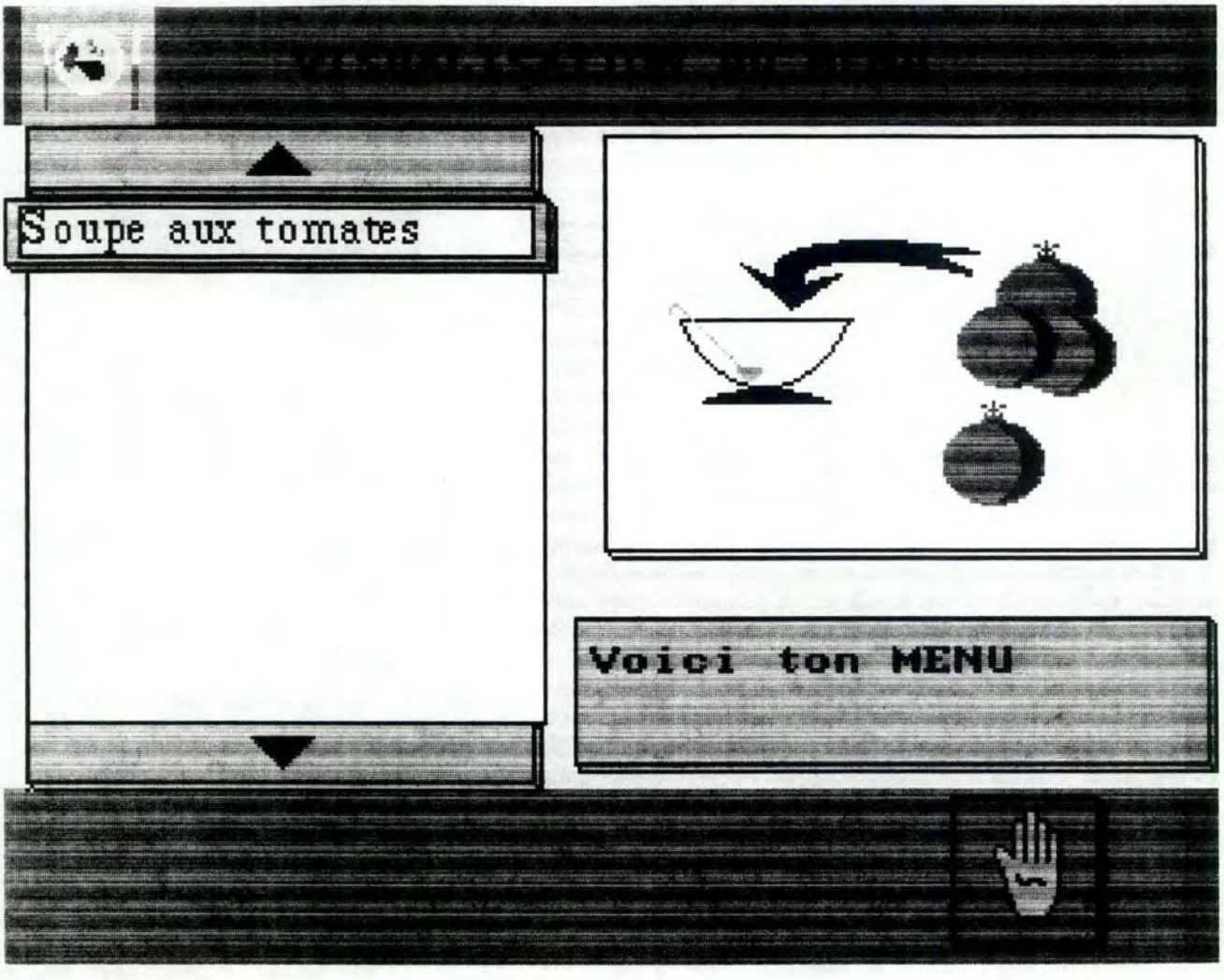
Lorsque l'écran 11 s'affiche, le nombre de personnes actuel du menu courant est dans la fenêtre du dérouleur. Vous ne pouvez pas modifier ce nombre.



Cliquer sur  vous permet de passer à l'écran suivant.

II.4.2.13 ECRAN 12.

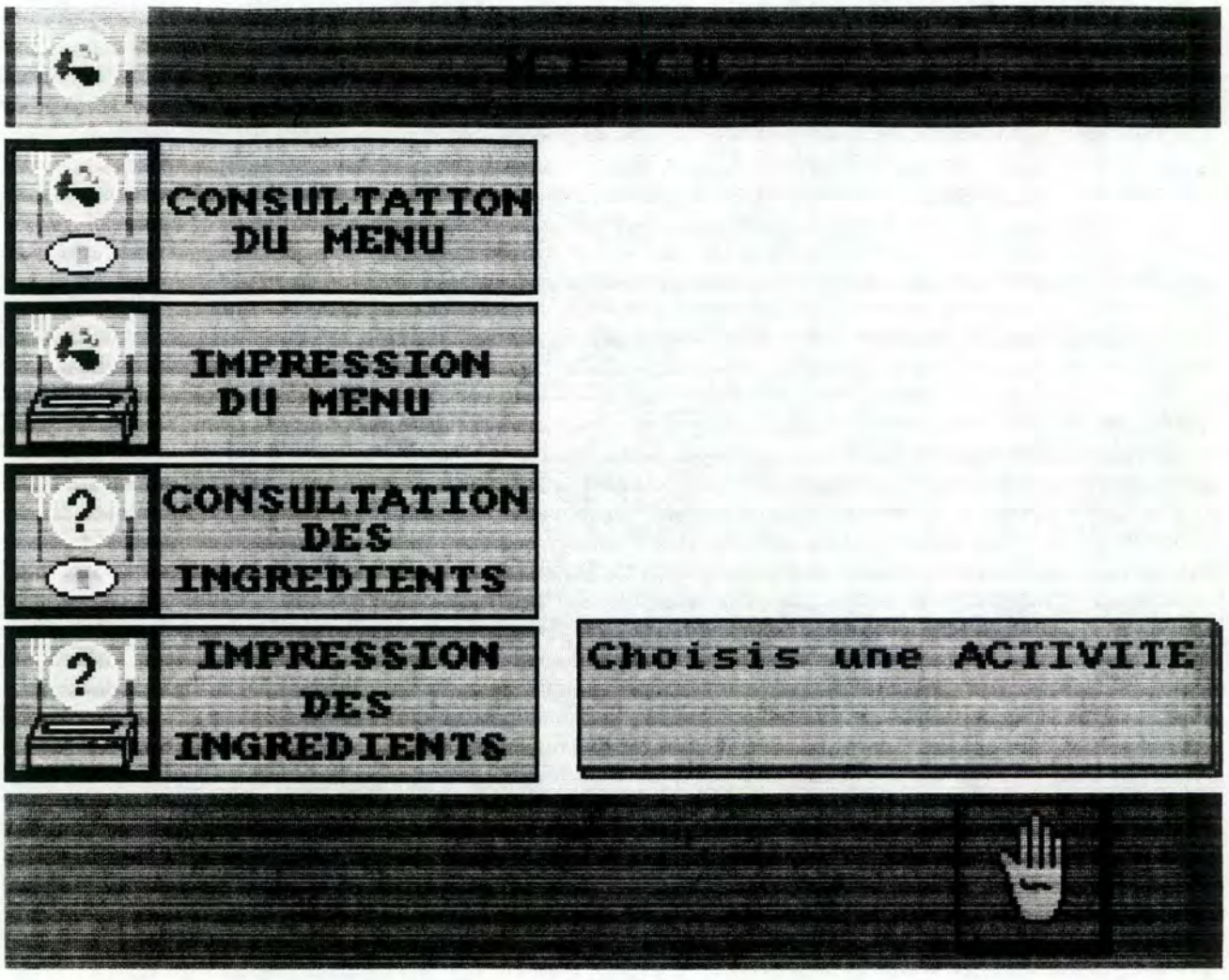
Sur l'écran 12, vous pouvez visualiser les thèmes et plats du menu courant. Vous ne pouvez rien modifier au contenu du menu.



Cliquer sur  vous permet de passer à l'écran suivant.

II.4.2.14 ECRAN 13.

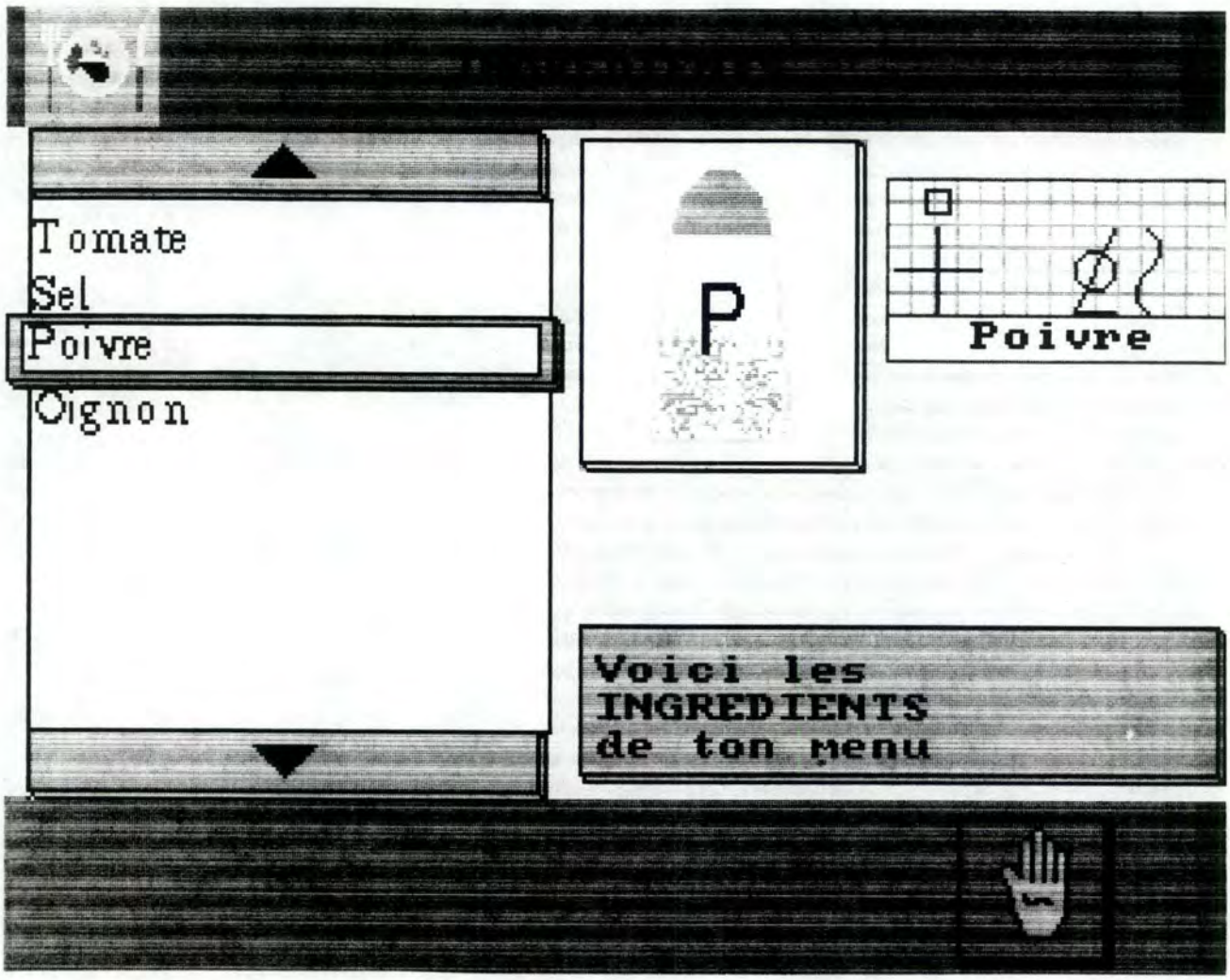
Sur l'écran 13, vous pouvez choisir entre quatre fonctionnalités.



Si vous ne souhaitez choisir aucune de ces fonctionnalités, vous quittez cet écran en cliquant sur  .

II.4.2.15 ECRAN 14.

Sur l'écran 14, vous pouvez visualiser la liste des produits entrant dans la composition des plats du menu courant.

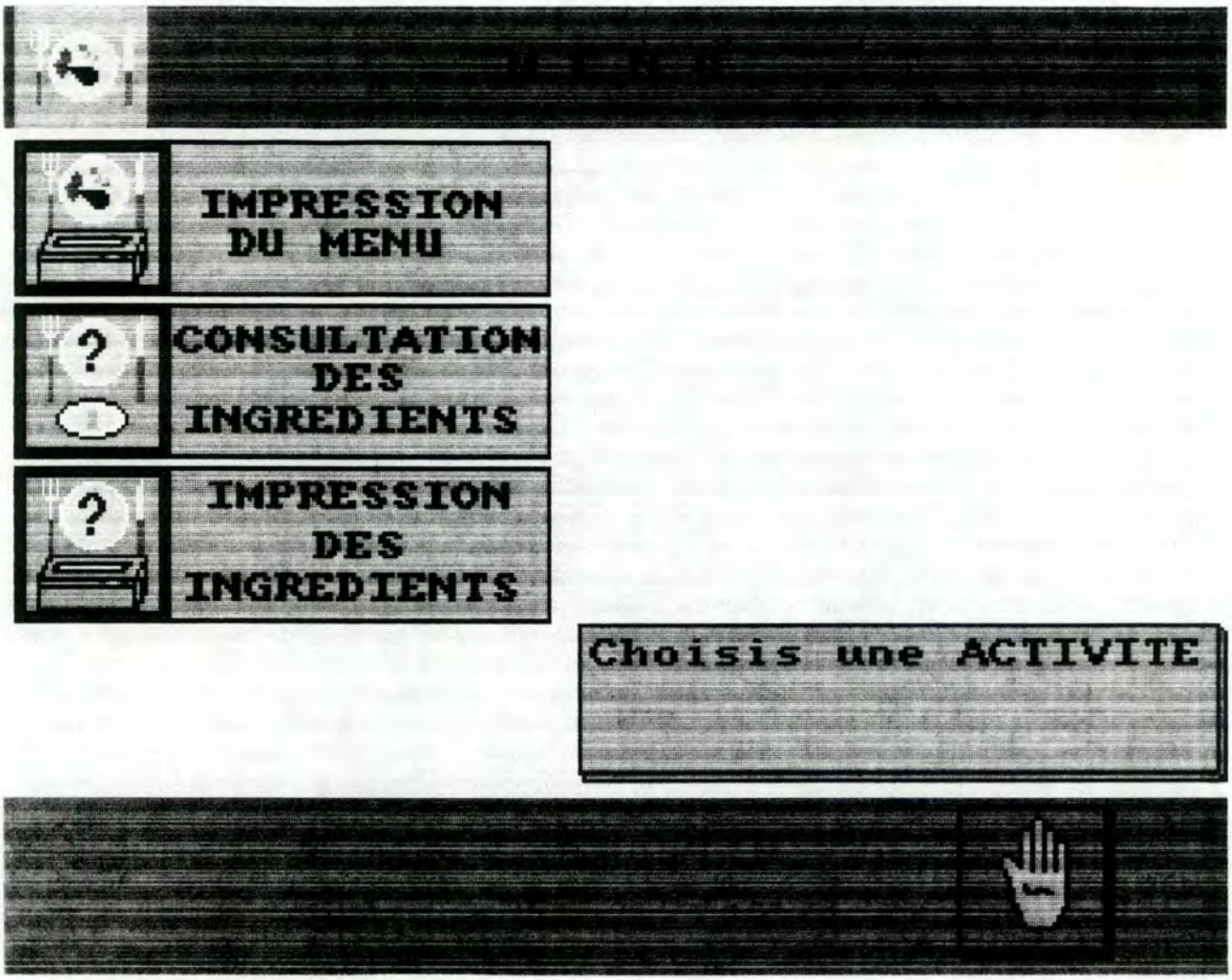


Les produits ont trois représentations : le texte (nom du produit), le dessin et le code Bliss associés au produit.

Vous quittez cet écran en cliquant sur  .

II.4.2.16 ECRAN 15.

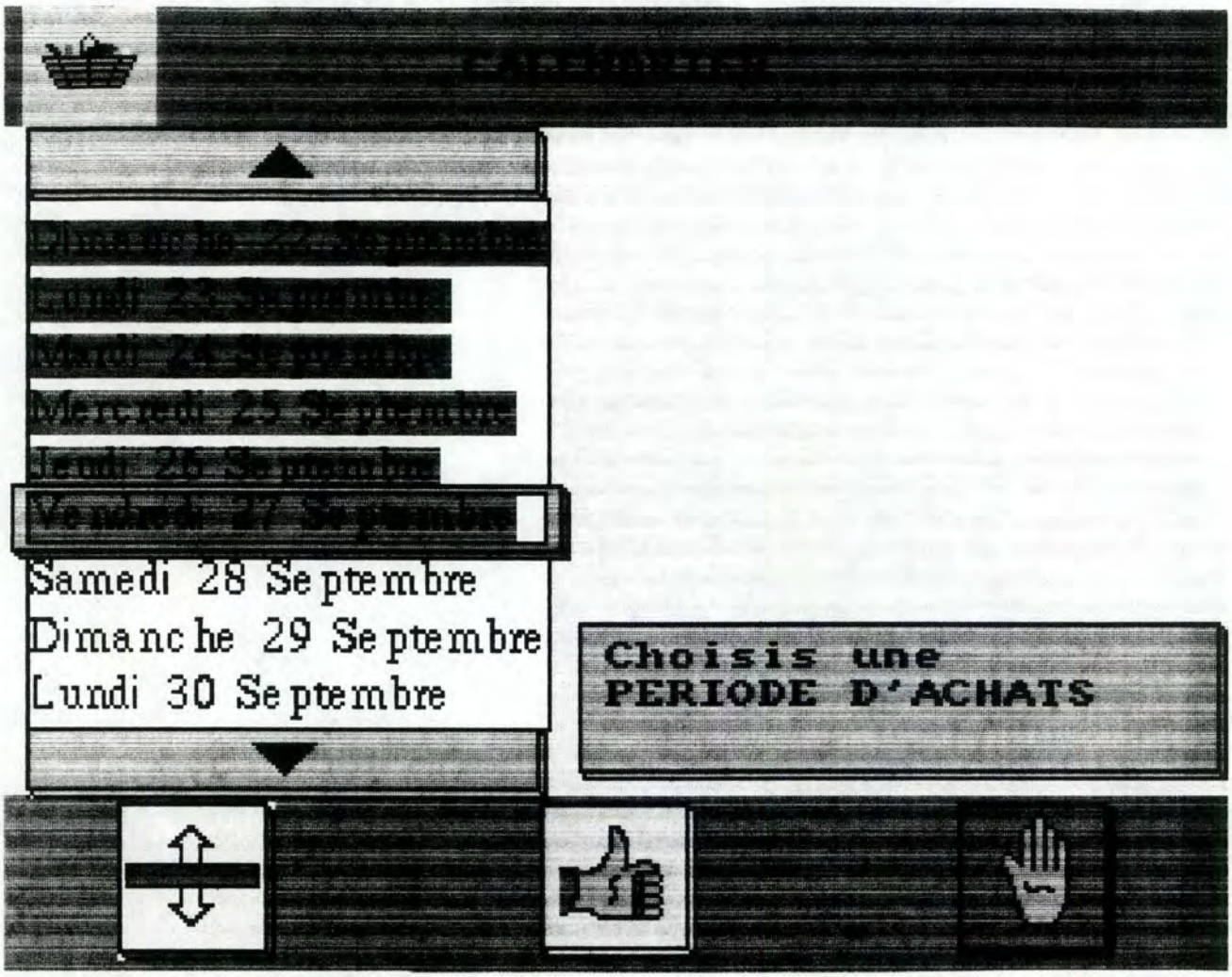
Sur l'écran 15, vous pouvez choisir entre trois fonctionnalités.



Si vous ne souhaitez choisir aucune de ces fonctionnalités, vous quittez cet écran en cliquant sur  .

II.4.2.17 ECRAN 16.

Sur cet écran, vous choisissez une période d'achat, une période à couvrir par la liste d'achats dont vous demandez l'établissement.



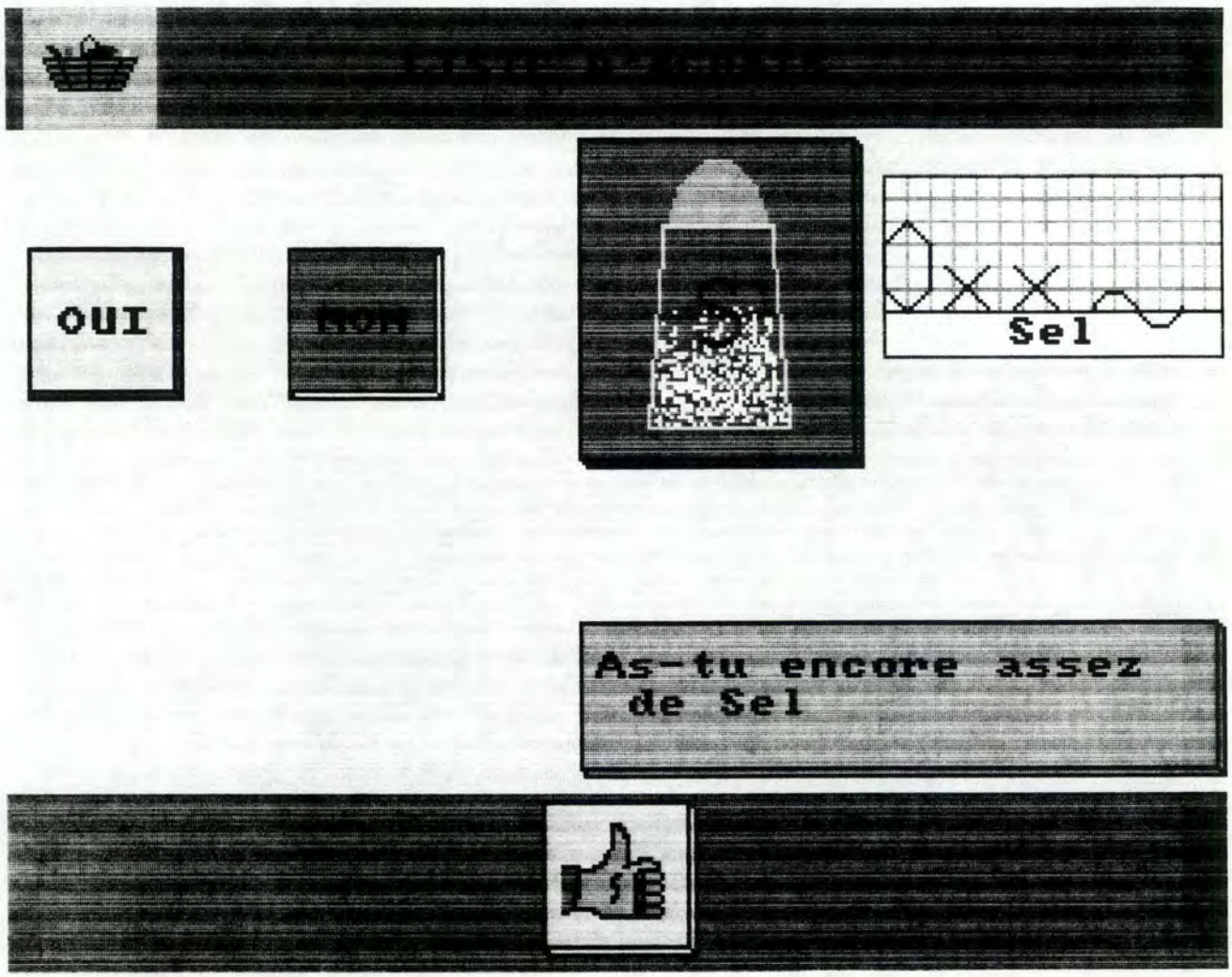
Vous spécifiez une date délimitant la période (date de début ou de fin, à votre choix) en amenant celle-ci dans la fenêtre du dérouleur et en cliquant sur  . Ensuite, vous spécifiez l'autre date délimitant la période en l'amenant celle-ci dans la fenêtre et en cliquant sur  , ce qui a pour effet de confirmer le choix d'une période. Cette dernière comprendra uniquement des jours traités.

Tant que vous n'avez pas confirmé ce choix, il vous est possible de changer la première date que vous avez introduite.

Dès que la période est spécifiée, vous passez automatiquement à l'écran suivant.

II.4.2.18 ECRAN 17.

Sur cet écran, différents produits vous sont proposés (un à la fois), et pour chacun d'entre eux, vous devez spécifier s'il vous en reste assez, au moyen des boutons OUI et NON .



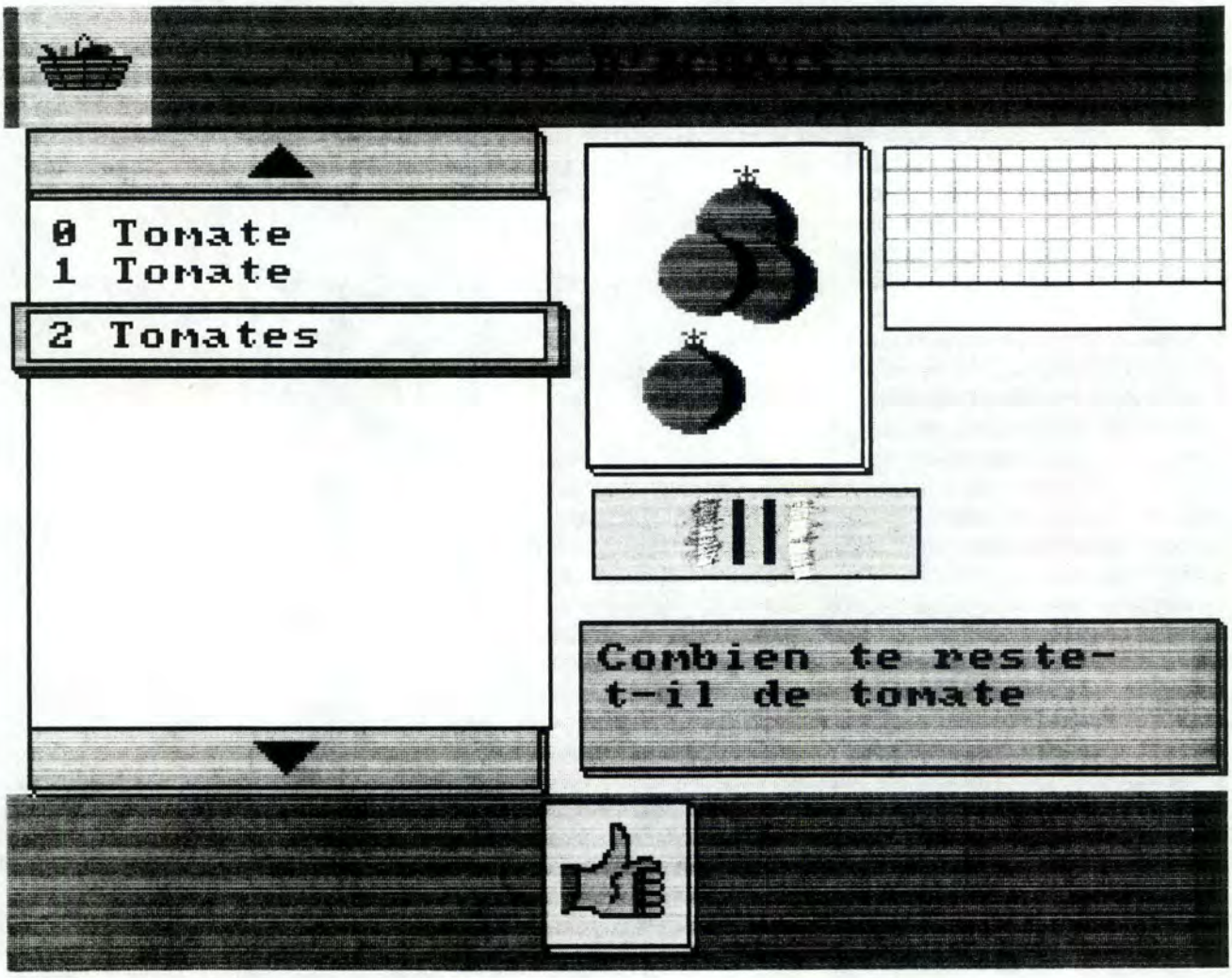
Vous confirmez votre choix d'une réponse (et passez automatiquement au produit suivant) au moyen du bouton .

Tous les produits qui vous sont ainsi proposés sont des produits nécessaires pour la période d'achat courante et dont la gestion de stock est réalisée par un système de ligne rouge.

Lorsque tous les produits ont été passé en revue, vous passez à l'écran suivant.

II.4.2.19 ECRAN 18.

Sur cet écran, différents produits vous sont proposés (un à la fois), et pour chacun d'entre eux, vous devez spécifier le nombre d'unités de conditionnement d'utilisation du produit qu'il vous reste, au moyen du dérouleur.



Lorsqu'un produit vous est présenté, la quantité nécessaire de celui-ci, exprimée en termes de conditionnement d'utilisation, est spécifiée dans la fenêtre du dérouleur (que vous faites apparaître, rappelons-le, en cliquant sur l'une des flèches du dérouleur). S'il vous reste moins de produit que la quantité nécessaire, vous le spécifiez au moyen du dérouleur. S'il vous en reste au moins autant que nécessaire, vous ne devez pas manipuler le dérouleur.

Vous confirmez votre choix d'une quantité (et passez au produit suivant) en cliquant sur  .

Tous les produits qui vous sont ainsi proposés sont des produits nécessaires pour la période d'achat courante et dont la gestion de stock n'est pas réalisée par un système de ligne rouge.

Lorsque tous les produits ont été passé en revue, vous passez à l'écran suivant.

II.4.3 LES FENETRES.

II.4.3.1 FENETRE 1.

Signale que le nom que vous avez introduit n'est pas celui de la disquette "BD".

II.4.3.2 FENETRE 2.

Confirmez ou infirmez que vous désirez réellement quitter le programme.

II.4.3.3 FENETRE 3.

Confirmez ou infirmez que vous désirez réellement quitter l'écran.

II.4.3.4 FENETRE 4.

Confirmez ou infirmez que vous désirez réellement quitter le travail sur les menus.

II.4.3.5 FENETRE 5.

Confirmez ou infirmez que vous désirez réellement créer un menu sans constituant.

II.4.3.6 FENETRE 6.

Le menu que vous désirez visualiser ne contient aucun plat.

II.4.3.7 FENETRE 7.

Le menu courant contient tous les thèmes possibles, vous ne pouvez plus en choisir d'autre.

II.4.3.8 FENETRE 8.

Veuillez préparer l'imprimante.

II.4.3.9 FENETRE 9.

Impression en cours, attendez la fin de ce travail, avant de poursuivre.

II.4.3.10 FENETRE 10.

Vous avez retirer tous les plats de votre menu, il n'en contient plus aucun.

II.4.3.11 FENETRE 11.

Veuillez introduire la disquette demandée dans un lecteur.

II.4.3.12 FENETRE 12.

Il y a un problème avec l'imprimante, le programme ne parvient pas à imprimer, vérifiez la.

II.4.3.13 FENETRE 13.

Le nom que vous avez introduit est-il correct ?

Annexe

Ci-dessous, vous trouverez les thèmes, les plats et leur composition, ainsi que les produits, qui vous sont proposés par le logiciel.

-1 : LES THEMES.

Thèmes
Entrée
Plat principal
Viande
Légume
Accompagnement
Dessert

Le thème Plat principal se décompose en : - Viande;
- Légume;
- Accompagnement;

et en fonction de la paramétrisation, vous avez accès soit au thème père, soit aux thèmes fils.

-2 : LES PLATS.

Thèmes	Plats en fonction des thèmes
Entrée	Soupe aux tomates
	Soupe aux poireaux
	Avocat crevettes
Plat principal	Poulet - Frites - Salade
	Steak - Pommes de terre - Petits pois et carottes
	Boulette - Frites
	Steak - Purée - Chou-fleur
	Côtelette - Pommes de terre - Salade de tomates
	Saucisse - Pommes de terre - Choux de Bruxelles
	Saucisse - Purée - Petits pois et carottes
	Boulette - Pommes de terre
	Steak - Pommes de terre - Haricots
	Poulet - Frites - Salade de tomates
Viande	Poulet
	Steak
	Saucisse
	Côtelette
	Boulette
Légume	Salade
	Petits pois et carottes
	Haricots
	Chicons
	Choux de Bruxelles
	Chou-fleur
	Salade de tomates
Accompagnement	Frites
	Pommes de terre
	Purée
Dessert	Glace
	Pomme

-3 : LES PRODUITS.

Produits	Prix	St.	Cond. utilisation	Cond. vente	Qt.
Ail	55	1	Gousse	Paquet	5
Avocat	30	0	Avocat	Avocat	0
Beurre	55	0	Paquet	Paquet	1
Chicon	9	0	Chicon	Chicon	0
Chou-fleur	50	0	Chou-fleur	Chou-fleur	0
Choux de Bruxelles	70	0	Choux de Bruxelles	Sac	20
Côtelette	40	0	Côtelette	Côtelette	0
Crevettes	100	0	Boîte	Boîte	1
Frites	85	0	Paquet	Paquet	1
Glace	70	0	Glace	Boîte	6
Haricots	35	0	Boîte	Boîte	1
Huile	30	1	Huile	Bouteille	0
Mayonnaise	65	0	Pot	Pot	1
Oeuf	3	0	Oeuf	Oeuf	0
Oignon	5	0	Oignon	Oignon	0
Persil	10	1	Persil	Persil	0
Petits pois et carottes	23	0	Boîte	Boîte	1
Poireau	50	0	Poireau	Botte	5
Poivre	20	1	Poivre	Pot	0
Pomme	6	0	Pomme	Pomme	0
Pomme de terre	150	0	Pomme de terre	Sac	50
Poulet	300	0	Poulet	Poulet	0
Purée	85	0	Sachet	Boîte	4
Salade	15	0	Salade	Salade	0
Saucisse	25	0	Saucisse	Saucisse	0
Sel	20	1	Sel	Paquet	0
Steak	50	0	Steak	Steak	0
Tomate	5	0	Tomate	Tomate	0
Viande hâchée	80	0	Barquette	Barquette	1
Vinaigre	25	1	Vinaigre	Bouteille	0

La seconde colonne indique le prix par unité de conditionnement de vente.
La troisième colonne indique le type de gestion de stock utilisée pour le produit : 1 = gestion type ligne rouge;
0 = autre gestion.
La quatrième et la cinquième colonne indiquent, respectivement, le conditionnement d'utilisation et le conditionnement de vente du produit.
La dernière colonne indique le nombre d'unités de conditionnement d'utilisation dans un conditionnement de vente.

-4 : LA COMPOSITION DES PLATS.

Plat	Produit	Qté.
Soupe aux tomates	Tomate	2
	Sel	0
	Poivre	0
	Oignon	1
Soupe aux poireaux	Poireaux	3
	Sel	0
	Poivre	0
Avocat crevettes	Avocat	1
	Crevettes	1
	Mayonnaise	0.1
Poulet	Poulet	0.5
	Beurre	0.25
	Sel	0
	Poivre	0
Steak	Steak	1
	Beurre	0.25
	Sel	0
	Poivre	0
Saucisse	Saucisse	1
	Sel	0
	Poivre	0
	Beurre	0.10
Côtelette	Côtelette	1
	Sel	0
	Poivre	0
	Beurre	0.10
Boulette	Viande hâchée	1
	Oeuf	1
	Sel	0
	Poivre	0
Salade	Salade	0.25
Petits pois et carottes	Petits pois et carottes	1
	Sel	0
	Persil	0
Haricots	Haricots	1
	Sel	0
	Poivre	0
	Persil	0
	Ail	0

Chicons	Chicons	3
	Sel	0
	Poivre	0
	Beurre	0.10
Choux de Bruxelles	Choux de Bruxelles	10
	Sel	0
Chou-fleur	Chou-fleur	1
	Sel	0
	Poivre	0
	Beurre	0.10
Salade de tomates	Salade	0.50
	Tomates	2
	Ail	0
	Vinaigre	0
	Huile	0
	Sel	0
	Poivre	0
Frites	Frites	0.25
Pommes de terre	Pommes de terre	6
	Sel	0
	Persil	0
Purée	Purée	1
	Sel	0
	Poivre	0
	Beurre	0.10
Glace	Glace	1
Pomme	Pomme	1

La dernière colonne indique la quantité de conditionnement d'utilisation du produit nécessaire à la préparation du plat pour une personne. Si cette quantité vaut 0, cela signifie qu'il faut "un peu" du produit (c'est le cas des produits pour lesquels la gestion de stock est du type ligne rouge).

Année académique 1990-1991

L'autonomie de la personne ayant une déficience mentale : un logiciel d'aide à la gestion des achats alimentaires.

ANNEXES.

Léopold Topet **Xavier Vonèche**

Promoteur : Madame le professeur Monique Noirhomme-Fraiture.
(Institut d'Informatique)
Co-promoteur : Monsieur le professeur Michel Mercier.
(Département de psychologie de la Faculté de Médecine)

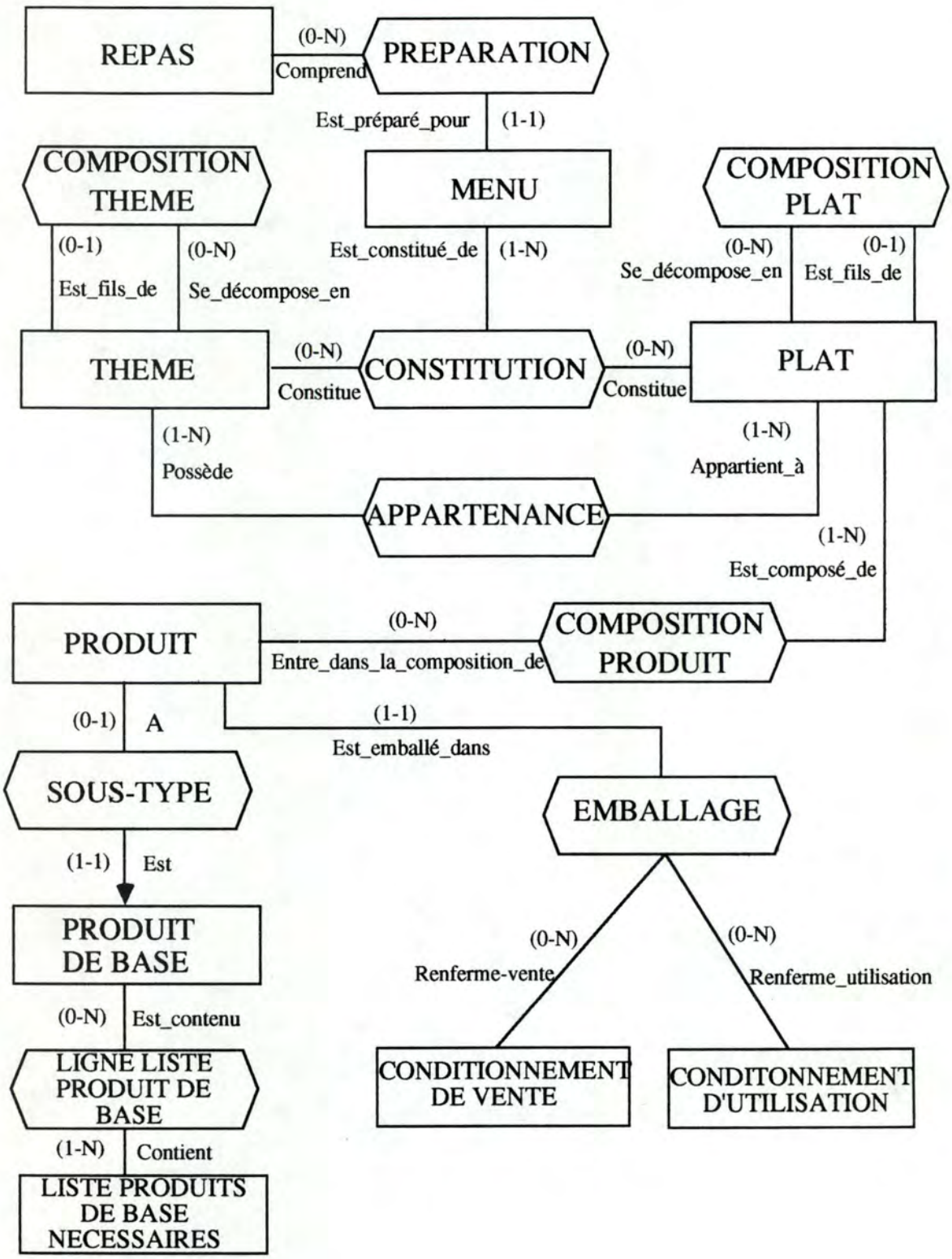
Mémoire présenté en vue de l'obtention du titre de Licencié et Maître en Informatique.

Table des matières.

Analyse fonctionnelle.

1 LA STRUCTURATION DES INFORMATIONS.

1.1 SCHEMA ENTITE - ASSOCIATION.



1.2 DEFINITION DES ELEMENTS DU SCHEMA E.A.

TYPE D'ENTITE : **REPAS.**

DEFINITION :
Un repas est un moment de la journée pour lequel on peut établir un menu.
IDENTIFIANTS :
No_repas : le numéro associé au repas.
Nom_repas : le nom du repas.
Priorité_repas : le numéro de l'ordre d'apparition des repas dans la journée.
ATTRIBUTS :
Article_repas : l'article, défini, du repas.

TYPE D'ENTITE : **MENU.**

DEFINITION :
Un menu est un ensemble de plats prévu pour un repas d'un jour, chaque plat étant associé à un thème dans le menu (et un thème ne peut être associé qu'a un seul plat dans le menu).
IDENTIFIANT :
(No_repas, Date_préparation) avec :
No_repas : le numéro du repas pour lequel est préparé le menu.
Date_préparation : la date du jour pour lequel est préparé le menu.
ATTRIBUTS :
Nombre_personnes_menu : le nombre de personnes prévue pour le menu au moment de l'impression de la dernière liste d'achats, ou à la création du menu si ce dernier a été créé après l'impression de la dernière liste d'achats.
NNombre_personnes_menu : le nombre de personnes prévues actuellement pour le menu
Modifie_menu : signale si le menu a été modifié depuis la dernière impression de listes d'achats.
Imprime_menu : signale si le menu a déjà été imprimé.
REMARQUES :
Nom_repas est un attribut de l'entité Repas.
Date_préparation est un attribut de l'association Préparation.

TYPE D'ENTITE : **THEME.**

DEFINITION :
Un thème est une rubrique potentielle d'un menu et pouvant ou non être décomposable en d'autres thèmes; à chaque thème appartient un ensemble de plats.
IDENTIFIANTS :
No_thème : la numéro associé au thème.
Nom_thème : le nom du thème.
Priorité_thème : le numéro de l'ordre d'apparition du thème dans un menu.
Couleur_thème : la couleur associée au thème.
ATTRIBUT :
Article_thème : l'article, défini, du thème.

TYPE D'ENTITE : **PLAT.**

DEFINITION :
Un plat est un constituant potentiel d'un menu, se composant de produits, appartenant à un ou plusieurs thèmes et pouvant ou non être décomposés en d'autres plats.
IDENTIFIANTS :
No_plat : le numéro associé au plat.
Nom_plat : le nom du plat.
Image_plat : le dessin du plat.
ATTRIBUTS :
Coût_plat : l'estimation de la cherté du plat.
Article_plat : l'article, indéfini, du plat.

TYPE D'ENTITE : **PRODUIT.**

DEFINITION :
Un produit est un aliment qui peut ou non entrer dans la composition de plats et que l'on peut acheter dans un magasin.
IDENTIFIANTS :
No_produit : le numéro associé au produit.
Nom_produit : le nom du produit.
Image_produit : le dessin du produit.
Bliss_produit : la représentation du produit en code Bliss.
ATTRIBUTS :
Article_produit : l'article, partitif, du produit.
Article2_produit : l'article faisant le lien entre les conditionnements (de vente ou d'utilisation) du produit et le produit.
Classe_stock_produit : définit la manière dont on gère le stock du produit (système dit de ligne rouge ou non).

Prix_produit : prix approximatif du produit par unité d'achat, exprimé dans l'unité monétaire du pays.

TYPE D'ENTITE : **PRODUIT_DE_BASE.**

DEFINITION :
Un produit de base est soit un produit qu'il faut toujours avoir chez soi, qu'il entre ou non dans la composition de plats, soit un produit qui n'entre dans la composition d'aucun plat et que l'on peut acheter de temps à autre sans pour autant en avoir toujours sous la main.

ATTRIBUTS :
Article_produit : l'article, partitif, du produit.
Type_produit_de_base : détermine si un produit de base est obligatoire ou facultatif.
Consommation_hebdomadaire_produit_de_base : donne la consommation hebdomadaire du produit de base, sans tenir compte de l'éventuel utilisation du produit dans un plat, exprimé termes de conditionnement d'utilisation (non nécessairement des quantités entières).
Classe_produit_de_base : définit une appartenance à un ensemble de produits de base .

REMARQUES :
Produit de base est un sous-type du type d'entité Produit; par conséquent, il en hérite toutes les propriétés et tous les attributs.

TYPE D'ENTITE : **CONDITIONNEMENT_DE_VENTE.**

DEFINITION :
Un conditionnement_de_vente est un conditionnement dans lequel un produit est vendu.

IDENTIFIANTS :
No_conditionnement_de_vente : le numéro associé au conditionnement.
Nom_conditionnement_de_vente : nom du conditionnement.

TYPE D'ENTITE : **CONDITIONNEMENT_D'UTILISATION.**

DEFINITION :
Un conditionnement_d'utilisation est la plus petite unité d'emballage utilisable du produit[1].
IDENTIFIANT :
No_conditionnement_d'utilisation : le numéro associé au conditionnement.
Nom_conditionnement_d'utilisation : le nom du conditionnement d'utilisation.

TYPE D'ENTITE : **LISTE_PRODUITS_DE_BASE_NECESSAIRES.**

DEFINITION :
Une liste_produits_de_base_nécessaires est une liste reprenant les produits de base obligatoires ou facultatifs demandés pour une période.
IDENTIFIANT :
(Date_début_période_liste, Date_fin_période_liste) avec :
Date_début_période_liste : date du début de la période concernée par cette liste.
Date_fin_période_liste : date de la fin de la période concernée par cette liste.

TYPE D'ASSOCIATION : **PREPARATION.**

DEFINITION :
Exprime qu'un menu est créé pour un repas.
IDENTIFIANT :
(Date_préparation, No_repas).

TYPE D'ASSOCIATION : **CONSTITUTION.**

DEFINITION :
Exprime qu'un plat est un constituant d'un menu pour un thème.
IDENTIFIANT :
(Date_préparation, No_repas, No_thème).
ATTRIBUT :
Modifie_plat_constitution : dit si oui ou non un plat a été ajouté à un menu pour un thème , supprimé d'un menu pour un thème, n'a ni été ajouté à un menu pour un thème ni été supprimé d'un menu pour un thème depuis la dernière impression de liste d'achats.

---

[1] Le conditionnement d'utilisation est à prendre au sens large. Par exemple, celui d'une pomme est la pomme elle-même.

TYPE D'ASSOCIATION : **APPARTENANCE.**

DEFINITION :
Exprime l'appartenance d'un plat à un thème.
IDENTIFIANT :
(No_thème, No_plat).

TYPE D'ASSOCIATION : **COMPOSITION_THEME.**

DEFINITION :
Exprime que certains thèmes sont décomposables en thèmes élémentaires.
IDENTIFIANT :
(No_thème, No_thème).

TYPE D'ASSOCIATION : **SOUS_TYPE.**

DEFINITION :
Exprime qu'un produit de base est un produit.

TYPE D'ASSOCIATION : **COMPOSITION_PLAT.**

DEFINITION :
Exprime que certains plats sont décomposables en plats élémentaires.
IDENTIFIANT :
(No_plat, No_plat).

TYPE D'ASSOCIATION : **COMPOSITION_PRODUIT.**

DEFINITION :
Exprime le fait qu'un produit entre dans la composition d'un plat.
IDENTIFIANT :
(No_plat, No_produit).
ATTRIBUT :
Quantité_composition_produit : quantité du produit utilisée pour la préparation du plat pour une personne, exprimée dans l'unité de mesure Mesure_produit..

TYPE D'ASSOCIATION : **LIGNE_LISTE_PRODUITS_DE_BASE.**

DEFINITION :
Exprime qu'une liste de produits de base se compose de produits_de_base.
IDENTIFIANT :
(Date_début_période_liste, Date_fin_période_liste, No_produit).

TYPE D'ASSOCIATION : **EMBALLAGE.**

DEFINITION :
Exprime qu'une unité de produit est emballée (s'achète) dans un certain emballage (unitaire) et que plusieurs unités d'emballage d'utilisation de ce produit sont emballés (s'achètent) dans un emballage de vente.
IDENTIFIANT :
(No_produit, No_conditionnement_de_vente, No_conditionnement _d'utilisation).
ATTRIBUTS :
Quantité_conditionnement_d'utilisation_dans_conditionnement_de_-vente : donne la quantité d'unités de conditionnement_d'utilisation dans une unité de conditionnement_de_vente.

1.3 CONTRAINTES.

-1 : tous les attributs du schéma sont obligatoires sauf Classe_produit_de-_base qui est facultatif (voir contrainte 6 ci-dessus), en outre, ils sont tous élémentaires et simples.
-2 : toutes les occurrences de toutes les entités et de toutes les associations ont une durée de vie indéterminée, c'est-à-dire que c'est l'utilisateur lui-même qui les supprime quand il le désire, à l'exception des occurrences des entités Menu et Liste_produits_de_base_nécessaires et des occurrences des associations Préparation, Constitution et Ligne_liste_produits_de_base : toute occurrence de Menu est supprimée dès que la date du Menu est antérieure à la date du jour courant, de même, l'occurrence de Préparation et les occurrences de Constitution auxquelles participe ce Menu sont supprimées; toute occurrence de Liste_produits_de_base_nécessaires est supprimée dès que la date de fin de la liste est antérieure à la date du jour courant, et les occurrences de Ligne_liste_produits_de_base auxquelles participe cette Liste sont également supprimées.
-3 : si un thème se décompose, les thèmes obtenus par décomposition ne peuvent pas se décomposer.
-4 : si un plat se décompose, les plats obtenus par décomposition ne peuvent pas se décomposer.

-5 : un thème et un plat ne se décomposent pas en eux-mêmes.
-6 : si un thème se décompose en x thèmes, quel que soit x, tout plat possible pour ce thème se décompose aussi en x plats, éventuellement sans valeur significative, auquel cas leur numéro vaut l'infini.
-7 : un plat se décomposant ne peut appartenir qu'à un seul thème.
-8 : un même thème et un même plat ne peuvent participer ensemble qu'à une seul occurrence de Appartenance.
-9 : un plat obtenu par décomposition d'un autre plat ne peut appartenir qu'à un thème obtenu par décomposition d'un autre thème; de même, un thème obtenu par décomposition d'un thème ne peut participer qu'à des occurrences d'Appartenance auxquelles des plats obtenus par décomposition de plats participent.
-10 : la priorité des thèmes élémentaires obtenus par décomposition d'un autre thème est strictement supérieure à la priorité de ce dernier.
-11 : un même thème ne peut participer qu'à une et une seule occurrence de constitution à laquelle participe un même menu.
-12 : un plat et un thème ne peuvent participer à une même occurrence de Constitution que s'ils participent à une même occurrence de Appartenance.
-13 : les attributs Nombre_personnes_menu et NNombre_personnes_menu ont une valeur comprise entre 0 et 6 inclus.
-14 : pour tout plat, la valeur de l'attribut Coût_plat est CHER, NORMAL, BON MARCHE.
-15 : si un produit de base est obligatoire, alors l'attribut Classe_produit_-de_base a la valeur INEXISTANTE.
-16 : pour tout produit, la valeur de l'attribut Classe_stock_produit appartient à {APPROXIMATION, ACHAT UNITE}.
- si un produit est de la classe APPROXIMATION, cela signifie que la quantité restante de ce produit ne peut être qu'approximée (via un système de ligne rouge par exemple), c'est le cas des épices;
- si un produit est de la classe ACHAT UNITE, cela signifie que l'on peut compter les unités restantes de ce produit et qu'il s'agit d'un produit que l'on achète à l'unité, c'est le cas des boites de conserve.
-17 : pour tout menu, la valeur de l'attribut Modifie_menu appartient à {VRAI, FAUX}.
- si la valeur de cet attribut est VRAI, cela signifie que le menu a été modifié depuis la dernière impression de liste d'achats relative à au moins ce menu;
- cette valeur est FAUX sinon.
-18 : pour tout menu, la valeur de l'attribut Imprime_menu appartient à {VRAI, FAUX}.
- si la valeur de cet attribut est VRAI, cela signifie que le menu a déjà été imprimé;
- cette valeur est FAUX sinon.

-19 : pour toute Constitution, la valeur de l'attribut Modifie_plat_constitution appartient à {AJOUTE, ENLEVE, INCHANGE}.
- si la valeur de cet attribut est AJOUTE, cela signifie que le plat participant à cette occurrence de Constitution a été ajouté au menu qui participe à cette même occurrence après la dernière impression de liste d'achats relative à au moins ce menu;
- si la valeur de cet attribut est ENLEVE, cela signifie que le plat participant à cette occurrence de Constitution a été supprimé du menu qui participe à cette même occurrence après la dernière impression de liste d'achats relative à au moins ce menu;
- cette valeur est INCHANGE sinon.

-20 : pour tout produit de base, la valeur de l'attribut Classe_produit_de_base appartient à {TARTINES, FRUITS, DIVERS}.
- si la valeur de cet attribut est TARTINES, cela signifie que le produit de base est un produit que l'on utilise pour garnir les tartines;
- si la valeur de cet attribut est FRUITS, cela signifie que le produit de base est un fruit;
- cette valeur est DIVERS sinon.

-21 : pour tout produit de base, la valeur de l'attribut Type_produit_de_base appartient à {FACULTATIF, OBLIGATOIRE}.

2 LA STATIQUE DES TRAITEMENTS.

2.1 DEFINITION DU PROJET ET DES APPLICATIONS.

1. projet JE FAIS MES COURSES.

Objectif : permettre à l'utilisateur de constituer des menus pour des repas à certaines dates, de modifier les menus constitués, de demander leur consultation et leur impression, de choisir un certain nombre de produits alimentaires qu'il consommera en dehors des menus, ainsi que de demander l'établissement et l'impression d'une liste d'achats qui reprend les produits alimentaires et les quantités nécessaires de ces produits pour la préparation des menus constitués, pour satisfaire à la liste des produits demandés hors menus et pour qu'il y ait toujours un stock minimum de certains produits dans les réserves de l'utilisateur. Le logiciel doit pouvoir être utilisé par une personne ayant une déficience mentale, et la forme de la liste d'achats doit être telle qu'une personne ayant une déficience mentale soit à même de pouvoir l'utiliser pour réaliser, de façon la plus autonome possible, les achats prescrits par la liste.

Message-donnée : /.

Propriétés : /.

Message-résultat : /.

2. application MENU.

Objectif : permettre à l'utilisateur de constituer des menus pour des repas à certaines dates et d'enregistrer les menus valides ainsi constitués, de modifier les menus constitués et d'enregistrer les modifications valides ainsi apportées, et de demander la consultation et l'impression de menus constitués. Il y a deux scénarios de création de menus. Dans le premier, l'utilisateur est obligé de constituer un menu (éventuellement vide) pour chaque repas de chaque jour, l'utilisateur est obligé de traiter chaque repas de chaque jour. Dans le second, il n'est plus obligé de traiter chaque repas, il peut constituer de zéro à un menu pour chaque jour.

Message-donnée : - Paramètres_utilisateur.

Propriété : - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser l'application en fonction de l'utilisateur.

Message-résultat : /.

3. application PRODUITS DE BASE.

Objectif : permettre à l'utilisateur d'établir une liste de produits de base facultatifs qu'il désire pour une période, la modification de cette liste, et établir la liste des produits de base nécessaires et des quantités nécessaires de ces produits pour cette période.

Message-donnée : - Paramètres_utilisateur.

Propriété : - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser l'application en fonction de l'utilisateur.

Message-résultat : /.

4. application LISTE D'ACHATS.

Objectif : établir la liste des produits à acheter et des quantités à acheter de ces produits pour une période et imprimer une liste d'achats. Il y a trois scénarios de création de listes d'achats, chacun étant associé à un scénario de création de menus. Au premier scénario de création de menus est associé l'établissement d'une liste d'achats qui reprend les produits à acheter et les quantités dans lesquels acheter ces produits nécessaires pour la préparation des constitués pour une certaine période, pour satisfaire à la liste des produits demandés hors menu et pour qu'il y ait toujours un stock minimum de certain produits dans les réserves de l'utilisateur. Au deuxième scénario de création de menus est associé le second type d'établissement d'une liste d'achats, avec lequel la liste obtenue reprend uniquement les produits à acheter et les quantités à acheter de ces produits nécessaires à la préparation de menus constitués pour une certaine période.

En outre, cette application doit permettre à l'utilisateur de pouvoir introduire le montant du budget dont il dispose pour effectuer ces achats pour une certaine période; ce montant, comparé au coût approximatif des achats à réaliser pour cette même période permettra de signaler à l'utilisateur s'il a ou non assez s'argent pour effectuer ses courses pour cette période et donc s'il est nécessaire ou pas qu'il modifie les menus constitués pour cette période ou la liste des produits de base établie pour cette période.

Message-donnée : - Paramètres_utilisateur.

Propriété : - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser l'application en fonction de l'utilisateur.

Message-résultat : /.

2.2. DEFINITION DES PHASES ET DES FONCTIONS.

APPLICATION MENU.

1. DEFINITION DES PHASES.

1.1. CREATION DE MENUS.

Objectif : permettre à l'utilisateur, en fonction du scénario de création de menus, de constituer un ou plusieurs menus à certaines dates, et enregistrer toutes les informations concernant ces menus si elles sont valides, c'est-à-dire pour chacun des menus, sa date, le repas qu'il concerne, le nombre de personnes pour lequel il est prévu, les plats le constituant, et les thèmes auxquels sont associés ces plats dans le menu. En outre, pour chaque menu créé, l'utilisateur peut demander son impression ainsi que l'affichage et l'impression de la liste des produits entrant dans la composition des plats du menu.

Messages-données : -Paramètres_utilisateurs.

Propriétés : - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser la phase en fonction de l'utilisateur.

Consultation MSI : Repas, Préparation, Thème, Plat, Appartenance, Composition_produit, Produit.

Messages-résultats : /.

Actions MSI : - ajout de m Menu;
- ajout de m Préparation;
- ajout de n Constitution;

où m est le nombre de repas connus du S.I. multiplié par le nombre de jours traités lors de cette phase, et n est la somme des nombres de plat constituant les menus constitués.

1.2. MODIFICATION DE MENUS.

Objectif : permettre à l'utilisateur de modifier des menus constitués (nombre de personnes et/ou des couples (thème, plat)) et d'enregistrer les modifications valides ainsi apportées. En outre, pour chaque menu modifié, l'utilisateur peut demander son impression ainsi que l'affichage et l'impression de la liste des produits entrant dans la composition des plats du menu.

Messages-données : - Paramètres_utilisateurs.

Propriétés: - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser la phase en fonction de l'utilisateur.

Consultation MSI : Repas, Préparation, Thème, Plat, Appartenance, Constitution, Composition_produit, Produit.

Messages-résultats : /.

Actions MSI : - mise-à-jour de la valeur de l'attribut Nombre_personnes_menu de m occurrences de Menu;
- mise-à-jour de la valeur de l'attribut Modifie_plat_constitution de n occurrences de Constitution;
- ajout de p occurrences de Constitution;
- mise-à-jour de la valeur de l'attribut Modifie_menu de m occurrences de Menu;

où m est le nombre de menus modifiés, n est le nombre de thèmes qui ont été retirés de menus ou dont le plat associé dans un menu modifié a changé, et p est le nombre de thèmes ajoutés à des menus.

1.3. CONSULTATION DE MENUS.

Objectif : permettre à l'utilisateur de demander l'affichage à l'écran du contenu de menus à choisir par lui-même. En outre, pour chaque menu consulté, l'utilisateur peut demander son impression ainsi que l'affichage et l'impression de la liste des produits entrant dans la composition des plats du menu.

Messages-données : - Paramètres_utilisateur.

Propriétés : - Paramètres_utilisateur est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser la phase en fonction de l'utilisateur.

Consultation MSI : Menu, Constitution, Composition_produit, Produit.

Messages-résultats : /.

Action MSI : /.

1.4 IMPRESSION DE MENUS.

Idem phase *Consultation d'un menu*, sauf qu'il s'agit de permettre à l'utilisateur de demander l'impression de menus.

1.5. SUPPRESSION DE MENUS.

Objectif : supprimer automatiquement tous les menus dont la date est antérieure à la date du jour.

Message-donnée : Date_courante_d.

Propriété : Date_courante_d est la date du jour courant.

Consultation MSI : Préparation.

Message-résultat : /.

Actions MSI : - suppression de n Préparation et de n Menu;
- suppression de m Constitution.

2. DEFINITION DES FONCTIONS.

2.1. phase CREATION DE MENUS.

2.1.1. ENREGISTREMENT D'UN MENU VALIDE.

Objectif : enregistrer un menu valide.

Messages-données : - Date_menu_d;
- Repas_menu_d;
- Nombre_personnes_menu_d;
- n couples (Plat_menu_d, Thème_plat_d).

Propriétés : - Date_menu_d, Repas_menu_d, Nombre_personnes_menu_d, les couples et leurs composantes, valides;
- le nombre de plats composant le menu est valide;
- les Messages-données sont les informations concernant un seul et même menu.

Consultation MSI : /.

Message-résultat : /.

Actions MSI : - ajout d'un Menu identifié par (Date_menu_d, Repas_menu_d);
- ajout d'une Préparation identifiée par (Date_menu_d, Repas_menu_d);
- ajout de n Constitution, chacune étant identifiée par (Date_menu_d, Repas_menu_d) et un couple (Plat_menu_d, Thème_plat_d), pour tout couple (Plat_menu_d,Thème_plat_d).

Règles : - la valeur de l'attribut Modifie_menu du Menu identifié par (Date_menu_d, Repas_menu_d) est "VRAI";
- la valeur de l'attribut Modifie_plat_constitution de l'occurrence de Constitution identifiée par (Date_menu_d, Repas_menu_d, Plat_menu_d, Thème_plat_d) est "AJOUTE", pour tout (Plat_menu_d, Thème_plat_d).

2.1.2. VERIFICATION REPAS EXISTANT.

Objectif : vérifier qu'un repas pour lequel on veut enregistrer un menu est un repas connu du S.I.

Message-donnée : Nom_repas_d.

Propriété : Nom_repas_d est syntaxiquement valide.

Consultation MSI : Repas.

Messages-résultats : Nom_repas_à_accepter_r
ou Nom_repas_à_refuser_r.

Action MSI : /.

Règle : si Nom_repas_d est le nom d'un repas connu du S.I.
alors Nom_repas_à_accepter_r
sinon Nom_repas_à_refuser_r.

2.1.3. VERIFICATION REPAS NON TRAITE.

Objectif : vérifier qu'un repas pour lequel, à une certaine date, on veut enregistrer un menu, est un repas non traité.

Messages-données : - Nom_repas_d;
- Date_repas_d.

Propriétés : - Nom_repas_d est le nom d'un repas connu du S.I.;
- Date_repas_d est syntaxiquement valide.

Consultation MSI : Préparation.

Messages-résultats : Repas_à_accepter_r
ou Repas_à_refuser_r.

Action MSI : /.

Règle : si le repas Nom_repas_d du jour Date_jour_d est non traité
alors Repas_à_accepter_r
sinon Repas_à_refuser_r.

2.1.4. VERIFICATION PREMIER REPAS NON TRAITE.

Objectif: vérifier qu'un repas non traité pour lequel, à une certaine date, on veut enregistrer un menu, est le premier repas non traité de cette date.

Messages-données : - Nom_repas_d;
- Date_repas_d.

Propriétés : - Nom_repas_d est le nom d'un repas non traité à la date Date_repas_d;
- Date_repas_d est syntaxiquement valide.

Consultation MSI : Préparation.

Messages-résultats : Repas_à_accepter_r
ou Repas_à_refuser_r.

Action MSI : /.

Règle : si le repas Nom_repas_d est le premier repas non traité du jour Date_repas_d
alors Repas_à_accepter_r
sinon Repas_à_refuser_r.

2.1.5. VERIFICATION JOUR NON ANTERIEUR AU JOUR COURANT.

Objectif : vérifier qu'une date pour laquelle on veut enregistrer un menu est la date d'un jour non antérieur au jour courant.

Messages-données : - Date_jour_courant_d;
- Date_d.

Propriétés : - Date_jour_courant_d et Date_d sont syntaxiquement valides;
- Date_jour_courant_d est la date du jour courant.

Consultation MSI : /.

Messages-résultats : Jour_à_accepter_r
ou Jour_à_refuser_r.

Action MSI : /.

Règle : si Date_d >= Date_jour_courant_d
alors Jour_à_accepter_r
sinon Jour_à_refuser_r.

2.1.6. VERIFICATION ELOIGNEMENT D'UNE DATE.

Objectif : vérifier que le nombre de jours entre une date pour laquelle on veut enregistrer un menu et la date du jour courant est inférieur à un certain nombre représentant l'éloignement maximal accepté entre ces deux dates.

Messages-données : - Date_jour_courant_d;
- Date_d;
- Eloignement_maximum_d.

Propriétés : - Date_jour_courant_d et Date_d sont syntaxiquement valides;
- Eloignement_maximum est un entier positif ou nul;
- Date_jour_courant_d est la date du jour courant;
- Date_d est la date d'un jour non antérieur au jour courant.

Consultation MSI : /.

Messages-résultats : Date_à_accepter_r
ou Date_à_refuser_r.

Action MSI : /.

Règle : si (Date_d - Date_jour_courant_d + 1) <= Eloignement_maximum_d
alors Date_à_accepter_r
sinon Date_à_refuser_r.

2.1.7. VERIFICATION PREMIER JOUR NON TRAITE.

Objectif : vérifier qu'une date pour laquelle on veut enregistrer un menu est la date du premier jour non traité ultérieur au jour courant ou du jour courant lui-même s'il est non traité.

Messages-données : - Date_jour_courant_d;
- Date_d.

Propriétés : - Date_jour_courant_d et Date_d sont syntaxiquement valides;
- Date_jour_courant_d est la date du jour courant;
- Date_d est la date d'un jour non traité;
- Date_d >= Date_jour_courant_d.

Consultation MSI : Préparation.

Messages-résultats : Jour_à_accepter_r
ou Jour_à_refuser_r.

Règle : si Date_d = Date_jour_courant_d
alors Jour_à_accepter_r
sinon si (tous les jours compris entre Date_jour_courant_d et Date_d (non inclus) sont traités)
alors Jour_à_accepter_r
sinon Jour_à_refuser_r.

2.1.8. VERIFICATION THEME EXISTANT.

Objectif : vérifier qu'un thème à enregistrer comme constituant d'un menu est un thème connu du S.I.

Message-donnée : Nom_thème_d.

Propriété : Nom_thème_d est syntaxiquement valide.

Consultation MSI : Thème.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règle : si Nom_thème_d est le nom d'un thème connu du S.I.
alors Nom_thème_à_accepter_r
sinon Nom_thème_à_refuser_r.

2.1.9. VERIFICATION PLAT EXISTANT.

Objectif : vérifier qu'un plat à enregistrer comme constituant d'un menu pour un thème est un plat connu du S.I.

Message-donnée : Nom_plat_d.

Propriété : Nom_plat_d est syntaxiquement valide.

Consultation MSI : Plat.

Messages-résultats : Nom_plat_à_accepter_r
ou Nom_plat_à_refuser_r.

Action MSI : /.

Règle : si Nom_plat_d est le nom d'un plat connu du S.I.
alors Nom_plat_à_accepter_r
sinon Nom_plat_à_refuser_r.

2.1.10. VERIFICATION DE L'APPARTENANCE D'UN PLAT A UN THEME.

Objectif : vérifier qu'un plat à enregistrer comme constituant d'un menu pour un thème est un plat possible pour ce thème.

Messages-données : - Nom_plat_d;
- Nom_thème_d.

Propriétés : Nom_plat_d et Nom_thème_d sont syntaxiquement valides et sont respectivement des noms de plats et de thèmes connus du S.I.

Consultation MSI : Appartenance.

Messages-résultats : Plat_à_accepter_r
ou Plat_à_refuser_r.

Action MSI : /.

Règle : si Nom_plat_d est le nom d'un plat possible pour le thème de nom Nom_thème_d
alors Plat_à_accepter_r
sinon Plat_à_refuser_r.

2.1.11. VERIFICATION DE L'UNICITE D'UN THEME DANS UN MENU.

Objectif : vérifier qu'un thème à enregistrer comme constituant d'un menu apparaît une seule fois dans l'ensemble des thèmes de ce menu.

Messages-données : - Nom_thème_à_vérifier_d;
- n Nom_thème_d.

Propriétés : - les n+1 Nom_thème_(à_vérifier)d sont des noms de thèmes connus du S.I. et sont les noms des thèmes constituant d'un même menu.

Consultation MSI : /.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règles : s'il existe un Nom_thème_d tel que Nom_thème_à_vérifier_d lui soit égal
alors Nom_thème_à_accepter_r
sinon Nom_thème_à_refuser_r.

2.1.12. VERIFICATION DU NOMBRE DES CONSTITUANTS D'UN MENU.

Objectif : vérifier que le nombre des plats constituants d'un menu à enregistrer est inférieur ou égal à une borne.

Messages-données : - n Nom_plat_d;
- Borne_d.

Propriétés : - les n Nom_plat_d sont des noms de plats connus du S.I. et sont les noms des plats constituants d'un même menu et sont valides au niveau de leur appartenance à un thème, de même, dans l'ensemble des thèmes auxquels ces plats appartiennent, il n'y a pas deux thèmes identiques;
- Borne_d est un entier positif ou nul.

Consultation MSI : /.

Messages-résultats : Menu_à_accepter_r
ou Menu_à_refuser_r.

Action MSI : /.

Règle : si n <= Borne_d
alors Menu_à_accepter_r
sinon Menu_à_refuser_r.

2.1.13. VERIFICATION DE LA COHERENCE DU NOMBRE DE PERSONNES D'UN MENU.

Objectif : vérifier que le nombre de personnes d'un menu à enregistrer est cohérent vis-à-vis du nombre de plats de ce menu.

Messages-données : - Nombre_personnes_d;
- n Nom_plat_d.

Propriétés : - les n Nom_plat_d sont des noms de plats connus du S.I. et sont les plats constituants d'un même menu;
- Nombre_personnes_d est le nombre de personnes du menu constitué des n plats et est un entier positif ou nul;
- les n Nom_plat_d sont valides au niveau de leur appartenance à un thème et au niveau de leur nombre, de même, dans l'ensemble des thèmes auxquels ces plats appartiennent, il n'y a pas deux thèmes identiques.

Consultation MSI : /.

Messages-résultats : Nombre_personnes_à_accepter_r
ou Nombre_personnes_à_refuser_r.

Action MSI : /.

Règles : - si n<>0 et Nombre_personnes_d <= 0
alors Nombre_personnes_à_refuser_r;
- si n<>0 et Nombre_personnes_d > 0
alors Nombre_personnes_à_accepter_r.

2.1.14. VERIFICATION DU NOMBRE DE PERSONNES D'UN MENU.

Objectif : vérifier que le nombre de personnes d'un menu à enregistrer est inférieur ou égal à une borne.

Messages-données : - Nombre_personnes_d;
- Borne_d.

Propriétés : - Nombre_personnes_d cohérent;
- Borne_d est un entier positif ou nul.

Consultation MSI : /.

Messages-résultats : Nombre_personnes_à_accepter_r
ou Nombre_personnes_à_refuser_r.

Action MSI : /.

Règle : si Nombre_personnes_d <= Borne_d
alors Nombre_personnes_à_accepter_r
sinon Nombre_personnes_à_refuser_r.

2.1.15. AFFICHAGE DES INGREDIENTS D'UN MENU.

Objectif : afficher à l'écran les informations concernant les produits entrant dans la composition des plats d'un menu.

Messages-données : - Date_menu_d;
- Repas_menu_d.

Propriétés : (Date_menu_d, Repas_menu_d) identifie un menu connu du S.I.

Consultation MSI : Composition_produit, Produit.

Messages-résultats : Affichage_effectué_r
ou Affichage_non_effectué_r.

Règles : si l'affichage des ingrédients du menu identifié par (Date_menu_d, Repas_menu_d) a été effectué, alors Affichage_effectué_r sinon Affichage_non_effectué_r.

2.1.16. IMPRESSION DES INGREDIENTS D'UN MENU.

Idem fonction précédente, sauf qu'il s'agit de l'impression des ingrédients et non de l'affichage.

2.2. phase MODIFICATION DE MENUS.

2.2.1. ENREGISTREMENT DES MODIFICATIONS D'UN MENU.

Objectif : enregistrer les modifications validées apportées à un menu.

Messages-données : - Date_menu_d;
- Repas_menu_d;
- Nombre_personnes_menu_d;
- m couples (Plat_à_enlever_d, Thème_plat_à_enlever_d);
- n couples (Plat_à_ajouter_d, Thème_plat_à_ajouter_d).

Propriétés : - les messages-données sont les informations concernant un même menu;
- Date_menu_d, Repas_menu_d, Nombre_personnes_menu_d, les m+n couples et leurs composantes, valides;
- (Date_menu_d, Repas_menu_d) identifie un menu connu du S.I. et qui est celui pour lequel on veut enregistrer des modifications;
- le nombre de plats composant le menu modifié est valide;
- Nombre_personnes_menu_d est le nouveau nombre de personnes du menu;
- les m Plat_à_enlever_d sont les noms des plats à supprimer du menu initial (avant modification), et les m Thème_plat_à_enlever_d sont les noms des thèmes auxquels ces plats sont associés dans le menu initial;
- les n Plat_à_ajouter_d sont les noms des plats à ajouter au menu initial et les n Thème_plat_à_enlever_d sont les noms des thèmes auxquels ces plats sont associés dans le menu modifié.

Consultation MSI : /.

Message-résultat : /.

Actions MSI : - mise-à-jour de la valeur de l'attribut Nombre_personnes_menu du Menu identifié par (Date_menu_d, Repas_menu_d);
- mise-à-jour de la valeur de l'attribut Modifie_plat_constitution d'occurrences de Constitution;
- ajout de p occurrences de Constitution identifiées par un quadruplet (Date_menu_d,Repas_menu_d, Plat_à_ajouter_d, Thème_plat_à_ajouter_d);

- mise-à-jour de la valeur de l'attribut Modifie_menu du menu identifié par (Date_menu_d, Repas_menu_d);
- mise-à-jour de la participation d'occurrences de Plat à des occurrences de Constitution.

Règles : - la valeur de l'attribut Modifie_plat_constitution des occurrences de Constitution identifiées par

-a : (Date_menu_d, Repas_menu_d, Plat_à_enlever_d, Thème_plat_à_enlever_d), pour tout (Plat_à_enlever_d, Thème_plat_à_enlever_d), devient "ENLEVE";

-b : (Date_menu_d, Repas_menu_d, Plat_à_ajouter_d, Thème_plat_à_ajouter_d), pour tout (Plat_à_ajouter_d, Thème_plat_à_ajouter_d), devient "AJOUTE";

- pour toute occurrence de Constitution, avant la modification, identifiée par (Date_menu_d, Repas_menu_d, thème) [thème appartenant à {Thème_à_ajouter_d}], et pour laquelle la valeur de l'attribut Modifie_plat_constitution est "ENLEVE", à cette occurrence participe l'occurrence de Plat identifiée par plat, avec plat appartenant à {(Plat_à_ajouter_d, Theme_à_ajouter_d)}.

- la valeur de l'attribut Modifie_menu du Menu identifié par (Date_menu_d, Repas_menu_d) devient "VRAI".

2.2.2. VERIFICATION REPAS EXISTANT.

Idem la fonction du même nom de la phase *Création d'un menu*, sauf l'objectif : vérifier qu'un repas pour lequel on veut enregistrer des modifications pour un menu est un repas connu du S.I.

2.2.3. VERIFICATION JOUR NON ANTERIEUR AU JOUR COURANT.

Idem la fonction du même nom de la phase *Création d'un menu* sauf l'objectif : vérifier qu'une date pour laquelle on veut enregistrer des modifications pour un menu est la date d'un jour non antérieur au jour courant.

2.2.4. VERIFICATION REPAS TRAITE.

Objectif : vérifier qu'un repas pour lequel on veut enregistrer des modifications pour un menu est un repas traité.

Messages-données : - Nom_repas_d;
- Date_repas_d.

Propriétés : - Nom_repas_d est le nom d'un repas connu du S.I.;
- Date_repas_d est syntaxiquement valide.

Consultation MSI : Préparation.

Messages-résultats : Repas_à_accepter_r
ou Repas_à_refuser_r.

Action MSI : /.

Règle : si le repas Nom_repas_d est traité à la date Date_repas_d
alors Repas_à_accepter_r
sinon Repas_à_refuser_r.

2.2.5. VERIFICATION THEME EXISTANT.

Idem la fonction du même nom de la phase *Création d'un menu* sauf l'objectif : vérifier qu'un thème à ajouter comme constituant à un menu existant est un thème connu du S.I.

2.2.6. VERIFICATION PLAT EXISTANT.

Idem la fonction du même nom de la phase *Création d'un menu* sauf l'objectif : vérifier qu'un plat à ajouter comme constituant à un menu existant, pour un thème, est un plat connu du S.I.

2.2.7. VERIFICATION DE L'APPARTENANCE D'UN PLAT A UN THEME.

Idem la fonction du même nom de la phase *Création d'un menu* sauf l'objectif : vérifier qu'un plat à ajouter comme constituant à un menu existant, pour un thème, est un plat possible pour ce thème.

2.2.8. VERIFICATION THEME EXISTANT DANS UN MENU.

Objectif : vérifier qu'un thème à supprimer de l'ensemble des thèmes constituants d'un menu appartient à cet ensemble de thèmes.

Messages-données : - Nom_thème_d;
- Date_menu_d;
- Repas_menu_d.

Propriétés : - Nom_thème_d est syntaxiquement valide;
- (Date_menu_d, Repas_menu_d) identifie un Menu connu du S.I. et qui est le Menu pour lequel on veut enregistrer des modifications, et dont la date est non antérieure à la date du jour courant.

Consultation MSI : Constitution.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règles : si Nom_thème_d est le nom d'un thème enregistré comme constituant du Menu identifié par (Date_menu_d, Repas_menu_d)
alors Nom_thème_à_accepter_r
sinon Nom_thème_à_refuser_r.

2.2.9. VERIFICATION EXISTENCE D'UN PLAT POUR UN THEME DANS UN MENU.

Objectif : vérifier qu'un plat à supprimer des constituants d'un menu pour un thème est effectivement enregistré comme constituant de ce menu pour ce thème.

Messages-données : - Date_menu_d;
- Repas_menu_d;
- Nom_thème_d;
- Nom_plat_d.

Propriétés : - (Date_menu_d, Repas_menu_d) identifie un Menu connu du S.I. et qui est le Menu pour lequel on veut enregistrer des modifications , et dont la date est non antérieure à la date du jour courant;
- Nom_thème_d est le nom d'un thème constituant du Menu identifié par (Date_menu_d, Repas_menu_d);
- Nom_plat_d est le nom d'un plat connu du S.I.

Consultation MSI : Constitution.

Messages-résultats : Nom_plat_à_accepter_r
ou Nom_plat_à_refuser_r.

Action MSI : /.

Règle : si Nom_plat_d est le nom d'un plat enregistré comme constituant du Menu identifié par (Date_menu_d, Repas_menu_d) pour le thème de nom Nom_thème_d
alors Nom_plat_à_accepter_r
sinon Nom_plat_à_refuser_r.

2.2.10. VERIFICATION DE L'UNICITE D'UN THEME DANS UN MENU.

Objectif : vérifier qu'un thème à ajouter comme constituant d'un menu dont on veut enregistrer des modifications apparaît une seule fois dans l'ensemble des thèmes de ce menu.

Messages-données : - Nom_thème_à_vérifier_d;
- Date_menu_d;
- Repas_menu_d;
- m Nom_thème_à_ajouter_menu_d;
- n Nom_thème-à_supprimer_menu_d.

Propriétés : - les 1+m+n Nom_thème_(à_vérifier, à_ajouter_menu, à_supprimer_menu)d sont des noms de thèmes connus du S.I.;
- (Date_menu_d, Repas_menu_d) identifie un Menu connu du S.I. et qui est celui pour lequel on veut enregistrer des modifications , et dont la date est non antérieure à la date du jour courant;
- les m Nom_thème_à_ajouter_menu_d sont, avec Nom_thème_à_vérifier_d, les noms des thèmes dont l'ajout au menu identifié par (Date_menu_d, Repas_menu_d) a été demandé;
- les n Nom_thème_à_supprimer_menu_d sont les noms des thèmes dont la suppression du menu identifié par (Date_menu_d, Repas_menu_d) a été demandée.

Consultation MSI : Constitution.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règle : si il existe un thème dont le nom appartient à {{noms de thèmes constituant le menu identifié par (Date_menu_d, Repas_menu_d)} U {m Nom_thème_à_ajouter_menu_d} \ {n Nom_thème_à_supprimer_menu_d}} et dont le nom est égal à Nom_thème_à_vérifier_d
alors Nom_thème_à_refuser_r
sinon Nom_thème_à_accepter_r.

2.2.11. VERIFICATION DU NOMBRE DES CONSTITUANTS D'UN MENU.

Objectif : vérifier que le nombre de plats constituants d'un menu dont l'enregistrement des modifications a été demandé est inférieur ou égal à une borne.

Messages-données : - Date_menu_d;
- Repas_menu_d;
- m Nom_plat_à_ajouter_menu_d;
- n Nom_plat_à_supprimer_menu_d;
- Borne_d.

Propriétés : - (Date_menu_d, Repas_menu_d) identifie un menu connu du S.I. et qui est celui pour lequel on veut enregistrer des modifications, et dont la date est non antérieure à la date du jour courant;
- les m Nom_plat_à_ajouter_menu_d sont les noms des plats dont l'ajout au Menu identifié par (Date_menu_d, Repas_menu_d) a été demandé;
- les n Nom_plat_à_supprimer_menu_d sont les noms des plats dont la suppression du Menu identifié par (Date_menu_d, Repas_menu_d) a été demandée;
- les m+n Nom_plat_(à_ajouter_menu, à_supprimer_menu)d sont des noms de plats connus du S.I. et sont valides au niveau de l'appartenance à un thème; de même, dans l'ensemble des thèmes du menu modifié, il n'y a pas deux thèmes identiques;
- Borne_d est un entier positif ou nul.

Consultation MSI : Constitution.

Messages-résultats : Menu_à_accepter_r
ou Menu_à_refuser_r.

Action MSI : /.

Règle : si (# {plats constituants le menu identifié par (Date_menu_d, Repas_menu_d)} + m - n) <= Borne_d
alors Menu_à_accepter_r
sinon Menu_à_refuser_r.

2.2.12. VERIFICATION DE LA COHERENCE DU NOMBRE DE PERSONNES D'UN MENU.

Objectif : vérifier que le nombre de personnes d'un menu dont l'enregistrement des modifications a été demandé est cohérent vis-à-vis du nombre de plats de ce menu.

Messages-données : - Date_menu_d;
- Repas_menu_d;
- m Nom_plat_à_ajouter_menu_d;
- n Nom_plat_à_supprimer_menu_d;
- Nombre_personnes_d.

Propriétés : idem les propriétés de la fonction 11 de cette phase, sauf, en plus :

- le nombre de plats du menu modifié est valide;
- Nombre_personnes_d est le nombre de personnes du menu modifié et est un entier positif ou nul.

Consultation MSI : /.

Messages-résultats : Nombre_personnes_à_accepter_r
ou Nombre_personnes_à_refuser_r.

Action MSI : /.

Règle : si (# {plats constituant le menu identifié par (Date_menu_d, repas_menu_d)} + m - n) <> 0
et Nombre_personnes_d <= 0,
alors Nombre_personnes_à_refuser_r;
et Nombre_personnes_d > 0,
alors Nombre_personnes_à_accepter_r.

2.2.13. VERIFICATION DU NOMBRE DE PERSONNES D'UN MENU.

Objectif : vérifier que le nombre de personnes d'un menu dont l'enregistrement des modifications a été demandé est inférieur ou égal à une borne.

Messages-données : - Nombre_personnes_d;
- Borne_d.

Propriétés : - Nombre_personnes_d cohérent;
- Borne_d est un entier positif ou nul.

Consultation MSI : /.

Messages-résultats : Nombre_personnes_à_accepter_r
ou Nombre_personnes_à_refuser_r.

Action MSI : /.

Règle : si Nombre_personnes_d <= Borne_d
alors Nombre_personnes_à_accepter_r
sinon Nombre_personnes_à_refuser_r.

2.2.14. VERIFICATION NON EXISTENCE THEME DANS MENU INITIAL.

Objectif : vérifier qu'un thème à ajouter à un menu n'existe pas dans le menu initial.

Messages-données : - Nom_thème_d;
- Date_menu_d;
- Repas_menu_d.

Propriétés : - Nom_thème_d est syntaxiquement valide;
- (Date_menu_d, Repas_menu_d) identifie un menu connu du S.I. et qui est celui pour lequel on veut enregistrer des modifications, et dont la date est non antérieure à la date du jour courant.

Consultation MSI : Constitution.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règles : si Nom_thème_d est le nom d'un thème non constituant du menu identifié par (Date_menu_d, Repas_menu_d), alors Nom_thème_à_accepter_r, sinon Nom_thème_à_refuser_r.

2.2.15. VERIFICATION UNICITE THEME DANS LISTE A AJOUTER.

Objectif : vérifier qu'un thème à ajouter à un menu est unique dans la liste des thèmes à ajouter à ce menu.

Messages-données : - Nom_thème_d;
- Liste_thème_d.

Propriété : - Nom_thème_d est syntaxiquement valide;
- Liste_thème_d contient Nom_thème_d.

Consultation MSI : /.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règles : si Nom_thème_d est unique dans Liste_thème_d,
alors Nom_thème_à_accepter_r,
sinon Nom_thème_à_refuser_r.

2.2.16. VERIFICATION UNICITE THEME DANS LISTE A ENLEVER.

Objectif : vérifier qu'un thème à ajouter à un menu est unique dans la liste des thèmes à enlever de ce menu.

Messages-données : - Nom_thème_d;
- Liste_thème_d.

Propriété : - Nom_thème_d est syntaxiquement valide;
- Liste_thème_d contient Nom_thème_d.

Consultation MSI : /.

Messages-résultats : Nom_thème_à_accepter_r
ou Nom_thème_à_refuser_r.

Action MSI : /.

Règles : si Nom_thème_d est unique dans Liste_thème_d,
alors Nom_thème_à_accepter_r,
sinon Nom_thème_à_refuser_r.

2.2.17. AFFICHAGE DES INGREDIENTS D'UN MENU.

Idem fonction du même nom, phase précédente.

2.2.18. IMPRESSION DES INGREDIENTS D'UN MENU.

Idem fonction du même nom, phase précédente.

2.3. phase CONSULTATION DE MENUS.

2.3.1. RECHERCHE D'UN MENU.

Objectif : afficher à l'écran toutes les informations concernant un menu.

Messages-données : - Date_menu_d;
- Repas_menu_d.

Propriétés : (Date_menu_d, Repas_menu_d) identifie un Menu connu du S.I. et qui est celui que l'on veut consulter, et dont la date est non antérieure à la date du jour courant.

Consultation MSI : Menu, Constitution, Plat, Thème.

Messages-résultats : Affichage_effectué
ou Affichage_non_effectué.

Action MSI : /.

Règle : si l'affichage de Menu identifié par (Date_menu_d, Repas_menu_d) a été effectué, alors Affichage_effectué, sinon Affichage_non_effectué.

2.3.2. VERIFICATION REPAS EXISTANT.

Idem la fonction du même nom dans la phase *Création d'un menu* sauf l'objectif : vérifier qu'un repas pour lequel on veut obtenir les informations concernant un menu est un repas connu du S.I.

2.3.3. VERIFICATION REPAS TRAITE.

Idem la fonction du même nom dans la phase *Modification d'un menu* sauf l'objectif : vérifier qu'un repas pour lequel on veut obtenir les informations concernant un menu est un repas traité.

2.3.4. VERIFICATION JOUR NON ANTERIEUR AU JOUR COURANT.

Idem la fonction du même nom dans le phase *Création d'un menu* sauf l'objectif : vérifier qu'une date pour laquelle on veut obtenir les informations concernant un menu est la date d'un jour non antérieur au jour courant.

2.3.5. AFFICHAGE DES INGREDIENTS D'UN MENU.

Idem fonction du même nom, phase *Création de menus.*

2.3.6. IMPRESSION DES INGREDIENTS D'UN MENU.

Idem fonction du même nom, phase *Création de menus.*

2.4. phase IMPRESSION DE MENUS.

Idem la phase *Consultation d'un menu* sauf qu'il s'agit ici d'imprimer les informations concernant un menu et sauf une fonction supplémentaire :

2.4.1. MISE-A-JOUR DE L'ATTRIBUT IMPRIME_MENU.

Objectif : mettre à jour la valeur de l'attribut Imprime_menu du menu imprimé.

Mesages-données : - Date_menu_d;
- Repas_menu_d.

Propriétés : (Date_menu_d, Repas_menu_d) identifie un Menu connu du S.I. et qui est celui que l'on veut imprimer, et dont la date est non antérieure à la date du jour courant.

Consultation MSI : Menu.

Message-résultat : /.

Action MSI : mise-à-jour de la valeur de l'attribut Imprime_menu du Menu identifié par (Date_menu_d, Repas_menu_d).

Règle : la valeur de l'attribut Imprime_menu de cette occurence de Menu devient VRAI.

2.5. phase SUPPRESSION DE MENUS.

Objectif : supprimer tous les menus dont la date est antérieure à la date du jour courant.

Message-donnée : Date_courante_d.

Propriété : Date_courante_d est la date du jour courant.

Consultation MSI : Préparation.

Message-résultat : /.

Actions MSI : - suppression de n Préparation et de n Menu;
- suppression de m Constitution.

Règle : toute occurrence de Menu participant à une association Préparation dont l'attribut Date_préparation a une valeur strictement inférieure à Date_courante_d est supprimée, et toutes les occurrences de Constitution auxquelles participe ce Menu sont également supprimées.

APPLICATION PRODUITS DE BASE.

1. DEFINITION DES PHASES.

1.1. CREATION D'UNE LISTE DES PRODUITS DE BASE NECESSAIRES.

Objectif : permettre à l'utilisateur de créer une liste de produits de base facultatifs pour une période à choisir par lui-même, et enregistrer cette liste si elle est valide; de même, enregistrer la liste des produits de base obligatoires pour la période.

Messages-données : - Paramètres_utilisateurs.

Propriétés : - Paramètres_utilisateurs est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser la phase en fonction de l'utilisateur.

Consultation MSI : Produit, Sous-type, Produit_de_base, Préparation.

Messages-résultats : /.

Actions MSI : - ajout d'une Liste_produits_de_base_nécessaires;
- ajout de m Ligne_liste_produits_de_base_nécessaires.

où m est le nombre de (produits de base facultatifs spécifiés par l'utilisateur + produits de base obligatoires).

1.2. MODIFICATION D'UNE LISTE DES PRODUITS DE BASE NECESSAIRES.

Objectif : permettre à l'utilisateur de modifier une liste de produits de base précédemment créée, c'est-à-dire d'y ajouter/enlever des produits de base facultatifs, et enregistrer les modifications si elles sont valides.

Messages-données : - Paramètres_utilisateurs.

Propriétés : - Paramètres_utilisateurs est un ensemble d'informations associées à l'utilisateur et destinées à paramétriser la phase en fonction de l'utilisateur.

Consultation MSI : Liste_produits_de_base_nécessaires; Ligne_liste_produits_de_base; Produit, Sous_type, Produit_de_base.

Messages-résultats : /.

Actions MSI : - ajout de m Ligne_liste_produits_de_base;
- suppression de n Ligne_liste_produits_de_base.

1.3. SUPPRESSION DE LISTES DE PRODUITS DE BASE NECESSAIRES.

Objectif : supprimer automatiquement les listes de produits de base nécessaires dont la date de fin de période est antérieure à la date du jour.

Message-donnée : Date_courante_d.

Propriété : Date_courante_d est la date du jour courant.

Consultation MSI : Liste_produits_de_base_nécessaires.

Message-résultat : /.

Actions MSI : - suppression de m Liste_produits_de_base_nécessaires;
- suppression de n Ligne_liste_produits_de _base.

2. DEFINITION DES FONCTIONS.

2.1. phase CREATION D'UNE LISTE DES PRODUITS DE BASE NECESSAIRES.

2.1.1. ENREGISTREMENT DES PRODUITS DE BASE NECESSAIRES.

Objectif : enregistrer les produits de base nécessaires pour une période donnée, valides.

Messages-données : - Date_début_liste_d;
- Date_fin_liste_d;
- n Produit_base_facultatif_d.

Propriétés : - Date_début_liste_d et Date_fin_liste_d valides et représentent respectivement la date du début et la date de la fin de la période pour laquelle on veut enregistrer des produits de base nécessaires;
- les n Produit_base_facultatif_d sont valides et sont les noms des produits de base facultatifs demandés pour la période.

Consultation MSI : Produit, Sous-type, Produit_de_base.

Message-résultat : /.

Actions MSI : - ajout d'une Liste_produits_de_base_nécessaires;
- ajout de m Ligne_liste_produits_de_base.

Règles : - il y a p produits de base obligatoires pour la période délimitée par Date_début_liste_d et Date_fin_liste_d, p >= 0;
- pour cette même période, la suite ST des m produits de base nécessaires (à chacun desquels correspond une Ligne_liste_produits_de_base à ajouter) est formée de la suite ST1 des n Produit_base_facultatif_d et de la suite ST2 des p produits de base obligatoires pour cette période (p+n=m).

2.1.2. VERIFICATION DELIMITATION D'UNE PERIODE.

Objectif : vérifier que la date de début d'une période pour laquelle on veut enregistrer des produits de base nécessaires est non ultérieure à la date de fin de cette période.

Messages-données : - Date_début_d;
- Date_fin_d.

Propriétés : Date_début_d et Date_fin_d syntaxiquement valides.

Consultation MSI : /.

Messages-résultats : Délimitation_à_accepter_r
ou Délimitation_à_refuser_r.

Action MSI : /.

Règle : si Date_début_d <= Date_fin_d
alors Délimitation_à_accepter_r
sinon Délimitation_à_refuser_r.

2.1.3. VERIFICATION JOUR NON ANTERIEUR AU JOUR COURANT.

Idem fonction du même nom de la phase *Création d'un menu* de l'application *Menu* sauf l'objectif : vérifier que la date de début d'une période pour laquelle on veut enregistrer des produits de base nécessaires est la date d'un jour non antérieur au jour courant.

2.1.4. VERIFICATION JOURS TRAITES POUR UNE PERIODE.

Objectif : vérifier que les jours d'une période pour laquelle on demande l'enregistrement de produits de base nécessaires sont tous des jours traités (uniquement si l'on travaille dans le premier scénario de création de liste d'achats).

Messages-données : - Date_début_d;
- Date_fin_d.

Propriétés : - Date_début_d et Date_fin_d sont syntaxiquement valides;
- Date_début_d <= Date_fin_d;
- Date_début_d >= date du jour courant.

Consultation MSI : Préparation.

Messages-résultats : Période_à_accepter_r
ou Période_à_refuser_r.

Action MSI : /.

Règle : pour toute date D telle que Date_début_d <= D <= Date_fin_d,
si D est la date d'un jour traité
alors Période_à_accepter_r
sinon Période_à_refuser_r.

2.1.5. VERIFICATION PRODUIT DE BASE FACULTATIF EXISTANT.

Objectif: vérifier qu'un produit demandé pour une période est un produit de base facultatif connu du S.I.

Message-donnée : Nom_produit_d.

Propriété : Nom_produit_d est syntaxiquement valide.

Consultation MSI : Produit, Sous_type, Produit_de_base.

Messages-résultats : Nom_produit_à_accepter_r
ou Nom_produit_à_refuser_r.

Action MSI : /.

Règle : si Nom_produit_d est le nom d'un produit de base facultatif connu du S.I. alors Nom_produit_à_accepter_r
sinon Nom_produit_à_refuser_r.

2.1.6. VERIFICATION DE L'UNICITE D'UN PRODUIT DE BASE FACULTATIF DEMANDE.

Objectif : vérifier qu'un produit de base facultatif dont l'enregistrement est demandé dans une liste de produits de base nécessaires pour une période, n'apparaît qu'une fois dans l'ensemble des produits de base facultatifs dont l'enregistrement dans cette même liste est demandé.

Messages-données : - Nom_produit_à_vérifier_d;
- n Nom_produit_d.

Propriété : les n+1 Nom_produit_(à_vérifier_)d sont des noms de produits de base facultatifs connus du S.I. et sont les noms des produits de base facultatifs demandés pour une même période.

Consultation MSI : /.

Messages-résultats : Nom_produit_à_accepter_r
ou Nom_produit_à_refuser_r.

Action MSI : /.

Règle : si il existe un Nom_produit_d égal à Nom_produit_à_vérifier_d
alors Nom_produit_à_refuser_r
sinon Nom_produit_à_accepter_r.

2.2. phase MODIFICATION D'UNE LISTE DES PRODUITS DE BASE NECESSAIRES.

2.2.1 ENREGISTREMENT DES MODIFICATIONS APPORTEES A UNE LISTE DES PRODUITS DE BASE NECESSAIRES.

Objectif : enregistrer les modifications valides apportées à une liste de produits de base nécessaires pour une période.

Messages-données : - Date_début_liste_d;
- Date_fin_liste_d;
- m Nom_produit_à_ajouter_d;
- n Nom_produit_à_supprimer_d.

Propriétés : - (Date_début_liste_d, Date_fin_liste_d) identifie une Liste_-produits_de_base_nécessaires connue du S.I. et qui est celle pour laquelle on veut enregistrer des modifications;
- les (m+n) Nom_produit_(à_ajouter, à_supprimer)menu_d sont les noms des produits de base facultatifs respectivement

à ajouter et à supprimer de la Liste_produits_de_base_nécessaires identifiée par (Date_début_liste_d, Date_fin_liste_d), et sont valides.

Consultation MSI : /.

Message-résultat : /.

Actions MSI : - ajout de m Ligne_liste_produits_de_base;
- suppression de n Ligne_liste_produits_de_base.

Règles : - pour tout Nom_produit_à_ajouter_d, une occurrence de Ligne_lis-te_produits_de_base_nécessaires identifiée par (Date_début_liste_d, Date_fin_liste_d, Nom_produit_à_ajouter_d) est créée;
- pour tout Nom_produit_à_supprimer_d, l'occurrence de Ligne_lis-te_produits_de_base_nécessaires identifiée par (Date_début_liste_d, Date_fin_liste_d, Nom_produit_à_supprimer_d) est supprimée.

2.2.2. VERIFICATION PERIODE EXISTANTE.

Objectif : vérifier que la date de début d'une période pour laquelle on veut modifier la liste des produits de base nécessaires et la date de fin de cette période sont les dates de début et de fin d'une liste de produits de base.

Messages-données : - Date_début_d;
- Date_fin_d.

Propriétés : Date_début_d et Date_fin_d sont syntaxiquement valides et sont les dates de début et de fin d'une période pour laquelle on veut modifier la liste des produits de base nécessaires.

Consultation MSI : Liste_produits_de_base_nécessaires.

Messages-résultats : Période_à_accepter_r
ou Période_à_refuser_r.

Action MSI : /.

Règles : si il existe une occurrence de Liste_produits_de_base_nécessaires telle que cette liste ait une Date_début_période_liste = Date_début_d et Date_fin_période_liste_d = Date_fin_d
alors Période_à_accepter_r
sinon Période_à_refuser_r.

2.2.3. VERIFICATION PRODUIT DE BASE FACULTATIF EXISTANT.

Idem la fonction du même nom de la phase *Création d'une liste des produits de base nécessaires*, sauf l'objectif : vérifier qu'un produit dont l'ajout à une liste de produits de base nécessaires a été demandé est un produit de base facultatif connu du S.I.

2.2.4. VERIFICATION PRODUIT DE BASE FACULTATIF EXISTANT DANS LISTE.

Objectif : vérifier qu'un produit de base facultatif dont la suppression d'une liste de produits de base nécessaires pour une période a été demandée, fait partie de l'ensemble des produits de base de cette liste.

Messages-données : - Date_début_d;
- Date_fin_d;
- Nom_produit_d.

Propriétés : - Nom_produit_d est syntaxiquement valide;
- (Date_début_d, Date_fin_d) identifie une liste de produits de base nécessaires connue du S.I., et qui est celle pour laquelle on veut enregistrer des modifications.

Consultation MSI : Ligne_liste_produits_de_base.

Messages-résultats : Nom_produit_à_accepter_r
ou Nom_produit_à_refuser_r.

Action MSI : /.

Règles : si le nom Nom_produit_d existe dans l'ensemble des noms des produits de base constituant la liste identifiée par (Date_début_d, Date_fin_d)
alors Nom_produit_à_accepter_r
sinon Nom_produit_à_refuser_r.

2.2.5. VERIFICATION DE L'UNICITE D'UN PRODUIT DE BASE FACULTATIF.

Objectif : vérifier qu'un produit de base facultatif dont l'ajout à une liste de produits de base nécessaires pour une période a été demandé apparaît une seule fois dans l'ensemble de ces produits.

Messages-données : - Nom_produit_à_vérifier_d;
- Date_début_d;
- Date_fin_d;
- m Nom_produit_à_ajouter_liste_d;
- n Nom_produit_à_supprimer_liste_d.

Propriétés : - les 1+m+n Nom_produit_(à_vérifier, à_ajouter_liste, à_supprimer_liste)_d sont des noms de produits de base facultatifs connus du S.I.;
- (Date_début_d, Date_fin_d) identifie une Liste_produits_de_base_nécessaires et qui est celle pour laquelle on veut enregistrer des modifications;
- les m Nom_produit_à_ajouter_liste_d sont, avec Nom_produit_à_vérifier_d, les noms des produits dont l'ajout à la liste identifiée par (Date_début_d, Date_fin_d) a été demandé;
- les n Nom_produit_à_supprimer_liste_d sont les noms des produits dont la suppression de la liste identifiée par (Date_début_d, Date_fin_d) a été demandée.

Consultation MSI : /.

Messages-résultats : Nom_produit_à_accepter_r
ou Nom_produit_à_refuser_r.

Règle : si il existe un nom N de produit appartenant à {{noms de produits constituant la liste identifiée par (Date_début_d, Date_fin_d) U {Nom_produit_à_ajouter_liste_d} \ {Nom_produit_à_supprimer_d}} et qui soit égal à Nom_produit_à_vérifier_d
alors Nom_produit_à_accepter_r
sinon Nom_produit_à_refuser_r.

2.3. phase SUPPRESSION DE LISTES DE PRODUITS DE BASE NECESSAIRES.

2.3.1. SUPPRESSION LISTES PRODUITS DE BASE.

Objectif : supprimer les listes de produits de base nécessaires dont la date de fin de période est antérieure à la date du jour courant.

Message-donnée : Date_courante_d.

Propriété : Date_courante_d est la date du jour courant.

Consultation MSI : Liste_produits_de_base_nécessaires.

Message-résultat :/.

Actions MSI : - suppression de n Liste_produits_de_base_nécessaires;
- suppression de m Ligne_liste_produits_de_base.

Règle : toute occurrence de Liste_produits_de_base_nécessaires dont l'attribut Date_fin_période_liste a une valeur inférieure à la valeur de Date_courante_d, est supprimée, et les occurrences de Ligne_liste_produits_de_base auxquelles participe cette occurrence sont supprimées.

APPLICATION LISTE D'ACHATS.

1. DEFINITION DES PHASES.

1.1. PREPARATION LISTE PRODUITS NECESSAIRES.

Objectif : déterminer les produits nécessaires et les quantités nécessaires de ces produits pour une période donnée valide.

Messages-données : - Date_jour_courant_d;
- Date_début_période_d;
- Date_fin_période_d;
- Type_liste_d : - Liste_menu_d;
ou - Liste_produits_base_et_menu_d.
- <Date_début_ss_période, Date_fin_ss_période, Achats_faits_d> :
- Achats_faits_d : - Courses_faites_d;
ou - Courses_pas_faites_d.

Propriétés : - Date_jour_courant_d est la date du jour courant;
- Date_début_période_d est la date du début de la période pour laquelle on veut établir les listes d'achats;
- Date_fin_période_d est la date de fin de la période pour laquelle on veut établir les listes d'achats;
- Type_liste_d est le scénario de création de listes d'achats que l'on doit suivre;
- Achats_faits_d dit si oui ou non on a déjà fait les courses pour la période délimitée par <Date_début_ss_période_d, Date_fin_ss_période_d>, utile lorsqu'on fait une ressortie de liste.
- Les sous-périodes délimitées par <Date_début_ss_période_d, Date_fin_ss_période_d> sont une partition de la période délimitée par <Date_début_période_d, Date_fin_période_d>.

Consultation MSI : Repas, Menu, Plat, Thème, Produit, Produit_de_base, Préparation, Constitution, Composition_plat, Sous-type;

Message-résultat : Liste_produits_nécessaires <Nom_produit_r, Quantité_nécessaire_produit_r>; la liste des produits nécessaires pour la période et les quantités nécessaires de ceux-ci, exprimées en nombre d'unités de conditionnement d'utilisation.

Action MSI : /.

1.2. PREPARATION LISTE DES ACHATS ET AFFICHAGE DU MONTANT NECESSAIRE.

Objectif : déterminer les produits à acheter, la quantité à acheter de ces produits, la somme d'argent nécessaire pour faire les achats en fonction du contenu du portefeuille.

Messages-données : - n <Nom_produit_d, Quantité_nécessaire_produit_d, Quantité_stock_d>;
- contenu du portefeuille.

Propriétés : - chaque triplet est composé d'un nom du produit nécessaire pour la période, de la quantité nécessaire de ce produit pour la période et de la quantité restant actuellement en stock de ce produit;
- Quantité_nécessaire_d et Quantité_stock_d sont exprimées dans la même unité de mesure Mesure_produit du produit;
- le contenu du portefeuille est l'énumération en termes de billets et de pièces du contenu du portefeuille (budget alimentation) de l'utilisateur.

Consultation MSI : Produit.

Messages-résultats : - Liste_produits_à_acheter_r <Nom_produit_r, Quantité_à_acheter_r>; la liste des produits à acheter pour la période et les quantités à acheter de ceux-ci exprimées en nombre d'unités de conditionnement de vente du produit;
- Liste_prix_produits_à_acheter_r <Nom_produit_r, Prix_unitaire_produit_r, Prix_global_produit>; la liste des produits à acheter pour la période, le prix unitaire de ceux-ci et le prix global (fonction des quantités à acheter);
- Montant_total_r; le montant de la somme d'argent, exprimée dans l'unité monétaire du pays, nécessaire pour acheter les produits à acheter dans les quantités à acheter.

Action MSI : /.

1.3. SORTIE LISTE ACHATS.

Objectif : imprimer la liste des produits à acheter pour une période.

Messages-données : - Liste_produits_à_acheter_d;
- Liste_prix_produits_à_acheter_d;
- Montant_total_d;

Propriétés : - Liste_produits_à_acheter_d est la liste des produits à acheter pour une période et les quantités à acheter de ceux-ci, exprimées en nombre d'unités de conditionnement de vente;
- Liste_prix_produits_à_acheter_d est la liste des produits à acheter pour la période et le prix unitaire et global de ceux-ci;
- Montant_total_r est le montant de la somme d'argent nécessaire pour acheter les produits à acheter dans les quantités à acheter;

Consultation MSI : Repas, Menu, Plat, Préparation, Constitution.

Message-résultat : /

Actions MSI : - mise à jour de l'attribut Modifie_menu d'occurrences de l'entité Menu;
- mise à jour de l'attribut Modifie_plat_constitution d'occurrences de l'association Constitution.

2. DEFINITION DES FONCTIONS.

2.1. phase PREPARATION LISTE PRODUITS NECESSAIRES.

2.1.1. DETERMINATION DES PRODUITS ET DES QUANTITES NECESSAIRES.

Objectif : déterminer la liste des produits nécessaires et la quantité nécessaire de ces produits pour une période donnée.

Messages-données : - Date_jour_courant_d;
- Date_début_période_d;
- Date_fin_période_d;
- Type_liste_d : - Liste_menu_d; (TL1)
ou - Liste_prod_base_et_menu_d. (TL2)
- <Date_début_ss_période, Date_fin_ss_période, Achats_faits_d>

- Achats_faits_d : - Courses_faites_d; (AF1)
ou - Courses_pas_faites_d. (AF2)

Propriétés : - Date_début_période_d ≤ Date_fin_période_d;
- Date_début_période_d ≥ Date_jour_courant_d;
- si TL2 alors <Date_début_période_d, Date_fin_période_d> délimite une période pour laquelle on a déjà une Liste_produits_base_nécessaires;
- Date_jour_courant_d est la date du jour courant;
- Date_début_période_d est la date du début de la période pour laquelle on veut établir la liste d'achats;
- Date_fin_période_d est la date de fin de la période pour laquelle on veut établir la liste d'achats;
- Type_liste_d indique le scénario de création de liste d'achats que l'on doit suivre;
- Achats_faits dit si oui ou non on a déjà fait les courses pour la période délimitée par <Date_début_période_d, Date_fin_période_d>; utilisé lorsqu'on fait une ressortie de liste.

Consultation MSI : - si TL1 : Repas, Menu, Plat, Thème, Préparation, Constitution, Composition_plat;
- si TL2 : Repas, Menu, Plat, Thème, Produit, Produit_de_base, Préparation, Constitution, Composition_plat, Sous-type.

Message-résultat : Liste_produits_nécessaires <Nom_produit_r, Quantité_nécessaire_produit_r>; la liste des produits nécessaires pour la période et les quantités nécessaires de ceux-ci, exprimées en nombre d'unités de conditionnement d'utilisation.

Action MSI : /.

Règles : - Liste_produits_nécessaires = ensemble de n Ligne_liste_produits_nécessaires;
- une Ligne_liste_produits_nécessaires est composée d'une paire <Nom_produit_r, Quantité_nécessaire_produit_r>;

où - Nom_produit_r est

- le nom d'un produit entrant dans la composition d'un plat d'un menu pour un repas d'un jour compris entre <Date_début_période_d, Date_fin_période_d> compris si TL1;

- le nom d'un produit entrant dans la composition d'un plat d'un menu pour un repas d'un jour compris entre <Date_début_période_d, Date_fin_période_d> compris ou le nom d'un produit de base nécessaire appartenant à la Liste_produits_base_nécessaires identifiée par <Date_début_période_d, Date_fin_période_d> si TL2 et que Modifie_plat_constitution = "AJOUTE".

- Quantité_nécessaire_produit_r =

$$\sum_{\text{Jour}=\text{Date\_début\_ss\_période\_d}}^{\text{Date\_fin\_ss\_période\_d}} \sum_{\text{Menus du jour}} \sum_{\text{Plats constituant le menu}} \sum_{\substack{\text{si Modifie\_plat\_constitution= "AJOUTE"} \\ \text{ou si (Modifie\_plat\_constitution="INCHANGE"} \\ \text{et Courses\_pas\_faites).}}} (\text{Quantité\_produit utilisé du produit dans le plat} * \text{nombre\_personnes\_menu})$$

toutes les sous-périodes.

si TL1;

Date_fin_ss_période_d
:
$\sum\sum\Sigma\Sigma$(Quantité_produit utilisé du produit dans le plat
*nombre_personnes_menu)
: : : : si Modifie_plat_constitution= "AJOUTE"
: : : : ou si (Modifie_plat_constitution="INCHANGE"
: : : : et Courses_pas_faites).
: : : :
: : : Plats constituant le menu
: : Menus du jour
: Jour = Date_début_ss_période_d
toutes les sous-périodes

+

Date_fin_période_d
$\sum\sum$((Date_fin_période - Max(Date_début-période,
: : Date_jour_courant_d)*(Consommation_hebdomadaire)/7)
: :
: période comprise entre Date_début_période_d.
toutes les sous-périodes.

si TL2;

- Max(Date_début_période_d, Date_jour_courant_d) =
Si Date_début_période_d > Date_jour_courant_d
Alors Date_début_période_d
Sinon Date_jour_courant_d

2.1.2. VERIFICATION DATE DEBUT DE PERIODE ULTERIEURE AU JOUR COURANT.

Objectif : Vérifier que la Date_début_période_d pour laquelle on veut sortir une liste d'achats ≥ Date_jour_courant_d.

Messages-données : - Date_début_période_d;
- Date_jour_courant_d.

Propriétés : - Date_début_période et Date_jour_courant sont valides syntaxiquement;
- Date_jour_courant est la date du jour du traitement;
- Date_début_période_d est la date du début de la période pour laquelle on veut établir la liste d'achats.

Consultation MSI : /.

Messages-résultats : - Date_Début_Période_Antérieure;
ou - Date_Jour_Courant_Antérieure.

Action MSI : /.

Règle : si Date_début_période ≥ Date_jour_courant
alors Date_Jour_Courant_Antérieure
sinon Date_Début_Période_Antérieure.

2.1.3. VERIFICATION DATE DEBUT DE PERIODE ANTERIEURE A LA DATE DE FIN DE PERIODE.

Objectif : Vérifier que la Date_début_période de la période pour laquelle on veut sortir une liste d'achats ≤ Date_fin_période de la période pour laquelle on veut sortir une liste d'achats.

Messages-données : - Date_début_période;
- Date_fin_période.

Propriétés : - Date_début_période et Date_fin_période sont valides syntaxiquement;
- Date_début_période_d est la date du début de la période pour laquelle on veut établir la liste d'achats;
- Date_fin_période_d est la date de fin de la période pour laquelle on veut établir la liste d'achats.

Consultation MSI : /.

Messages-résultats : - Date_Début_Période_Antérieure;
ou - Date_Fin_Période_Antérieure.

Action MSI : /.

Règle : si Date_début_période ≥ Date_fin_période
alors Date_Fin_Période_Antérieure
sinon Date_Début_Période_Antérieure.

2.2. phase PREPARATION LISTE DES ACHATS ET AFFICHAGE DU MONTANT NECESSAIRE .

2.2.1. DETERMINATION DES PRODUITS A ACHETER POUR UNE PERIODE.

Objectif : déterminer les produits à acheter pour une période donnée et les quantités à acheter de ces produits.

Message-donnée : - n <Nom_produit_d, Quantité_nécessaire_produit_d, Quantité_stock_d>.

Propriétés : - Quantité_stock_d des produits de la classe "ACHAT UNITE" = 0;
- Quantité_stock_d et Quantité_nécessaire_produit_d sont exprimées en nombre d'unités de conditionnement d'utilisation du produit;
- chaque triplet est composé d'un nom de produit nécessaire pour une période, de la quantité nécessaire de ce produit pour la période et de la quantité restant actuellement en stock de ce produit.

Consultation MSI : /.

Message-résultat : Liste_produits_à_acheter_r <Nom_produit_r, Quantité_stock_r, Quantité_à_acheter_r>; la liste des produits à acheter pour la période et les quantités à acheter de ceux-ci, exprimées en nombre d'unités de conditionnement de vente.

Action MSI : /.

Règles : - Nom_produit_r est le nom d'un produit appartenant à Liste_produits_nécessaires qu'il faut acheter c'est-à-dire dont Quantité_à_acheter > 0 ;
- Quantité_à_acheter = 0 si la Quantité_nécessaire_d ≥ Quantité_stock_d
=((Quantité_stock_d - Quantité_nécessaire_d)-
/Quantité_conditionnement_d'utilisation_dans_conditionnement_de_vente) sinon.

2.2.2. DETERMINATION DU MONTANT TOTAL DES ACHATS.

Objectif : calculer une approximation du coût total des achats à effectuer pour une certaine période.

Message-donnée : Liste_produits_à_acheter_d.

Propriété : Liste_produits_à_acheter_d à été généré par la fonction "Détermination des produits à acheter pour une période".

Consultation MSI: Produit.

Messages-résultats : - Liste_prix_produits_à_acheter_r <Nom_produit_r, Prix_unitaire_produit_r, Prix_global_produit>; la liste des produits à acheter pour la période et le prix de ceux-ci;
- Montant_total_r; le montant nécessaire pour faire les achats.

Action MSI: /.

Règles : - tout Nom_produit_r correspond à un et un seul Nom_produit_d appartenant à la Liste_produits_à_acheter_d;
- tout Prix_unitaire_produit_r correspond à l'attribut Prix_produit du produit identifié par Nom_produit_r;
- tout Prix_global_produit_r correspond au Prix_unitaire_produit_r * Quantité_à_acheter_d du produit identifié par Nom_produit_r ;
- il y a autant d'éléments dans Liste_prix_produits_à_acheter_r que dans Liste_produits_à_acheter_d;
- Montant_total_r est la somme de tous les Prix_global_produit_r.

2.2.3. AFFICHAGE DE LA SOMME A EMPORTER.

Objectif : Afficher une somme d'argent à emporter pour réaliser les achats à effectuer pour une période donnée en fonction du contenu du portefeuille.

Messages-données : - Montant_total_d;
- Contenu du portefeuille.

Propriétés : - Montant_total_d à été généré par la fonction "Détermination du montant total des achats";
- Le contenu du portefeuille est valide et est l'énumération en termes de billets et de pièces du contenu du portefeuille (budget alimentation) de l'utilisateur.

Consultation MSI : /.

Messages-résultats: - Pas_Assez_Argent;
ou - Assez_Argent.

Action MSI : /.

Règle : si Montant_total_d > somme contenu dans le portefeuille
alors Pas_Assez_Argent
sinon arrondir Montant_total_d
si Montant_total_d arrondi ≥ contenu du portefeuille
alors afficher tout le portefeuille
sinon afficher la somme à emporter en tenant compte du contenu du portefeuille.

2.2.4. VERIFICATION DU CONTENU DU PORTEFEUILLE.

Objectif : vérifier qu'un montant introduit par l'utilisateur comme faisant partie du contenu de son portefeuille, correspond bien à un billet ou à une pièce défini dans les paramètres monétaires.

Messages-données : - Montant_d;
- Paramètres_monétaires_d.

Propriété : - Paramètres_monétaires_d est l'énumération des pièces et des billets en circulation dans la pays et est valide.

Consultation MS I: /.

Messages-résultats : - Montant_Valide;
ou - Montant_Invalide.

Règle : si Montant_d est un billet ou une pièce définie dans Paramètres_monétaires_d
alors Montant_Valide
sinon Montant_Invalide.

2.3. phase SORTIE LISTE D'ACHATS.

2.3.1. IMPRESSION LISTE.

Objectif : imprimer la liste des produits à acheter pour une période.

Messages-données : - Liste_produits_à_acheter_d;
- Liste_prix_produits_à_acheter_d;
- Montant_total_d;

Propriétés : - Liste_produits_à_acheter_d à été généré par la fonction "Détermination des produits à acheter pour une période";
- Liste_prix_produit_à_acheter à été généré par la fonction "Détermination du montant total des achats";
- Montant_total_d à été généré par la fonction "Détermination du montant total des achats".

Consultation MSI: Repas, Menu, Plat, Préparation, Constitution.

Message-résultat : /.

Actions MSI : - mise à jour de l'attribut Modifie_menu d'occurrences de l'entité Menu;
- mise à jour de l'attribut Modifie_plat d'occurrences de l'association Consultation.

Règles : - pour tous les produits de Liste_produits_à_acheter_d :
- imprimer Nom_produit_d, quantité_à_acheter_d, prix_unitaire_d, prix_global_d;
- imprimer le montant total à emmener;
- mise à l'état "FAUX" de l'attribut Modifie_menu de toutes les occurrences de Menu dont la date_préparation est comprise dans la période pour laquelle on imprime une liste d'achats;
- mise à l'état "INCHANGE" de l'attribut Modifie_plat_constitution de toutes les occurrences de Constitution auxquelles participent les occurrences de Menu mentionnées ci-dessus.

3 GLOSSAIRE.

remarque : un certain nombre de définitions ont déjà été données lors de la définition du schéma conceptuel des données, s'y référer.

- **S.I.** : système d'information.

- **consultation MSI** : consultation du système d'information.

- **action MSI** : action sur le système d'information.

- **constituer un menu** pour un repas d'un jour : définir le nombre de personnes pour ce menu, définir les thèmes constituant ce menu et les plats constituant ce menu pour ces thèmes. On peut très bien constituer un menu vide, c'est-à-dire que l'on associe un menu à une date et à un repas, mais l'ensemble des plats et des thèmes constituant ce menu est vide.

- **modifier un menu** : modifier le nombre de personnes de ce menu et/ou ajouter un ou plusieurs thèmes et un ou plusieurs plats à ce menu et/ou supprimer un ou plusieurs thèmes et un ou plusieurs plats de ce menu.

- **menu vide** : menu qui n'est constitué d'aucun plat. A telle date, pour tel repas, il n'y a pas de plat de prévu.

- **période d'achat** : suite de jours pour lesquels l'utilisateur veut établir une liste d'achats.

- **produit de base facultatif** : produit de base qui n'entre dans la composition d'aucun plat et que l'on peut acheter de temps à autre sans pour autant en avoir toujours sous la main.

- **produit de base facultatif demandé** pour une période d'achat : produit de base facultatif que l'utilisateur désire acheter au cours de cette période, l'utilisateur demande à acheter ce produit.

- **produit de base obligatoire** : produit de base qui entre ou non dans la composition de plats et qu'il faut toujours avoir chez soi.

- **produit de base nécessaire** pour une période d'achat : produit de base obligatoire ou facultatif demandé pour cette période.

- **produit nécessaire** pour une période d'achat : produit de base nécessaire pour cette période d'achat et/ou produit qui entre dans la composition d'au moins un plat constituant d'au moins un menu prévu pour un jour de cette période.

- **quantité nécessaire d'un produit pour une période** en fonction du scénario de création de liste d'achats : quantité nécessaire d'un produit pour pouvoir satisfaire à la demande de ce produit pour la préparation de plats sélectionnés dans des menus pour une certaine période et/ou à la demande de ce produit en tant que produit de base nécessaire pour cette même période; elle se distingue de la **quantité à acheter**, cette dernière tenant compte de la quantité du produit qu'il reste dans les réserves de l'utilisateur.

- **plat constituant d'un menu pour un thème** : plat qui fait partie de ce menu pour ce thème.

- **thème constituant d'un menu** : thème qui fait partie de ce menu; à ce thème, dans ce menu, est associé un plat (qui est aussi dit constituant du menu).

- **appartenance d'un plat à un thème** ou **plat possible pour un thème** : plat qui peut être constituant d'un menu pour ce thème.

- **thème élémentaire** : thème qui ne peut pas être décomposé en d'autres thèmes.

- **plat élémentaire** : plat qui ne peut pas être décomposé en d'autres plats.

- **repas non traité** à une certaine date : repas pour lequel, à cette date, aucun menu n'a été constitué.

- **repas traité** à une certaine date : repas pour lequel, à cette date, un menu a été constitué.

- **traiter un repas** : constituer un menu pour ce repas.

- **jour traité** : jour pour lequel il y a, pour chaque repas connu du S.I., un menu de constitué.

- **jour non traité** : jour pour lequel il y a au moins un repas pour lequel il n'y a pas de menu de constitué.

- **nombre de personnes cohérent** : un nombre de personnes prévu pour un menu est cohérent vis-à-vis du nombre de plats constituants de ce menu si:
    - si le menu est vide, le nombre de personnes est égal à 0;
    - si le menu est constitué d'au moins un plat, le nombre de personnes est un entier strictement positif.

- **dates de début et de fin de liste** et **dates de début et de fin de période d'une liste** : c'est la même chose; une liste d'achats concernant une période d'achat, on parlera indifféremment des dates de début et de fin de la liste ou des dates de début et de fin de période concernée par la liste.

- **unité d'achat** d'un produit : si le produit est vendu à la quantité (au poids, au volume, ...) alors l'unité d'achat est l'unité de la mesure dans laquelle on mesure les quantités consommées du produit ou utilisées dans la préparation de plats, sinon, l'unité d'achat est le conditionnement de vente.

- **plus petite unité d'emballage utilisable** d'un produit : si, à l'intérieur d'un conditionnement dans lequel le produit est vendu, le produit est emballé dans un autre conditionnement, alors la plus petite unité d'emballage utilisable de ce produit est ce dernier conditionnement, sinon, suivant les cas, ce sera le produit lui-même ou le conditionnement de vente. Exemples : 1) la purée en flocons est vendue dans des boîtes et les boîtes contiennent des sachets, le sachet est la plus petite unité d'emballage utilisable de ce produit;
  2) les pommes-de-terre sont vendues par sac et à l'intérieur du sac ne sont emballées dans rien d'autre, la pomme-de-terre est la plus petite unité d'emballage utilisable de ce produit;
  3) le beurre est vendu dans des paquets, le paquet est la plus petite unité d'emballage utilisable de ce produit.

- **scénario de création de liste d'achats** : il y a deux scénarios de création de liste d'achats. Dans le premier, on crée une liste d'achats qui reprend les produits et les quantités à acheter de ces produits nécessaires pour pouvoir préparer les menus constitués pour une certaine période. Dans le second, on crée une liste d'achats qui reprend les produits et les quantités à acheter de ces produits nécessaires pour pouvoir préparer les menus constitués pour une certaine période et pour pouvoir satisfaire à la demande de produits de base pour cette période. Chacun des deux est attaché à un scénario de création de menus.

- **scénario de création de menus** : il y a deux scénarios de création de menus. Dans le premier, l'utilisateur doit constituer un menu pour chaque repas de chaque jour, il est obligé de traiter chaque repas de chaque jour, et dans le second, il n'est plus obligé de traiter chaque repas, il peut constituer de zéro à un menu par jour.

Les fichiers.

Dans cette partie, nous présentons les fichiers utilisés par le logiciel. Toute la démarche qui nous a permis de passer du schéma Entité-Association que nous avons vu précédemment, à ces fichiers, ne sera pas expliquée ici. Signalons simplement que pour ce travail, notre référence a été [Hainaut 86].

Pour chacun des fichiers, nous donnons la définition de ses éléments en langage C, et pour chaque champs des éléments, sa signification. Chaque définition est complétée par les contraintes sur les fichiers, à l'exception, pour ne pas allourdir le texte, de celles déjà définies sur les données lors de la spécification du schéma Entité-Association (notamment les identifiants) , et qui restent bien sûr d'application ici.

- 1 : Fichier **REPAS.**

Ce fichier contient les repas connus du S.I.

- a : définition C.

```
typedef struct {
                short          no_repas;
                tnom_repas     nom;
                short          article;
                tson_repas     son_nom;
                short          priorité;
               } trepasint;

typedef char tnom_repas[16];
typedef char tson_repas[26];
```

- b : signification.

no_repas : le numéro du repas.
nom : le nom du repas.
article : le numéro de l'article du repas.
son_nom : le son du nom[1].
priorité : la priorité du repas.

[1] Le son du nom n'est pas le nom lui-même, du fait de l'emploi d'un synthétiseur anglais qui nous a obligé à découper les noms en phonèmes, afin d'obtenir une prononciation aussi claire que possible pour un francophone.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_repas.
- pour la correspondance du numéro de l'article, voir en fin de cette annexe.

- 2 : Fichier **MENU.**

Ce fichier contient une partie des renseignements sur les menus connus du S.I.; il est complété par le fichier PTR_MENU_CONST.

- a : définition C.

```
typedef struct {
                short       nnbr_pers;
                short       nbr_pers;
                booleanint  modifie;
                booleanint  imprime;
               } tmenuint;

typedef short booleanint[2];
```

- b : signification.

nnbr_pers : le nombre de personnes prévues actuellement pour le menu.
nbr_pers : le nombre de personnes prévues pour le menu au moment de l'impression de la dernière liste d'achats, ou à la création du menu si ce dernier a été créé après l'impression de la dernière liste d'achats.
modifie : signale si le menu a été modifié depuis la dernière impression de liste d'achats.
imprime : signale si le menu a déjà été imprimé.

-c : contraintes.

- nnbr_pers et nbr_pers <= 6

[2] Un booleanint est un boolean qui vaut 0 pour FAUX et 1 pout VRAI.

- 3 : Fichier **CONSTITUTION.**

Ce fichier contient une partie des renseignements sur les constitutions de menu connues du S.I.; il est complété par le fichier PTR_MENU_CONST.

- a : définition C.

```
typedef struct {
                short  no_thème;
                short  no_plat;
                char   modifie_plat;
                short  ptr_suivant;
               } tconstitutionint;
```

- b : signification.

no_thème : le numéro du thème.
no_plat : le numéro du plat.
modifie_plat : indique l'état de modification de la constitution.
ptr_suivant : position, dans le même fichier de la constitution suivante du menu dont cette constitution fait partie.

- c : contraintes.

- pour tout no_plat N de CONSTITUTION, N appartient à {no_plat} de PLAT.
- pour tout no_thème N de CONSTITUTION, N appartient à {no_thème} de THEME.
- modifie_plat vaut : 'A' si le plat no_plat a été ajouté (ou substitué à un autre plat pour le même thème) au menu, dont cette constitution fait partie, depuis la dernière impression de liste d'achats concernant ce menu;
  : 'E' si le plat a été retiré du menu (et aucun autre ne lui a été substitué pour le thème) depuis la dernière impression de liste d'achats;
  : 'I' sinon.
- ptr_suivant vaut 9999 si cette constitution est la denière du menu dont elle fait partie.

- 4 : Fichier **PTR_MENU_CONST.**

Ce fichier contient les pointeurs vers les MENU et les CONSTITUTION.

- a : définition C.

```
typedef struct {
                tdate  date;
                short  repas;
                short  ptr_menu;
                short  ptr_const;
               } tptr_menu_const;

typedef struct {
                short  annee;
                short  mois;
                short  jour;
               } tdate;
```

- b : signification.

date : la date de la préparation.
no_repas : le numéro du repas.
ptr_menu : la position, dans le fichier MENU, de l'article identifié par (date, repas).
ptr_const : la position, dans le fichier CONSTITUTION, de la première constitution du menu identifié par (date, repas).

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de date d'abord, de repas ensuite.
- pour tout repas R de CONSTITUTION, R appartient à {no_repas} de REPAS.
- toute date de CONSTITUTION est non antérieure à la date du jour courant.

- 5 : Fichier **THEME.**

Ce fichier contient les thèmes connus du S.I.

- a : définition C.

```
typedef struct {
                short          no_theme;
                tnom_theme     nom;
                short          article;
                tson_theme     son_nom;
                short          priorite;
                UBYTE          couleur;
                short          no_theme_pere;
                booleanint     a_fils;
               } tthemeint;

typedef char tnom_plat[21];
typedef char tson_plat[31];
```

- b : signification.

no_theme : le numero du thème.
nom : le nom du thème.
article : le numéro de l'article du thème.
son_nom : le son du nom.
priorité : la priorité du thème.
couleur : le numéro de la couleur associée au thème.
no_theme_pere : le numéro du thème père de ce thème, si ce dernier est obtenu par décomposition d'un autre.
a_fils : indique si le thème est décomposable.

- c : contraintes.

- si a_fils vaut 1, alors no_theme_pere vaut 9999.
- les éléments du fichier sont classés par ordre croissant de no_theme.
- pour la correspondance du numéro de l'article, voir en fin de cette annexe.

- 6 : Fichier **PLAT.**

Ce fichier contient les plats connus du S.I.

- a : définition C.

```
typedef struct {
                short       no_plat;
                tnom_plat   nom;
                short       article;
                tson_plat   son_nom;
                short       cout;
               } tplatint;

typedef char tnom_plat[26];
typedef char tson_plat[36];
```

- b : signification.

no_plat : le numéro du plat.
tnom_plat : le nom du plat.
article : le numéro de l'article du plat.
son_nom : le son du nom.
cout : indique la cherté du plat.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_plat.
- cout vaut : 3 pour CHER;
  : 2 pour NORMAL;
  : 1 pour BON MARCHE.
- pour la correspondance du numéro de l'article, voir en fin de cette annexe.

- 7 : Fichier **PTR_LIENS_PLATS.**

Ce fichier contient les pointeurs vers les plats fils de plats décomposables.

- a : définition C.

```
typedef struct {
                short  no_plat;
                short  ptr_liensplats;
                short  nbr_fils;
               } tptr_liens_plats;
```

- b : signification.

no_plat : le numéro du plat.
ptr_liensplats : la position, dans le fichier LIENS_PLATS, du premier plat fils du plat.
nbr_fils : le nombre de plats fils du plat. Ces fils sont tous placés dans le fichier LIENS_PLATS et sont contigüs au premier fils, pointé par ptr_liensplats.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_plat.
- pour tout no_plat N de PTR_LIENS_PLATS, N appartient à {no_plat}.

- 8 : Fichier **LIENS PLATS.**

Ce fichier contient les plats fils en fonction de leur père. L'accès à ce fichier se fait via le fichier PTR_LIENS_PLATS; se référer d'abord à la description de ce dernier.

- a : définition C.

```
typedef struct {
                short  no_plat_fils;
               } tliensplats;
```

- b : signification.

no_plat_fils : le numéro d'un plat.

- c : contraintes.

- pour tout no_plat_fils N de LIENS_PLATS, N appartient à {no_plat} de PLAT.

- 9 : Fichier **PTR_APPARTENANCE.**

Ce fichier contient les pointeurs vers les plats en fonction de leur appartenance aux thèmes.

- a : définition C.

```
typedef struct {
                short  no_theme;
                short  ptr_app;
                short  nbr_plats;
               } tptr_appartenance;
```

- b : signification.

no_theme : le numéro du thème.
ptr_app : la position, dans le fichier APPARTENANCE, du premier plat appartenant au thème.
nbr_plats : le nombre de plats appartenant au thème; ces plats sont placés de façon contigüe dans le fichier APPARTENANCE; après le premier du thème, pointé par ptr_app.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_thème.
- pour tout no_theme N de PTR_APPARTENANCE, N appartient à {no_theme} de THEME.

- 10 : Fichier **APPARTENANCE.**

Ce fichier contient les plats en fonction de leur appartenance à un thème. L'accès à ce fichier se fait via le fichier PTR_APPARTENANCE; se référer d'abord à la description de ce dernier.

- a : définition C.

```
typedef struct {
                short  no_plat;
               } tappartenanceint;
```

- b : signification.

no_plat : le numéro d'un plat.

- c : contraintes.

- pour tout no_plat N de APPARTENANCE, N appartient à {no_plat} de PLAT.

- 11 : Fichier **PRODUIT.**

Ce fichier contient les produits connus du S.I.

- a : définition C.

```
typedef struct {
                short           no_produit;
                tnom_produit    nom;
                short           article;
                short           article2;
                tson_produit    son_nom;
                short           classe_stock;
                short           prix;
                short           cond_vente;
                short           cond_util;
                float           qt_cond_util_cond_vente;
               } tproduitint;

typedef char tnomproduit[31];
typedef char tsonproduit[41];
```

- b : signification.

no_produit : le numéro du produit.
nom : le nom du produit.
article : le numéro de l'article du produit.
article2 : le numéro de l'article faisant la liaison entre le produit et ses conditionnements d'utilisation et de vente.
son_nom : le son du nom.
classe_stock : le numéro de la classe de gestion de stock du produit.
prix : le prix d'une unité de conditionnement de vente du produit, en francs belges.
cond_vente : le numéro du conditionnement de vente du produit.
cond_util : le numéro du conditionnement d'utilisation du produit.
qt_cond_util_cond_vente : le nombre d'unités de conditionnement d'utilisation du produit dans une unité de conditionnement de vente du produit.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_produit.
- classe_stock vaut : 1 si la gestion du stock du produit est de type APPROXIMATION;
  : 0 sinon.
- si cond_util=cond_vente=nom, alors qt_cond_util_cond_vente = 0.
- les numéros de conditionnement d'utilisation et de vente correspondent à :

0	<le nom du produit>
1	paquet
2	pot
3	boîte
4	sachet
5	sac
6	gousse
7	bouteille
8	botte
9	barquette

- pour la correspondance du numéro de l'article et de l'article2, voir en fin de cette annexe.

- 12 : Fichier **PTR_COMPOS_PLAT.**

Ce fichier contient les pointeurs vers la composition des plats.

- a : définition C.

```
typedef struct {
                short  no_plat;
                short  ptr_compos_plat;
                short  nbr_produits;
               } tptr_compos_plat;
```

- b : signification.

no_plat : le numéro du plat.
ptr_compos_plat : la position, dans le fichier COMPOS_PLAT, du premier produit entrant dans la composition du plat.
nbr_produits : le nombre de produits entrant dans la composition du plat; ces produits sont placés, dans le fichier COMPOS_PLAT, de façon contigüe, après le premier, pointé par ptr_compos_plat.

- c : contraintes.

- les éléments du fichier sont classés par ordre croissant de no_plat.
- pour tout no_plat N de PTR_COMPOS_PLAT, N appartient à {no_plat) de PLAT.

- 13 : Fichier **COMPOS_PLAT.**

Ce fichier contient les produits en fonction de leur participation à la composition de plats. L'accès à ce fichier se fait via le fichier PTR_COMPOS_PLAT; se référer d'abord à la description de ce dernier.

- a : définition C.

```
typedef struct {
                short no_produit;
                float qte_produit;
               } tcompos_platint;
```

- b : signification.

no_produit : le numéro du produit entrant dans la composition du plat.
qte_produit : la quantité du produit entrant dans la composition du plat, exprimée en quantité (non nécessairement entière) de conditionnement d'utilisation du produit.

- c : contraintes.

- pour tout no_produit N de COMPOS_PLAT, N appartient à {no_produit} de PRODUIT.
- pour tout no_produit identifiant un PRODUIT P tel que classe_stock de P = 1, qte_produit de P de COMPOS_PLAT=0.

- 14 : Fichiers **.PIC et .BLISS.**

Pour chaque dessin, ou représentation Bliss, d'un plat ou d'un produit, un fichier est créé dont le nom est, selon les cas :

PLAT<no_plat>.PIC
ou PRODUIT<no_produit>.PIC
ou PRODUIT<no_produit>.BLISS

Remarque :

Les numéros d'article correpondent à :

0	<rien>	6	une
1	le	7	des
2	la	8	de la
3	les	9	du
4	l'	10	de
5	un	11	d'

La modélisation du dialogue : quelques exemples.

1 GESTION D'UN BOUTON

INTERACTION_OBJECT Bouton **is**

VARS:

position	/* position du bouton */
label	/* nom du bouton */
fichier	/* fichier contenant l'image du bouton */
estSélectionné	/* dit si oui ou non le bouton est sélectionné */
estModifié := False	/* dit si oui ou non le bouton à été modifié */

METHODS:

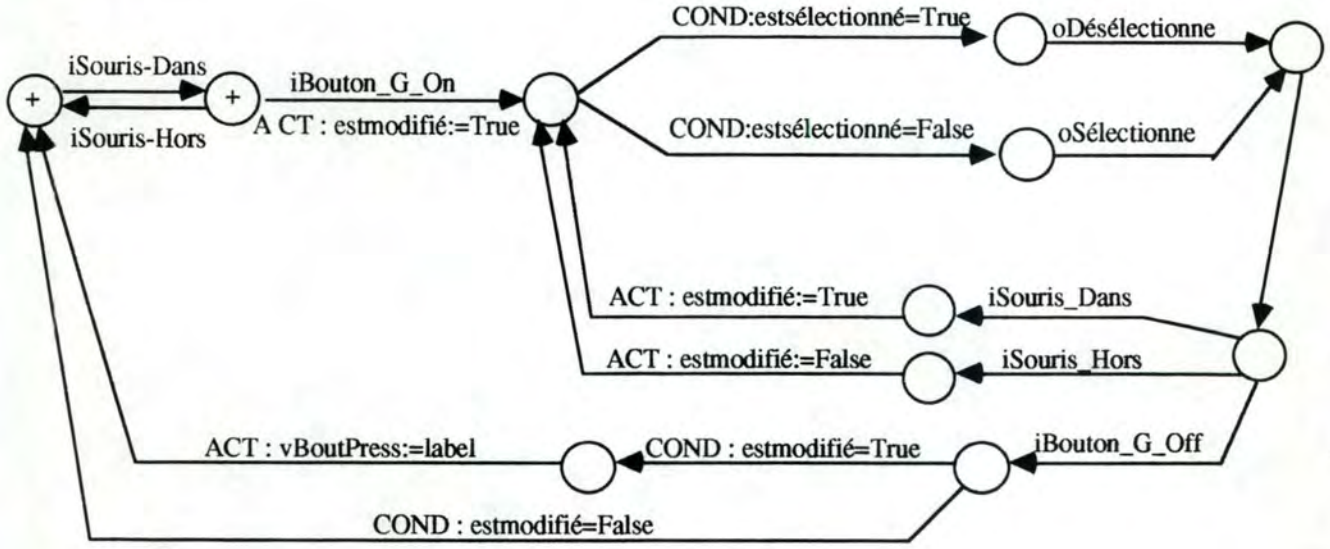
crée()	{CréeInstance(position, label, fichier, estSélectionné)};
détruit()	{Détruit Instance(label)};
affiche()	{AfficheBouton(label)};
efface()	{EffaceBouton(label)};
déselectionne()	{DésélectBouton(label)};

TOKENS:

iSouris-Dans	{le curseur est dans la zone délimitée par position}
iSouris-Hors	{le curseur est hors de la zone délimitée par position}
iBouton-G-On	{le bouton gauche de la souris est enfoncé}
iBouton-G-Off	{le bouton gauche de la souris est relâché}
oSélectionne	{**if not** estSélectionné **then** estSélectionné := True; MiseEvidence(position); **end if**}
oDésélectionne	{**if** estSélectionné **then** estSélectionné := False; MiseNormal(position); **end if**}

SYNTAX:

Gestion Bouton



end INTERACTION-OBJECT

2 GESTION DE L'EXCLUSIVITE DE DEUX BOUTONS

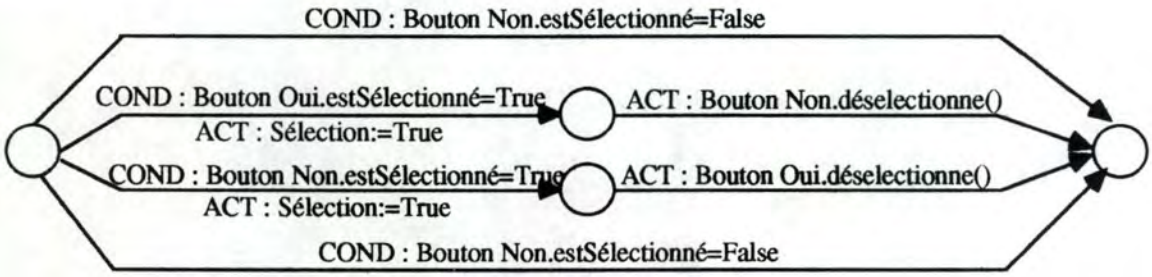
INTERACTION_OBJECT BoutonsExclusifs **is**

IVARS:

Sélection := False /* est vrai si un des deux boutons est sélectionné */

SUBS:

Gestion-Exclusivité



end INTERACTION_OBJECT

3 GESTION D'UN DEROULEUR

INTERACTION_OBJECT Dérouleur **is**

IVARS:

```
label                /* nom identifiant le dérouleur                    */
liste-val            /* liste des valeurs apparaissant dans le dérouleur */
Bouton Flèche-Haut   crée INTERACTION_OBJECT Bouton
                        (position=>{6,32,166,42},label=>'Flèche-Haut',
                        fichier=>'Flèche-Haut.lbm',estSélectionné=>False);
Bouton Flèche-Bas    crée INTERACTION_OBJECT Bouton
                        (position=>{6,190,166,200},label=>'Flèche-Bas',
                        fichier=>'Flèche-Bas.lbm',estSélectionné=>False);
Bouton Sélection-1l  crée INTERACTION_OBJECT Bouton
                        (position=>{2,?[1] ,166,?+15},label=>'Sélection',
                        fichier=>'Sélection.lbm',estSélectionné=>False);
Bouton Sélection-3l  crée INTERACTION_OBJECT Bouton
                        (position=>{2,?,166,?+45},label=>'Sélection',
                        fichier=>'Selection.lbm',estSélectionné=>False);
BoutonPressé            :='';
```

METHODS:

```
crée()          {créeInstance(label, bouton Flèche-Haut,
                bouton Flèche-Bas, bouton Sélection-1l,
                bouton Sélection-3l)};
détruit()       {détruitInstance(label)};
affiche()       {AfficheDérouleur()};
```

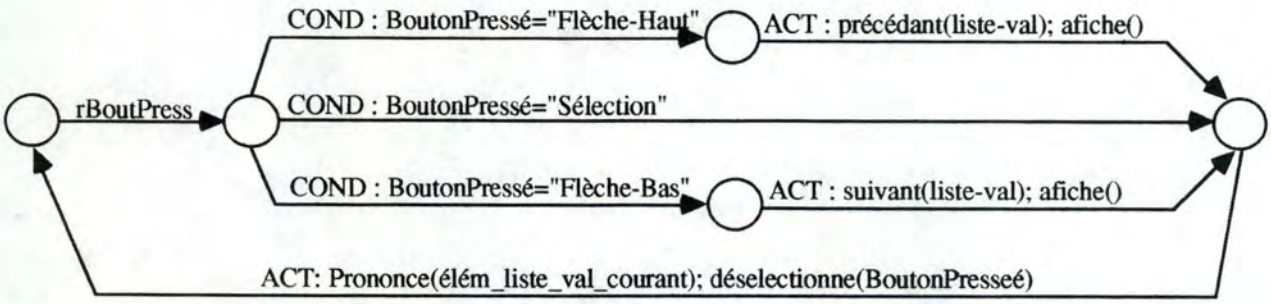
TOKENS:

```
rBoutPress      {BoutonPressé := vBoutPress};
```

[1] Les ? correspondent à des positions qui seront calculées par les procédures SUIVANT et PRECEDENT

SYNTAX:

Gestion Dérouleur



end INTERACTION_OBJECT

4 GESTION MESSAGE AVEC CONFIRMATION OU INFIRMATION

INTERACTION_OBJECT ConfirmationOui-Non **is**

FROM:

BoutonsExclusifs /*héritage des éléments de l'objet BoutonsExclusifs */

IVARS:

position /* position à laquelle apparaît la fenêtre de confirm. */
label /* nom identifiant la fenêtre de confirmation */
position_oui /* position du bouton Oui dans la fenêtre */
position_non /* position du bouton Non dans la fenêtre */
phrase Confirm /* phrase de confirm. apparaissant dans la fenêtre */
position_phrase /* position à laquelle apparaît la phrase */
Bouton Oui := **crée INTERACTION_OBJECT** Bouton
(position=>position_oui, label=>'Oui',
fichier=>'Oui.lbm', estSélectionné=>False);
Bouton Non := **crée INTERACTION_OBJECT** Bouton
(position=>position_non, label=>'Non',
fichier=>'Non.lbm', estSélectionné=>False);
BoutonPressé :='';

METHODS:

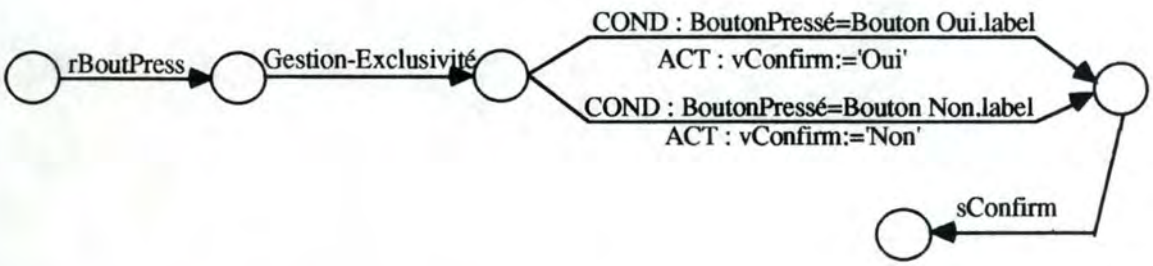
crée()	{CréeInstance(label, position, position_phrase, bouton Oui, bouton Non, phrase Confirm)};
détruit()	{Détruit Instance(label)};
affiche()	{AfficheConfirm(position)};

TOKENS:

rBoutPress	{BoutonPressé=vBoutPress}

SYNTAX:

Gestion ConfirmOuiNon



end INTERACTION_OBJECT

5 GESTION MESSAGE AVEC CONFIRMATION DE LA PRISE DE CONNAISSANCE

INTERACTION_OBJECT ConfirmationOK **is**

IVARS:

position	/* position à laquelle apparaît la fenêtre de confirm. */
label	/* nom identifiant la fenêtre de confirmation */
position_OK	/* position du bouton OKdans la fenêtre */
phrase Confirm	/* phrase de confirm. apparaissant dans la fenêtre */
position_phrase	/* position à laquelle apparaît la phrase */
Bouton OK	:= **crée INTERACTION_OBJECT** Bouton (position=>position_OK, label=>'OK', fichier=>'OK.lbm', estSélectionné=>False);
BoutonPressé	:='';

METHODS:

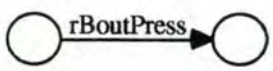
crée()	{CréeInstance(label, position, position_phrase, bouton OK, phrase Confirm)};
détruit()	{Détruit Instance(label)};
affiche()	{AfficheConfirm(position)};

TOKENS:

rBoutPress	{BoutonPressé=vBoutPress}

SYNTAX:

Gestion ConfirmOK



end INTERACTION_OBJECT

6 GESTION DE L'ECRAN DES THEMES (ecran4 dans le mémoire)

INTERACTION_OBJECT EcranThemes **is**
IVARS:

label	:= 'écran choix';
liste_thèmes	/* liste des thèmes apparaissant dans le dérouleur */
Dérouleur-Thèmes	**crée INTERACTION_OBJECT** Dérouleur (label=>'Dérouleur-Thèmes', liste-val=>liste-thèmes);
Bouton Fin	**crée INTERACTION_OBJECT** Bouton (position=>{249,210,290,251},label=>'Fin', fichier=>'stop.lbm',estSélectionné=>False);
Bouton But	**crée INTERACTION_OBJECT** Bouton (position=>{30,210,71,251},label=>'but', fichier=>'message.lbm',estSélectionné=>False);
Bouton Ok	**crée INTERACTION_OBJECT** Bouton (position=>{140,210,181,251},label=>'Ok', fichier=>'Ok.lbm',estSélectionné=>False);

```
ConfirmFinThèmes   crée INTERACTION_OBJECT ConfirmationOui-Non
                     (position=>{10,300,88,400},label=>'ConfirmFinThèmes',
                     position_oui=>{59,50,100,91},
                     position_non=>{200,50,241,91},
                     phrase Confirm=>'As  tu fini le travail  sur les thèmes ?',
                     position_phrase=>{15,25});
ConfirmPlusPlats   crée INTERACTION_OBJECT ConfirmationOK
                     (position=>{10,300,88,400},
                     label=>'ConfirmPlusPlats,
                     position_oui=>{180,50,221,91},
                     phrase Confirm=>'Il  n'y a plus de thèmes',
                     position_phrase=>{15,25});
BoutonPressé       := '';
BoutonExclusif     := '';
Confirm            := '';
```

METHODS:

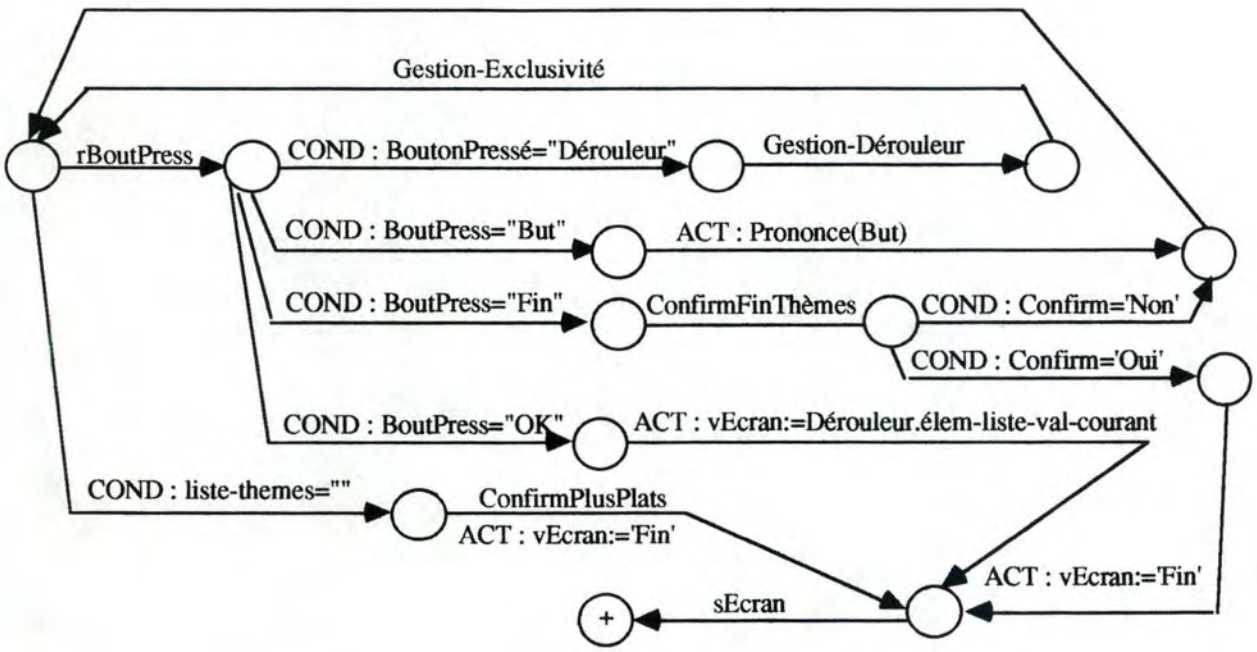
```
crée()             {créeInstance(label, Dérouleur-Thèmes,
                   bouton Fin, bouton But, bouton Ok, ConfirmFinThèmes,
                   ConfirmPasPlats, ConfirmPlusPlats)};
détruit()          {détruitInstance(label)};
affiche()          {AfficheEcran()};
efface()           {EffaceEcran()};
```

TOKENS:

```
rBoutPress         {BoutonPressé := vBoutPress};
rConfirm           {Confirm := vConfirm};
```

SYNTAX:

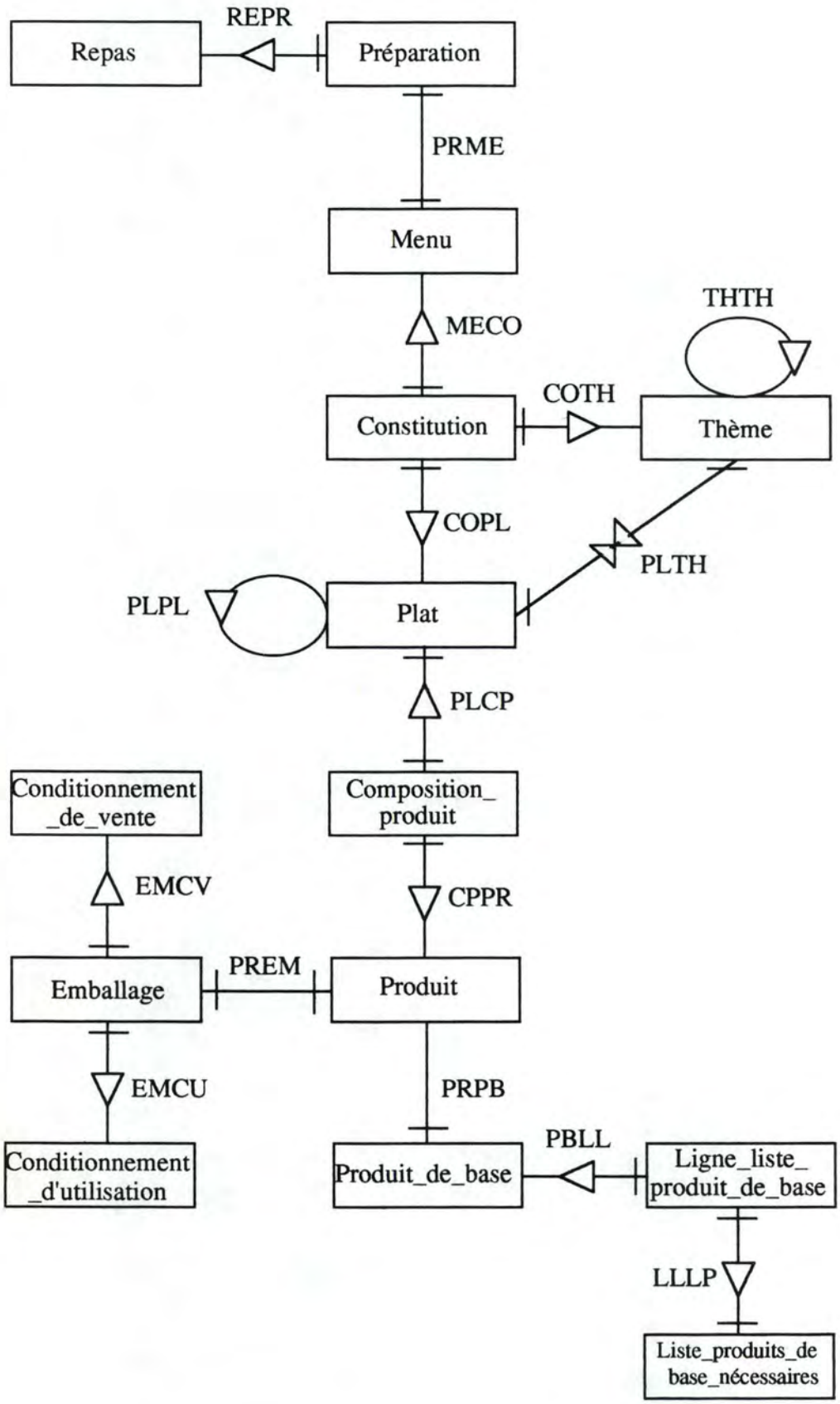
Gestion Ecran Thèmes



end INTERACTION_OBJECT

Schéma des accès possibles.

Ce schéma des accès possibles, [Hainaut 86], simplifié, nous sera nécessaire, et suffisant, pour la spécification de certains modules physiques.



Définition des modules physiques.

```
/*
 * Module   : Filtrage collection
 * Type     : Spécifications externes
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_types2.h"
#include "H_Privacy.h"

/*********************************/
/*********************************/
/*  MODULE FILTRAGE COLLECTION  */
/*********************************/
/*********************************/

PUBLIC void fournir_preparations(tpreparation Tab_preparations[],
                                 short *Nombre);

/*
BUT : fournir les articles PREPARATION connus du S.I.

IN : \.

OUT : -Tab_preparations.
      -Nombre.

POST: -Tab_preparation est un pointeur vers le tableau qui contient la
       valeur des items des articles PREPARATION connus du S.I.
      -Nombre est un pointeur vers la taille réelle de ce tableau.
*****************************************************************************/

PUBLIC void fournir_repas(trepas Tab_repas[],short *Nombre);

/*
BUT : fournir les articles REPAS connus du S.I.

IN : \.

OUT : -Tab_repas.
      -Nombre.

POST : -Tab_repas est un pointeur vers le tableau qui contient la valeur
        des items des articles REPAS connus du S.I.
       -Nombre est un pointeur vers la taille de ce tableau.
*****************************************************************************/

PUBLIC void fournir_themes_et_fils(ttheme_pere_fils Tab_themes[],
                                   short *Nombre);

/*
BUT : fournir les articles THEME, non-décomposables,décomposables ou fils,
      connus du S.I.

IN : \.

OUT : -Tab_themes.
      -Nombre.
```

```
POST : -Tab_themes est un pointeur vers le tableau contenant la valeur des
        items des THEME connus du S.I. Ce tableau est organisé de la
        manière suivante : un THEME non-décomposable est contenu dans une
        cellule du tableau et la valeur du champs nbr_fils vaut 0, un THEME
        décomposable est contenu dans une cellule du tableau, la valeur
        du champs nbr_fils vaut le nombre de THEME fils de ce THEME, et
        ses fils sont contenus dans les Nbr_fils cellules suivantes du
        tableau.
       -Nombre est un pointeur vers la taille réelle de ce tableau.
*****************************************************************************/

PUBLIC void fournir_menu_complet(tdate Date,short Repas,short *Nbr_pers,
                                 tconstituants_ext Tab_const[],
                                 short *Nbr_const,
                                 tplat_pere_fils Tab_plats[],
                                 short *Nbr_plats,boolean *Existe);
/*
BUT : fournir les caractéristiques d'un menu complet.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -Existe.
      -Nbr_pers.
      -Tab_const.
      -Nbr_const.
      -Tab_plats.
      -Nbr_plats.

POST : -C1 : il existe un MENU identifié par (Date,Repas).
       -E1 : Existe pointe vers la valeur VRAI.
       -E2 : Existe pointe vers la valeur FAUX.
       -E3 : Nbr_pers, Tab_const, Nbr_const, Tab_plats, Nbr_plats, non
             définis.
       -E4 : Nbr_pers pointe vers le nombre de personnes du menu identifié
             par (Date,Repas).
       -E5 : Tab_const est un pointeur vers le tableau contenant la valeur
             des items des THEME constituants du menu identifié par
             (Date,Repas).  Chaque cellule du tableau contient un champs
             ptr_plat qui donne la position dans le tableau Tab_plats du
             PLAT correspondant à chaque THEME dans ce menu.
       -E6 : Nbr_const pointe vers la taille réelle du tableau Tab_const.
       -E7 : Tab_plats pointe vers le tableau contenant la valeur des items
             des PLAT décomposables, non-décomposables, ou fils, du MENU
             identifié par (Date,Repas). Chaque cellule de ce tableau
             contient un champs nbr_fils qui vaut 0 si le PLAT est
             non-décomposable et qui vaut le nombre de fils du PLAT
             s'il est décomposable; ses PLAT fils sont placés dans les
             Nbr_fils suivantes du tableau.
       -E8 : Nbr_plats est la taille réelle du tableau Tab_plats.

       -si C1 alors E1 et E3 à E8, sinon E2.
*****************************************************************************/
```

```
PUBLIC void fournir_plats_et_fils_sur_clel(short Theme,
                                           tplat_pere_fils Tab_plats[],
                                           short *Nbr_plats);

/*
BUT : fournir les articles PLAT appartenant à un THEME donné.

IN : -Theme, le numéro d'un THEME.

PRE : -Theme est le numéro d'un thème connu du S.I.

OUT : -Tab_plats.
      -Nbr_plats.

POST : -Tab_plats pointe vers le tableau contenant la valeur des items
        PLAT appartenenant au THEME identifié par Theme; le champs
        Nbr_fils des cellules de ce tableau vaut 0 si les PLAT sont
        non-décomposables, il vaut le nombre de fils des PLAT sinon,
        et les caractéristiques des PLAT fils sont placées dans les
        Nbr_fils cellules suivantes après celle du père.
       -Nbr_plats est la taille réelle de ce tableau.
*************************************************************************/

PUBLIC void fournir_menu_prep_repas(tmenupreprep Tab_menupreprep[],
                                    short *Nombre);

/*
BUT : fournir les articles MENU, PREPARATION et REPAS connus du S.I.

IN : \.

OUT : -Tab_menupreprep.
      -Nombre.

POST : -Tab_menupreprep pointe vers le tableau contenant la valeur des
        items des MENU, PREPARATION et REPAS connus du S.I., chaque cellule
        contenant une PREPARATION, un REPAS, et le MENU y relatif.
       -Nombre pointe vers la taille réelle de ce tableau.
*************************************************************************/

PUBLIC void fournir_produits_compos__sur_clel(short Plat,
                                              tproduitcompos Tab_produits[],
                                              short *Nombre);

/*
BUT : fournir les articles PRODUIT entrant dans la composition d'un PLAT,
      y compris la quantité du produit entrant dans la composition du
      PLAT.

IN : -Plat, le numéro d'un PLAT.

PRE : -Plat est le numéro d'un PLAT connu du S.I.

OUT : -Tab_produits.
      -Nombre.

POST : -Tab_produits pointe vers le tableau contenant la valeur des items
        des PRODUIT entrant dans la composition du PLAT identifié par Plat.
        A chaque item est associée la quantité du produit entrant dans la
```

```
        composition du plat, quantité exprimée en nombre d'unités de
        conditionnement d'utilisation du produit.
       -Nombre pointe vers la taille réelle de ce tableau.
*****************************************************************************/

PUBLIC void fournir_produits_menu(short Tab_plats,short Tab_plats_long,
                                  tproduitcompos Tab_produits[],
                                  short *Tab_produits_long);
/*
BUT : retourner les articles PRODUIT entrant dans la composition d'une
      suite de PLAT.

IN : -Tab_plats, pointeur vers le tableau contenant les numéros de PLAT.
     -Tab_plats_long, la longueur effective de ce tableau.

PRE : -les numéros de PLAT font référence à des PLAT connus du S.I.

OUT : -Tab_produits.
      -Tab_produits_long.

POST : -Tab_produits pointe vers le tableau contenant les articles PRODUIT
        entrant dans la composition des PLAT référencés par Tab_plats.
        Le champs qtecompos de chaque cellule a une valeur non définie.
       -Tab_produits_long ppointe vers la taille réelle de ce tableau.
*****************************************************************************/
```

```
/*
 * Module   : Courses
 * Type     : Spécifications externes
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"

PUBLIC tmenupreprep tab_menupreprep[max_eloignement*maxrepas];
PUBLIC short nbrmenupreprep;

/********************/
/********************/
/*  MODULE COURSES  */
/********************/
/********************/

PUBLIC void deter_prod_qte_nec_menu_glo(tperiode_achat Tab_ss_periode[],
                                        short Tab_ss_periode_long,
                                        tproduitcompos Tab_prod_nec[],
                                        short *Tab_prod_nec_long);
/*
BUT : déterminer les PRODUIT nécessaires pour une période d'achat.

IN : -Tab_ss_periode, pointe vers le tableau des sous-périodes d'achat.
     -Tab_ss_période_long est la taille réelle de ce tableau.

PRE : -les sous-périodes d'achat forment une partition de la période
       comprise entre Tab_ss_periode[0].debut et
       Tab_ss_periode[Tab_ss_periode_long].fin.
      -le champs achat_fait de chaque sous-période indique si les achats
       ont été faits pour la sous-période.

OUT : -Tab_prod_nec.
      -Tab_prod_nec_long.

POST : -Tab_prod_nec pointe vers le tableau des PRODUIT nécessaires pour
        la période délimitée Tab_ss_periode[0].debut et
        Tab_ss_periode[Tab_ss_periode_long].fin.
       -le champs qtecompos indique la quantité nécessaire de chaque PRODUIT
        pour la période, quantité exprimée en nombre de conditionnements
        d'utilisation du PRODUIT
       -Tab_prod_nec_long pointe vers la longueur réelle de ce tableau.
*****************************************************************************/
```

```
/*
 * Module   : Macromenu
 * Type     : Spécifications externes
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Privacy.h"
#include "H_types2.h"

/************************/
/************************/
/*  MODULE MACRO MENU  */
/************************/
/************************/

PUBLIC void enregistrer_menu_complet(tdate Date,short Repas,short Nbr_pers,
                                     tconstituants_enreg Constituants[],
                                     short Nbr_const);

/*
BUT : enregistrer un menu complet valide.

IN : -Date, la date du menu.
     -Repas, le numero du repas concerné par le menu.
     -Nbr_pers, le nombre de personnes du menu.
     -Constituants, pointeur vers le tableau des numéros de thèmes
      constituants du menu, et des numéros de plats associés à ces thèmes
      dans le menu.
     -Nbr_const, la taille réelle du tableau Constituants.

PRE : -Date est une date existante dans le calendrier.
      -Repas est le numéro d'un REPAS connu du S.I.
      -il n'y a pas d'article MENU existant connu du S.I qui soit identifié
       par (Date,Repas).
      -les numéros de plats du tableau Constituants sont des numéros de
       PLAT connus du S.I.
      -les numéros de thèmes du tableau Constituants sont des numéros de
       THEME connus du S.I.
      -pour tout thème de Constituants, le plat y associé dans ce tableau
       est un plat appartenant à ce thème.
      -chaque thème de Constituants est unique dans ce tableau.
      -chaque thème de Constituants est un thème non obtenu par décomposi-
       tion d'un autre thème dans ce tableau.
      -Date est la date d'un jour non antérieur au jour courant.
      -(Date - date du jour courant + 1) <= param.eloignement_max.
      -si on travaille dans le premier scénario de création de menus, Date
       est la date du premier jour traité non antérieur au jour courant et
       Repas est le numéro du premier REPAS non traité de ce jour.
      -les thèmes, dans le tableau Constituants, sont classés par ordre de
       priorité décroissante.

OUT : \.

EFFETS sur le S.I. :
     -ajout d'une occurrence de MENU identifiée par (Date,Repas), les
      valeurs des attributs associés à cette occurrence étant :
          -nnbr_pers : Nbr_pers,
          -nbr_pers : Nbr_pers,
```

```
          -modifie : VRAI,
          -imprime : FAUX.
     -ajout d'une occurrence de PREPARATION identifiée par (Date,Repas),
      la valeur de l'attribut date associée à cette occurrence étant Date.
     -ajout de Nbr_const occurrences de CONSTITUTION identifiées chacune
      par (Date,Repas,plat,theme), pour tout theme du tableau Constituants
      et le plat auquel est associé ce theme dans ce tableau; la valeur de
      l'attribut modifie_plat de ces occurrences étant 'A'.
*****************************************************************************/

PUBLIC void modifier_menu_complet(tdate Date,short Repas,short Nbr_pers,
                                  tconstituants_enreg Const_a_enlever[],
                                  short Nbr_a_enlever,
                                  tconstituants_enreg Const_a_ajouter[],
                                  short Nbr_a_ajouter);

/*
BUT : enregistrer les modifications valides apportées à un menu existant.

IN : -Date, la date du menu pour lequel on veut enregistrer des
      modifications.
     -Repas, le numéro du repas concerné par le menu pour lequel on veut
      enregistrer des modifications.
     -Nbr_pers, le nouveau nombre de personnes du menu modifié.
     -Const_a_enlever, pointeur vers le tableau des numéros de plats à
      retirer du menu initial et des des numéros de thèmes auxquels ces
      plats étaient associés dans ce menu.
     -Nbr_a_enlever, la taille réelle du tableau Const_a_enlever.
     -Const_a_ajouter, pointeur vers le tableau des numéros de plats à
      ajouter au menu initial et des numéros de thèmes auxquels ces plats
      sont associés dans le menu modifié.
     -Nbr_a_ajouter, la taille réelle du tableau Const_a_ajouter.

PRE : -(Date,Repas) identifie un MENU connu du S.I.
      -les numéros de thèmes du tableau Const_a_enlever sont des numéros de
       thèmes constituants du menu initial.
      -pour tout thème de Const_a_enlever, le plat associé à ce thème dans
       ce tableau est aussi associé à ce thème dans le menu initial.
      -les numéros de thèmes de Const_a_ajouter sont des numéros de THEME
       connus du S.I.
      -les numéros de plats de Const_a_ajouter sont des numéros de PLAT
       connus du S.I.
      -pour tout thème de Const_a_ajouter, le plat y associé est un plat
       possible pour ce thème.
      -chaque thème de Const_a_ajouter y est unique.
      -aucun thème de Const_a_ajouter n'est constituant du menu initial.
      -chaque thème de Const_a_enlever y est unique.
      -les thèmes, dans Const_a_enlever/a_ajouter, sont classés par ordre
       de priorité décroissante.
      -les thèmes dans Const_a_enlever/a_ajouter, sont des thèmes non
       obtenus par décomposition d'un autre thème de ce tableau.

OUT : \.

EFFETS sur le S.I. :
     -mise_ajour de la valeur d'attributs du MENU identifié par
      (Date,Repas) :
          -nnbr_pers devient Nbr_pers,
          -modifie devient VRAI.
```

```
      -ajout de Nbr_a_ajouter occurrences de CONSTITUTION identifiées
       chacune par (date,repas,plat,thème), pour tout plat de
       Const_a_ajouter et le thème auquel y est associé ce plat, la valeur
       de l'attribut modifie_plat de chacune de ces occurrences étant 'A'.
      -mise-à-jour de la valeur de l'attribut modifie_plat associé aux
       occurrences de CONSTITUTION identifiée par (Date,Repas,plat,thème),
       pour tout plat de const_a_enlever et le thème auquel y est associé
       ce plat, la valeur de cet attribut, pour ces occurrences devient
       'E'.
*****************************************************************************/

PUBLIC void supprimer_menu_complet(tdate Date);

/*
BUT : supprimer tous les menus dont la date est antérieure à Date.

IN : -Date, la date telle que tout menu ayant une date antérieure à elle
     doit être supprimé.

PRE : \.

OUT : \.

EFFETS sur le S.I. :
     -suppression de n occurrences de MENU, de n occurrences de
      PREPARATION et de m occurrences de CONSTITUTION, selon la règle
      suivante : toute occurrence de MENU participant à une association
      PREPARATION dont l'attribut date a une valeur strictement inférieure
      à date est supprimée, ainsi que ces occurrences de PREPARATION et
      toutes les occurrences de CONSTITUTION auxquelles participent ces
      MENU.
*****************************************************************************/
```

```
/*
 * Module   : Opérations externes
 * Type     : Spécifications externes
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Privacy.h"
#include "H_types2.h"

/*********************************/
/*********************************/
/*  MODULE OPERATIONS EXTERNES  */
/*********************************/
/*********************************/

/************************/
/************************/
/*  1 : ACCES EXTERNES  */
/************************/
/************************/

/*************************/
/*************************/
/*  1.1 : ACCES SUR CLE  */
/*************************/
/*************************/

PUBLIC short menu_sur_id1(tdate Date, short Repas,tmenu *Art_menu);

/*
BUT : retourner l'article MENU identifié par (Date, Repas).

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -Art_menu.
      -la valeur de la fonction.

POST : -C1 : il existe un MENU identifié par (Date, Repas).
       -E1 : Art_menu est un pointeur vers une structure qui contient
             la valeur des items de ce MENU.
       -E2 : Art_menu est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_MAUV_CLE.

       -si C1 alors E1 et E3 sinon E2 et E4.
*****************************************************************************/

PUBLIC short plat_sur_id1(short No_plat,short *Art_plat);

/*
BUT : retourner l'article PLAT identifié par No_plat.

IN : -No_plat, le numéro d'un PLAT.

PRE : \.
```

```
OUT : -Art_plat.
      -la valeur de la fonction.

POST : -C1 : il existe un PLAT identifié par No_plat.
       -E1 : Art_plat est un pointeur vers une structure qui contient
             la valeur des items de ce PLAT.
       -E2 : Art_plat est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_MAUV_CLE.

       -si C1 alors E1 et E3 sinon E2 et E4.
*****************************************************************************/

PUBLIC short theme_sur_id1(short No_theme,ttheme *Art_theme);

/*
BUT : retourner l'article THEME identifié par No_theme.

IN : -No_theme, le numéro d'un THEME.

PRE : \.

OUT : -Art_theme.
      -la valeur de la fonction.

POST : -C1 : il existe un THEME identifié par No_theme.
       -E1 : Art_theme est un pointeur vers une structure qui contient
             la valeur des items de ce THEME.
       -E2 : Art_theme est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_MAUV_CLE.

       -si C1 alors E1 et E3 sinon E2 et E4.
*****************************************************************************/

PUBLIC short constitution_sur_id1(tdate Date,short Repas,short Theme,
                                  tconstitution *Art_constitution);

/*
BUT : retourner l'article CONSTITUTION identifié par (Date, Repas, Theme).

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.
     -Theme, le numéro d'un THEME.

PRE : \.

OUT : -Art_constitution.
      -la valeur de la fonction.

POST : -C1 : il existe une CONSTITUTION identifiée par (Date, Repas, Theme).
       -E1 : Art_constitution est un pointeur vers une structure qui
             contient la valeur des items de cette CONSTITUTION.
       -E2 : Art_constitution est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_MAUV_CLE.

       -si C1 alors E1 et E3 sinon E2 et E4.
```

```
*****************************************************************************/

PUBLIC short produit_sur_id1(short Numero,tproduit *Art_produit);

/*
BUT : retourner l'article PRODUIT identifié par Numero.

IN : -Numero, le numéro d'un PRODUIT.

PRE : \.

OUT : -Art_produit.
      -la valeur de la fonction.

POST : -C1 : il existe un PRODUIT identifié par Numero.
       -E1 : Art_produit est un pointeur vers une structure qui contient
             la valeur des items de ce PRODUIT.
       -E2 : Art_produit est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_MAUV_CLE.

       -si C1 alors E1 et E3 sinon E2 et E4.
*****************************************************************************/

PUBLIC short premier_plat_sur_cle1(short Theme,tplat *Art_plat);

/*
BUT : retourner le premier article PLAT appartenant à un THEME.

IN : -Theme, le numéro d'un THEME.

PRE : \.

OUT : -Art_plat.
      -la valeur de la fonction.

POST : -C1 : il existe un THEME T identifié par Theme connu du S.I.
       -C2 : la séquence S des PLAT appartenant au THEME T n'est pas vide.
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_FIN_FICH.
       -E3 : Art_plat est un pointeur vers une structure contenant la
             valeur des items du premier PLAT de S.
       -E4 : Art_plat non défini.

       -si C1 et C2 alors E1 et E3,
        si non C1 ou non C2 alors E2 et E4.
*****************************************************************************/

PUBLIC short suivant_plat_sur_cle1(short Theme,tplat *Art_plat);

/*
BUT : retourner l'article PLAT suivant le dernier repéré appartenant à un
      THEME.

IN : -Theme, le numéro d'un THEME.

PRE : -il existe un THEME T identifié par Theme.
      -la séquence S des articles PLAT appartenant à T est non vide et un de
       ces PLAT, A, y est repéré.
```

```
OUT : -Art_plat.
      -la valeur de la fonction.

POST : -C1 : il y a dans S un article A' suivant A.
       -E1 : Art_plat est un pointeur vers une structure contenant la
             valeur des items de A'.
       -E2 : Art_plat est non défini.
       -E3 : la fonction retourne F_OK.
       -E4 : la fonction retourne F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short premier_plat_sur_cle2(short Plat_pere,tplat *Art_platfils);

/*
BUT : soit P un PLAT identifié par Plat_pere, soit P qui se décompose, P
      est donc cible d'un seul chemin PLTH dont un THEME T est origine, T
      qui se décompose en n THEME fils : T1, T2, ..., Tn, classés par
      ordre de priorité décroissante.  A chacun de ces thèmes Ti
      correspond un PLAT Pi fils de P.  Le but de cette fonction est
      de retourner l'article PLAT P1.

IN : -Plat_pere, le numéro du PLAT père.

PRE : \.

OUT : -la valeur de la fonction.
      -Art_platfils.

POST : -C1 : il existe un PLAT P connu du S.I. identifié par Plat_pere.
       -C2 : P a au moins un PLAT fils.
       -E1 : la valeur de la fonction est F_OK.
       -E2 : la valeur de la fonction est F_FIN_FICH.
       -E3 : Art_platfils est un pointeur vers une structure contenant
             la valeur des items de P1.
       -E4 : Art_platfils est non défini.

       -si C1 et C2 alors E1 et E3,
        si non C1 ou non C2 alors E2 et E4.
*****************************************************************************/

PUBLIC short suivant_plat_sur_cle2(short Plat_pere,tplat *Art_platfils);

/*
BUT : (voir au préalable les spécifications de la fonction précédente).
      Retourner l'article PLAT Pi correspondant au THEME Ti suivant le
      dernier THEME Tj pour lequel on a recherché le PLAT correspondant.

IN : -Plat_pere, le numéro du PLAT père.

PRE : -il existe un PLAT P identifié par Plat_pere et qui se décompose.
      -la séquence S des articles PLAT fils de P, ordonnée selon le
       même ordre que les THEME auxquels ils correspondent n'est pas vide,
       et il y a un de ces PLAT de repéré dans S.
```

```
OUT : -la valeur de la fonction.
      -Art_platfils.

POST : -C1 : il existe un PLAT Pi de S suivant le dernier y repéré.
       -E1 : la valeur de la fonction est F_OK.
       -E2 : la valeur de la fonction est F_FIN_FICH.
       -E3 : Art_platfils est un pointeur vers une structure contenant
             la valeur des items de Pi.
       -E4 : Art_platfils est non défini.

       -si C1 alors E1 et E3,
        si non C1 alors E2 et E4.
*****************************************************************************/

PUBLIC short premier_produit_sur_cle1(short Plat,tproduit *Art_produit);

/*
BUT : retourner le premier article PRODUIT entrant dans la composition
      d'un PLAT.

IN : -Plat, le numéro d'un PLAT.

PRE : \.

OUT : -Art_produit.
      -la valeur de la fonction.

POST : -C1 : il existe un PLAT PL identifié par Plat connu du S.I.
       -C2 : la séquence S des PRODUIT entrant dans la composition de PL
             n'est pas vide.
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_FIN_FICH.
       -E3 : Art_produit est un pointeur vers une structure qui contient
             la valeur des items du premier PRODUIT de S.
       -E4 : Art_produit non défini.

       -si C1 et C2 alors E1 et E3,
        si non C1 ou non C2 alors E2 et E4.
*****************************************************************************/

PUBLIC short suivant_produit_sur_cle1(short Plat,tproduit *Art_produit);

/*
BUT : retourner l'article PRODUIT suivant le dernier repéré entrant dans la
      composition d'un PLAT.

IN : Plat,le numéro d'un PLAT.

PRE : -il existe un PLAT PL identifié par Plat.
      -la séquence S des articles PRODUIT entrant dans la composition
       de PL n'est pas vide et un de ces articles, PR, y est repéré.

OUT : -Art_produit.
      -la valeur de la fonction.

POST : -C1 : il y a dans S un article PR' suivant PR.
       -E1 : Art_produit est un pointeur vers une structure contenant la
             valeur des items de PR'.
       -E2 : Art_produit est non défini.
```

```
       -E3 : la fonction retourne F_OK.
       -E4 : la fonction retourne F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short premier_themeplat_sur_cle1(tdate Date,short Repas,
                                        ttheme *Art_theme,tplat *Art_plat);

/*
BUT : retourner le premier couple (THEME,PLAT) constituant d'un MENU.

IN : -Date, une date.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -Art_theme.
      -Art_plat.
      -la valeur de la fonction.

POST : -C1 : il existe un MENU identifié par (Date,Repas).
       -C2 : la séquence S des couples (THEME,PLAT) constituants de ce
             MENU est non vide.
       -E1 : Art_theme est un pointeur vers une structure qui contient la
             valeur des items du THEME du premier couple de S.
       -E2 : Art_plat est un pointeur vers une structure qui contient la
             valeur des items du PLAT de premier couple de S.
       -E3 : Art_theme et Art_plat non définis.
       -E4 : la fonction vaut F_OK.
       -E5 : la fonction vaut F_FIN_FICH.

       -si C1 et C2 alors E1 et E2 et E4,
        si non C1 ou non C2 alors E3 et E5.
*****************************************************************************/

PUBLIC short suivant_themeplat_sur_cle1(tdate Date,short Repas,
                                        ttheme *Art_theme,tplat *Art_plat);


/*
BUT : retourner le couple (THEME,PLAT) constituant d'un MENU suivant le
      dernier repéré.

IN : -Date, une date.
     -Repas, le numéro d'un REPAS.

PRE : -il existe un MENU M identifié par (Date, Repas).
      -la séquence S des couples (THEME,PLAT) constituants de M est non
       vide, et un de ces couples est repéré dans S.

OUT : -Art_theme.
      -Art_plat.
      -la valeur de la fonction.
```

```
POST : -C1 : la séquence S contient un couple C suivant le dernier
             y repéré.
       -E1 : Art_theme est un pointeur vers une structure qui contient la
             valeur des items du THEME de C.
       -E2 : Art_plat est un pointeur vers une structure qui contient la
             valeur des items du PLAT de C.
       -E3 : Art_theme et Art_plat non définis.
       -E4 : la fonction vaut F_OK.
       -E5 : la fonction vaut F_FIN_FICH.

       -si C1 alors E1 et E2 et E4,
        sinon E3 et E5.
*****************************************************************************/

/*****************************/
/*****************************/
/*  1.2 : ACCES SEQUENTIELS  */
/*****************************/
/*****************************/

PUBLIC short premier_seq_menupreprep(tmenupreprep *Art_menupreprep);

/*
BUT : retourner le premier article PREPARATION et les articles REPAS et
      MENU lui associés.

IN : \.

OUT : -Art_menupreprep.

POST : -C1 : la séquence S des articles PREPARATION connus du S.I.
             n'est pas vide.
       -E1 : Art_menupreprep est un pointeur vers une structure qui
             contient la valeur des items de la première PREPARATION
             de S, ainsi que la valeur des items du REPAS et du MENU lui
             associés.
       -E2 : Art_menupreprep est non défini.
       -E3 : la fonction retourne F_OK.
       -E4 : la fonction retounre F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short suivant_seq_menupreprep(tmenupreprep *Art_menupreprep);

/*
BUT : retourner l'article PREPARATION suivant le dernier repéré, ainsi que
      les articles REPAS et MENU lui associés.

IN : \.

PRE : -la séquence S des articles PREPARATION connus du S.I. n'est pas
       vide et il y a un de ces articles de repéré.

OUT : -Art_menupreprep.
```

```
      -si C1 alors E1 et E2 et E4,
       sinon E3 et E5.
*****************************************************************************/

PUBLIC short premier_seq_preparation (tpreparation *Art_preparation);

/*
BUT : retourner le premier article PREPARATION.

IN : \.

OUT : -Art_preparation.
      -la valeur de la fonction.

POST : -C1 : la séquence des articles PREPARATION n'est pas vide.
       -E1 : Art_preparation est un pointeur vers une structure qui contient
             la valeur des items du premier article de S.
       -E2 : Art_preparation est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short suivant_seq_preparation (tpreparation *Art_preparation);

/*
BUT : retourner l'article PREPARATION suivant le dernier repéré.

IN : \.

PRE : -la séquence S des articles PREPARATION n'est pas vide et un de ces
       articles, P, y est repéré.

OUT : -Art_preparation.
      -la valeur de la fonction.

POST : -C1 : il y a dans S un article qui suit P.
       -E1 : Art_preparation est un pointeur vers une structure qui
             contient la valeur des items de l'article suivant P dans S.
       -E2 : Art_preparation est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short premier_seq_repas(trepas *Art_repas);

/*
BUT : retourner le premier article REPAS.

IN : \.

OUT : -Art_repas.
      -la valeur de la fonction.
```

```
POST : -C1 : la séquence S des articles REPAS n'est pas vide.
       -E1 : Art_repas est un pointeur vers une structure qui contient
             la valeur des items du premier article de S.
       -E2 : Art_repas est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

PUBLIC short suivant_seq_repas(trepas *Art_repas);

/*
BUT : retourner l'article REPAS suivant le dernier repéré.

IN : \.

PRE : -la séquence S des articles REPAS et un de ces articles y est repéré.

OUT : -Art_repas.
      -la valeur de la fonction.

POST : -C1 : il y a dans S un article suivant le dernier y repéré.
       -E1 : Art_repas est un pointeur vers une structure qui contient
             la valeur des items de l'article de S suivant le dernier
             repéré.
       -E2 : Art_repas est non défini.
       -E3 : la fonction vaut F_OK.
       -E4 : la fonction vaut F_FIN_FICH.

       -si C1 alors E1 et E3,
        sinon E2 et E4.
*****************************************************************************/

/****************************/
/****************************/
/*  2 : CREATIONS EXTERNES  */
/****************************/
/****************************/

PUBLIC void creer_preparation(tdate Date,short Repas);

/*
BUT : créer une PREPARATION identifiée par (Date, Repas) et insérer
      cet article comme cible d'un chemin REPR.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : -il n'existe pas de PREPARATION identifiée par (Date, Repas).
      -Date est une date existant dans un calendrier.
      -Date est la date d'un jour non antérieur au jour courant.
      -(Date - date du jour courant + 1) <= param.eloignement_max.
      -Repas identifie un REPAS.
      -dans le premier scénario de création de menus, Date est la date
       du premier jour non traité non antérieur au jour courant, et Repas
       est le numéro du premier REPAS non traité de ce jour.
```

```
OUT : \.

EFFETS sur le S.I. :
      -création d'un article PREPARATION ayant Date pour valeur de l'item
       date.
      -insertion de cet article comme cible du chemin REPR dont l'article
       REPAS identifié par Repas est origine.
*************************************************************************/

PUBLIC void creer_menu(tdate Date,short Repas,short Nbr_personnes);

/*
BUT : créer une MENU identifié par (Date, Repas) et ayant Nbr_personnes
      pour valeur de l'item nbr_pers, et inserer cet article comme cible
      d'un chemin PRME.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : -(Date, Repas) identifie un article PREPARATION connu du S.I.
      -il n'existe pas d'article MENU cible du chemin PRME dont l'article
       PREPARATION identifié par (Date, Repas) est origine.
      -0<=Nbr_personnes<=param.borne_pers.

OUT : \.

EFFETS sur le S.I. :
      -création d'un article MENU qui aura pour valeur de l'item :
         -nbr_pers : Nbr_personnes,
         -modifie : VRAI,
         -imprime : FAUX.
      -insertion de cet article comme cible du chemin PRME dont l'article
       PREPARATION identifié par (Date, Repas) est origine.
*************************************************************************/

PUBLIC void creer_constitution(tdate Date,short Repas,short Theme,
                               short Plat);

/*
BUT : créer un article CONSTITUTION et l'insérer comme cible d'un chemin
      MECO et comme origine d'un chemin COTH et d'un chemin COPL.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.
     -Theme, le numéro d'un THEME.
     -Plat, le numéro d'un PLAT.

PRE : -(Date, Repas) identifie un MENU connu du S.I.
      -Theme identifie un article THEME connu du S.I.
      -Plat identifie un article PLAT connu du S.I.
      -Plat identifie un PLAT possible pour le THEME Theme.
      -il n'existe pas de CONSTITUTION identifiée par (Date,Repas,Thème).

OUT : \.

EFFETS sur le S.I. :
      -création d'un article CONSTITUTION ayant 'A' pour valeur de l'item
       modifie_plat.
```

```
      -insertion de cet article comme cible du chemin MECO dont l'article
       MENU identifié par (Date, Repas) est origine.
      -insertion de cet article CONSTITUTION comme origine d'un chemin dont
       l'article THEME identifié par Theme est cible.
      -insertion de cet article CONSTITUTION comme origine d'un chemin dont
       l'article PLAT identifié par Plat est cible.
*****************************************************************************/

/*******************************/
/*******************************/
/*  3 : MISES-A-JOUR EXTERNES  */
/*******************************/
/*******************************/

PUBLIC short maj_menu_nbrpers_modifie_sur_id1(tdate Date,short Repas,
                                              short Nbr_pers,
                                              boolean Modifie);

/*
BUT : remplacer la valeur actuelle des items nbr_pers et modifie du MENU
      identifié par (Date, Repas).

IN : -Date, la date d'une PREPARATION.
     -Repas,le numéro d'un REPAS.
     -Nbr_pers, la nouvelle valeur du nombre de personnes du menu.
     -Modifie,la nouvelle valeur de l'item modifie.

PRE : -0<=Nbr_pers<=param.borne_pers.
      -Modifie vaut VRAI.

OUT : -la valeur de la fonction.

POST : -C1 : il existe un MENU identifié par (Date, Repas).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      - si C1 alors :
         -la valeur actuelle de l'item nbr_pers du MENU identifié par
          (Date, Repas) est remplacée par Nbr_pers,
         -la valeur actuelle de l'item modifie de ce MENU est remplacée
          par Modifie,
      - sinon rien.
*****************************************************************************/

PUBLIC short maj_menu_imprime_sur_id1(tdate Date,short Repas,
                                      boolean Imprime);

/*
BUT : remplacer la valeur actuelle de l' item imprime du MENU identifié
      par (Date, Repas).

IN : -Date, la date d'une PREPARATION.
     -Repas,le numéro d'un REPAS.
     -Imprime,la nouvelle valeur de l'item imprime.

PRE : \.
```

```
OUT : -la valeur de la fonction.

POST : -C1 : il existe un MENU identifié par (Date, Repas).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      - si C1 alors :
        -la valeur actuelle de l'item imprime du MENU identifié par
         (Date, Repas) est remplacée par Imprime,
      - sinon rien.
*****************************************************************************/

PUBLIC short maj_constitution_modifieplat_sur_id1(tdate Date, short Repas,
                                                  short Theme,
                                                  char Modifie_plat);

/*
BUT : remplacer la valeur actuelle de l'item modifie_plat de la
      CONSTITUTION identifié par (Date, Repas, Theme).

IN : -Date, la date d'une PREPARATION.
     -Repas,le numéro d'un REPAS.
     -Theme, le numéro d'un THEME.
     -Modifie_plat,la nouvelle valeur de l'item modifie_plat.

PRE : -Modifie_plat vaut 'A', 'E' ou 'I'.

OUT : -la valeur de la fonction.

POST : -C1 : il existe une CONSTITUTION identifiée par (Date, Repas, Theme).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      - si C1 alors :
        -la valeur actuelle de l'item modifie_plat de la CONSTITUTION
         identifiée par (Date, Repas, Theme) est remplacée par
         Modifie_plat,
      - sinon rien.
*****************************************************************************/

PUBLIC short maj_constitution_plat_sur_id1(short Plat,tdate Date,
                                           short Repas,short Theme,
                                           char Modif_plat);

/*
BUT : remplacer la valeur actuelle de l'item modifie_plat de la
      CONSTITUTION identifié par (Date, Repas, Theme), retirer un PLAT d'un
      chemin COPL et en insérer un autre.

IN : -Date, la date d'une PREPARATION.
     -Repas,le numéro d'un REPAS.
     -Theme, le numéro d'un THEME.
```

```
     -Plat, le numéro d'un PLAT.
     -Modifie_plat,la nouvelle valeur de l'item modifie_plat.

PRE : -Modifie_plat vaut 'A'.
      -Plat identifie un PLAT connu du S.I.
      -si le Theme identifie un THEME connu du S.I., alors le PLAT
       identifié par Plat appartient à ce THEME.

OUT : -la valeur de la fonction.

POST : -C1 : il existe une CONSTITUTION identifiée par (Date, Repas, Theme).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      - si C1 alors :
         -la valeur actuelle de l'item modifie_plat de la CONSTITUTION
          identifiée par (Date, Repas, Theme) est remplacée par
          Modifie_plat,
         -le PLAT cible du chemin COPL dont cette CONSTITUTION est origine
          est retiré de ce chemin et le PLAT identifié par Plat est inséré
          comme cible de ce chemin.
      - sinon rien.
*****************************************************************************/

/*****************************/
/*****************************/
/* 4 : SUPPRESSIONS EXTERNES */
/*****************************/
/*****************************/

PUBLIC short suppression_preparation_sur_id1(tdate Date,short Repas);

/*
BUT : supprimer une PREPARATION.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -la valeur de la fonction.

POST : -C1 : il existe une PREPARATION identifiée par (Date, Repas).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      -si C1 alors suppression de l'article PREPARATION identifié par
       (Date,Repas), sinon rien.
*****************************************************************************/
```

```
PUBLIC short suppression_menu_sur_id1(tdate Date,short Repas);

/*
BUT : supprimer un MENU.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -la valeur de la fonction.

POST : -C1 : il existe une MENU identifiée par (Date, Repas).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      -si C1 alors suppression de l'article MENU identifié par
       (Date,Repas), sinon rien.
*****************************************************************************/

PUBLIC short suppression_constitution_sur_cle1(tdate Date,short Repas);

/*
BUT : supprimer les CONSTITUTION associées à un MENU.

IN : -Date, la date d'une PREPARATION.
     -Repas, le numéro d'un REPAS.

PRE : \.

OUT : -la valeur de la fonction.

POST : -C1 : il existe une CONSTITUTION au moins identifiée par
             (Date, Repas).
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.

       - si C1 alors E1 sinon E2.

EFFETS sur le S.I. :
      -si C1 alors suppression des articles CONSTITUTION cibles des
       chemins MECO dont le MENU identifié par (Date,Repas) est origine,
       sinon rien.
*****************************************************************************/
```

```
/*********************************/
/*********************************/
/*  5 : VERIFICATIONS DIVERSES  */
/*********************************/
/*********************************/

PUBLIC short verifier_theme_origine_thth(short Theme,boolean *Origine);

/*
BUT : vérifier si un THEME est origine d'un chemin THTH.

IN : -Theme, le numéro d'un THEME.

PRE : \.

OUT : -Origine.
      -la valeur de la fonction.

POST : -C1 : il existe un THEME identifié par Theme connu du S.I.
       -C2 : le THEME identifié par Theme est origine d'un chemin THTH.
       -E1 : la fonction vaut F_OK.
       -E2 : la fonction vaut F_MAUV_CLE.
       -E3 : Origine est un pointeur vers la valeur VRAI.
       -E4 : Origine est un pointeur vers la valeur FAUX.
       -E5 : Origine est non défini.

       -si C1 et C2 alors E1 et E3, si C1 et non C2 alors E1 et E4,
        si non C1 alors E2 et E5.

*****************************************************************************/
```

```
/*
 * Module   : Opérations sur les dates
 * Type     : Spécifications externes
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "time.h"
#include "H_PrimBD.h"
#include "H_Types2.h"

PUBLIC tdatecournom jour_courant;
PUBLIC tparamutil param;

/*************************************/
/*************************************/
/*  MODULE OPERATIONS SUR LES DATES  */
/*************************************/
/*************************************/

PUBLIC calculer_jour_courant();

/*
BUT : fournir la date du jour courant.

IN : /.

PRE : /.

OUT : jour_courant (variable globale de type tdatecournom).

POST : jour_courant est la date du jour courant.
*****************************************************************************/

PUBLIC boolean compare_date(tdate Date1,tdate Date2,char Signe);

/*
BUT : comparer deux dates.

IN : -Date1, la première date.
     -Date2, la seconde date.
     -Signe, le signe de comparaison.

PRE : -Signe vaut '<', '>' ou '='.

OUT : -la valeur de la fonction.

POST : -si (Date1 Signe Date2) alors la fonction vaut VRAI, sinon la
        fonction vaut FAUX.
*****************************************************************************/

PUBLIC date_suivante(tdate *Date);

/*
BUT : retourner la date suivant celle donnée.

IN : -Date, un pointeur vers une date.

PRE : -la date pointée existe au calendrier.
```

```
OUT : -Date'.

POST : -Date' est un pointeur vers la date suivant Date.
*****************************************************************************/

PUBLIC calculer_calendrier();

/*
BUT : retourner le calendrier des jours traités, des jours à traiter,
      l'état de l'impression des menus du jour, compte tenu du scénario
      de création de menus dans lequel on travaille.

IN : -Param, les paramètres utilisateur (variable globale);

PRE : /.

OUT : -Calendrier (variable globale de type tcalend).
      -Calendrier_long (variable globale de type short).

POST : -Calendrier pointe vers le tableau contenant la date des jours
       qu'il est possible de traiter (compte tenu du scénario de création
       de menus et de param) et des jours traités non antérieurs
       au jour courant.
       Chaque cellule du tableau contient :
          -la date du jour sous la forme AAAA-MM-JJ-NOM DU JOUR, avec MM
           compris entre 0 et 11.
          -un champs ("traite") indiquant si le jour a déjà été traité :
              -si le jour a été traité, alors traite vaut True,
              -il vaut False sinon,
          -un champs ("imprime") indiquant si les menus du jour ont déjà
           été imprimés :
              -si les menus du jour ont déjà été imprimés, alors imprime
               vaut True,
              -il vaut False, sinon.
      -Calendrier_long est la taille effective de ce tableau.
*****************************************************************************/
```

Texte du code source.

```
/
 * Définition des types
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#ifndef H_TYPES2
   #define H_TYPES2 "Définiton des types standard"
   #include <exec/types.h>
   #include "H_Privacy.h"
   #define longnomplat          25
   #define longsonplat          35
   #define longnomtheme         20
   #define longsontheme         30
   #define longnomrepas         15
   #define longsonrepas         25
   #define longnomproduit       30
   #define longsonproduit       40
   #define longnomarticle        5
   #define longsonarticle        8
   #define longnomjour           8
   #define longsonjour          15
   #define longnommois           9
   #define longsonmois          18
   #define longprenom           15
   #define maxthememenu         10
   #define infini             9999
   #define maxthemedec           1
   #define maxfilstheme          3
   #define maxrepas              1
   #define maxthemes             8
   #define maxplatstheme        20
   #define maxproduitsplat      10
   #define max_periode_a_traiter 7
   #define max_eloignement       9
   #define max_unite            15
   #define max_dizaine          15
   #define maxproduits          30
   #define max_quantite         50
   #define max_quantite_affiche 11

   typedef short   booleanint;
   typedef char    tnom_plat[longnomplat+1];
   typedef char    tnom_repas[longnomrepas+1];
   typedef char    tnom_theme[longnomtheme+1];
   typedef char    tnom_produit[longnomproduit+1];
   typedef char    tson_plat[longsonplat+1];
   typedef char    tson_repas[longsonrepas+1];
   typedef char    tson_theme[longsontheme+1];
   typedef char    tson_produit[longsonproduit+1];
   typedef char    tnom_article[longnomarticle+1];
   typedef char    tson_article[longsonarticle+1];
   typedef char    tnom_jour[longnomjour+1];
   typedef char    tson_jour[longsonjour+1];
   typedef char    tnom_mois[longnommois+1];
   typedef char    tson_mois[longsonmois+1];
   typedef char    tprenom[longprenom+1];
```

```
typedef struct {
                short annee;
                   short mois;
                   short jour;
              } tdate;

typedef struct {
                short     annee;
                short     mois;
                short     jour;
                short     nom;
              } tdatenom;

typedef struct {
                short     annee;
                short     mois;
                short     jour;
                char      nom[9];
              } tdatecournom;

typedef struct {
               short     no_plat;
                tnom_plat nom;
                short     article;
                tson_plat son_nom;
                short     cout;
              } tplat;

typedef struct {
                short      no_theme;
                tnom_theme nom;
                short      article;
                tson_theme son_nom;
                short      priorite;
                UBYTE      couleur;
              } ttheme;

typedef struct {
                short   nnbr_pers;
                short   nbr_pers;
                boolean modifie;
                boolean imprime;
              } tmenu;

typedef struct {
                short      no_repas;
                tnom_repas nom;
                short      article;
                tson_repas son_nom;
                short      priorite;
              } trepas;

typedef struct {
                char modifie_plat;
              } tconstitution;

typedef struct {
                tdate date;
              } tpreparation;
```

```
typedef struct {
                short no_produit;
                tnom_produit nom;
                short article;
                short article2;
                tson_produit son_nom;
                short classe_stock;
                short prix;
                short cond_vente;
                short cond_util;
                short qte_cond_util_cond_vente;
              } tproduit;

typedef struct {
                short no_produit;
                tnom_produit nom;
                short article;
                short article2;
                tson_produit son_nom;
                short classe_stock;
                short prix;
                short cond_vente;
                short cond_util;
                short qte_cond_util_cond_vente;
                float qtecompos;
              } tproduitcompos;

typedef struct {
                short no_plat;
              } tappartenanceint;

typedef struct {
                short no_theme;
                short no_plat;
                char  modifie_plat;
                short ptr_suivant;
              } tconstitutionint;

typedef struct {
                short      no_theme;
                tnom_theme nom;
                short      article;
                tson_theme son_nom;
                short      priorite;
                UBYTE      couleur;
                short      no_theme_pere;
                booleanint a_fils;
              } tthemeint;

typedef struct {
                short      nnbr_pers;
                short      nbr_pers;
                booleanint modifie;
                booleanint imprime;
              } tmenuint;
```

```
typedef struct {
                short      no_repas;
                tnom_repas nom;
                short      article;
                tson_repas son_nom;
                short      priorite;
              } trepasint;

typedef struct {
                short     no_plat;
                tnom_plat nom;
                short     article;
                tson_plat son_nom;
                short     cout;
              } tplatint;

typedef struct {
                tdate date;
              } tpreparationint;

typedef struct {
                short no_produit;
                tnom_produit nom;
                short article;
                short article2;
                tson_produit son_nom;
                short classe_stock;
                short prix;
                short cond_vente;
                short cond_util;
                short qt_cond_util_cond_vente;
              } tproduitint;

typedef struct {
                short no_produit;
                float qte_produit;
              } tcompos_platint;

typedef struct {
                short no_plat;
                short ptr_liensplats;
                short nbr_fils;
              } tptr_liens_plats;

typedef struct {
                short no_theme;
                short ptr_app;
                short nbr_plats;
              } tptr_appartenance;

typedef struct {
                short no_plat;
                short ptr_compos_plat;
                short nbr_produits;
              } tptr_compos_plat;
```

```
typedef struct {
                 tdate date;
                 short repas;
                 short ptr_menu;
                 short ptr_const;
               } tptr_menu_const;

typedef struct {
                 short no_plat_fils;
               } tliens_plats;

typedef struct {
                 short theme;
                 short plat;
               } tconstituants_enreg;

typedef struct {
                 short      no_theme;
                 tnom_theme nom;
                 short      article;
                 tson_theme son;
                 short      priorite;
                 UBYTE      couleur;
                 short      ptr_plat;
               } tconstituants_ext;

typedef struct {
                 short     no_plat;
                 tnom_plat nom;
                 short     article;
                 tson_plat son;
                 short     cout;
                 short     nbr_fils;
               } tplat_pere_fils;

typedef struct {
                 short      no_theme;
                 tnom_theme nom;
                 short      article;
                 tson_theme son;
                 short      priorite;
                 UBYTE     couleur;
                 short      nbr_fils;
               } ttheme_pere_fils;

typedef struct {
                 short      no_theme;
                 tnom_theme nom;
                 short      article;
                 tson_theme son;
                 short      priorite;
                 UBYTE      couleur;
                 short      nbr_fils;
                 boolean    select;
               } ttheme_pere_fils_bool;
```

```
typedef struct {
                short   nnbr_pers;
               short   nbr_pers;
                boolean modifie;
                boolean imprime;
                tdate   date;
                short   no_repas;
              } tmenupreprep;

typedef struct {
                tprenom prenom;
                boolean son;
                short   scmenus;
                short   sclistc;
                boolean tutoye;
                boolean themeelem;
                short   eloignement_max;
                short   periode_max_a_traiter;
                boolean budget;
              } tparamutil;

typedef struct {
                short annee;
                short mois;
                short jour;
                short nom;
                boolean traite;
                boolean imprime;
              } tcalendrier;

typedef struct {
                tnom_jour nom;
                tson_jour son;
              } tjour;

typedef struct {
                tnom_mois nom;
                tson_mois son;
              } tmois;

typedef struct {
                short annee;
                short mois;
                short jour;
                short repas;
                short nbrpers;
                tconstituants_enreg constituants[maxthememenu];
                short nbrconst;
              } tmenu_enreg;
typedef struct {
                tdate debut;
                tdate fin;
                boolean achat_fait;
              } tperiode_achat;
```

```
typedef struct {
                  short num_plat;
                  char nom[3*(longnomplat+2)];
                  char son[3*(longsonplat+2)];
                  char fichier[12];
                  short num_theme;
                  UBYTE couleur_theme;
                  boolean decomp;
                } tplatstheme;

typedef struct {
                  short num_plat;
                  tnom_plat nom;
                  boolean entier;
                  UBYTE couleur;
                } tplatsaffiche;

typedef struct {
                  char nom[6];
                  char son[7];
                 } tarticle;

typedef struct {
                  char nom[10];
                  char son[10];
                 } tconditionnement;

#endif
```

```
/*
 * ---> Module  : Coordinateur
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "stdio.h"
#include "H_Gadget.h"
#include "H_Ecran.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_PrimBD.h"

/*--------------------------------*/
/* Variables globales au programme */
/*--------------------------------*/
boolean               VOIX, CONFIRM_OK=True;
short                 calendrier_long, themes_long, plat_long, repas_long,
                      nbrrepas, nbrmenupreprep;
struct Screen         *Screen;
struct Gadget         gadget[max_gadgets];
st_msg                tab_msg[nbre_msg];
tarticle              Tab_Article[12];
tconditionnement      Tab_Cond[10];
tmenupreprep          tab_menupreprep[(max_eloignement*maxrepas)];
tparamutil            param;
tdatecournom          jour_courant;
tcalendrier           calendrier[max_eloignement];
ttheme_pere_fils      Tab_themes[maxthemes];
tplat_pere_fils       Tab_plats[maxplatstheme*(maxfilstheme+1)];
tmenu_enreg           Menu_enreg;
tjour                 tab_nomjours[7];
tmois                 tab_nommois[12];
char                  tab_unite[17][max_unite],
                      tab_dizaine[11][max_dizaine],
                      *disk_name="r";
tplatstheme           Tab_Plat_Theme[maxthemes];
ttheme_pere_fils_bool Tab_themes_b[maxthemes];
tproduitcompos        Tab_Produit[maxproduits];
FILE                  *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
                      *fdappartenance, *fdptrappartenance, *fdptrmenuconst,
                      *fdconst, *fdptrapp, *fdmenu, *fdptrcomposplat,
                      *fdproduit, *fdapp,
                      *fdptrcomposplat, *fdproduit, *fdcomposplat;

PRIVATE USHORT selection, selection_menu, sel;

PRIVATE char *upper(string)
char *string;
{
 int i;
 for(i=0;i<strlen(string);i++)
    if((string[i]>='a')&&(string[i]<='z'))
      string[i]=string[i]+'a'-'A';
    else if((string[i]=='à')||
            (string[i]=='ä')||(string[i]=='â'))
         string[i]='A';
    else if((string[i]=='é')||(string[i]=='è')||
```

```
          (string[i]=='ë')||(string[i]=='ê'))
        string[i]='E';
    else if((string[i]=='ï')||(string[i]=='î'))
        string[i]='I';
    else if((string[i]=='ù')||(string[i]=='ü')||
          (string[i]=='û'))
        string[i]='U';
    else if((string[i]=='ö')||(string[i]=='ô'))
        string[i]='O';
    else if(string[i]=='ÿ')
        string[i]='Y';
    else if(string[i]=='ç')
        string[i]='C';
 return string;
}

PRIVATE char *strflush(chaine, lng)
char *chaine;
short lng;
{
 int i;
 for(i=0;i<lng;chaine[i++]='\0');
 return chaine;
}

main()
{
 char prenom[256];
 USHORT fin;
 /*----------------------------------*/
 /* Coordinateur                     */
 /*----------------------------------*/
 Initialisation();
 ChargeFond("ecran_vide.lbm");
 strflush(prenom,256);
 strcpy(prenom,param.prenom);
 printf("avant ecran 0\n");
 fin=ecran0(prenom);
 printf("apres ecran 0\n");
 while((!eval(upper(prenom),"=",param.prenom))&&(fin!=0))
 {
  ConfirmationOK(tab_msg[1]);
  fin=ecran0(prenom);
 }
 if(fin!=0)
 {
  do
  {
   selection=ecran1();
   switch(selection)
   {
      case(1) :
         do
         {
            selection_menu=ecran2();
            switch(selection_menu)
            {
               case(1) :
                  do {} while (CreationMenu()!=0); break;
```

```
                case(2) :
                   do {} while (ModificationMenu()!=0); break;
                case(3) :
                   do {} while (ConsultationMenu(NULL)!=0); break;
                case(4) :
                   do {} while (ImpressionMenu(NULL)!=0); break;
                default : break;
             }
          } while (selection_menu!=0);break;
       case(2) : ListeCourses();break;
       default : break;
    }
   } while (selection!=0);
 }
 Fermeture();
}
```

```
/*
 * Module   : CoordinateurInitialisation
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "stdio.h"
#include "H_Gadget.h"
#include "H_Ecran.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_PrimBD.h"

/*--------------------------------*/
/* Variables globales au programme */
/*--------------------------------*/
PUBLIC struct Screen *Screen;
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC boolean VOIX;
PUBLIC boolean CONFIRM_OK;
PUBLIC tparamutil param;
PUBLIC short nbrmenupreprep;
PUBLIC tarticle Tab_Article[12];
PUBLIC tconditionnement Tab_Cond[10];
PUBLIC tmenupreprep tab_menupreprep[(max_eloignement*maxrepas)];
PUBLIC tdatecournom jour_courant;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC ttheme_pere_fils Tab_themes[maxthemes];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme*(maxfilstheme+1)];
PUBLIC tmenu_enreg Menu_enreg;
PUBLIC short calendrier_long;
PUBLIC short themes_long;
PUBLIC short plat_long;
PUBLIC short repas_long;
PUBLIC short nbrrepas;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC tproduitcompos Tab_Produit[maxproduitsplat*maxthememenu];
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
            *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
            *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit, *fdapp,
            *fdptrcomposplat, *fdproduit, *fdcomposplat;
PUBLIC char *disk_name;

void ChargerMessages(void)
{
 FILE *fdmsg;
 short Pos=POS_DEBUT;
 fdmsg=ouvrir_fichier("Messages",R,disk_name);
 lire_fichier(fdmsg,disk_name,(char *)tab_msg,nbre_msg*sizeof(st_msg),&Pos);
 fermer_fichier(fdmsg,disk_name);
}
```

```
void ChargerParams(void)
{
 FILE *fdparams;
 short Pos=POS_DEBUT;
 fdparams=ouvrir_fichier("Parametres",R,disk_name);
 lire_fichier(fdparams,disk_name,(char *) &param,sizeof(tparamutil),&Pos);
 fermer_fichier(fdparams,disk_name);
}

/*-----------------------------------------------------------------*/
/* Procédure de chargement des gadgets dans la table gadget[0..x] */
/*-----------------------------------------------------------------*/
void ChargerGadgets()
{
 OuvreGadget(&gadget[0],"Choix_Menu.lbm");
 OuvreGadget(&gadget[1],"Choix_Liste.lbm");
 OuvreGadget(&gadget[2],"Stop.lbm");
 OuvreGadget(&gadget[3],"Message.lbm");
 OuvreGadget(&gadget[4],"Ok.lbm");
 OuvreGadget(&gadget[5],"Oui.lbm");
 OuvreGadget(&gadget[6],"Non.lbm");
 OuvreGadget(&gadget[7],"Retour.lbm");
 OuvreGadget(&gadget[8],"Choix_Création_Menu.lbm");
 OuvreGadget(&gadget[9],"Choix_Modif_Menu.lbm");
 OuvreGadget(&gadget[10],"Choix_Consult_Menu.lbm");
 OuvreGadget(&gadget[11],"Choix_Impress_Menu.lbm");
 OuvreGadget(&gadget[12],"Flèche_Haut.lbm");
 OuvreGadget(&gadget[13],"Flèche_bas.lbm");
 OuvreGadget(&gadget[14],"Sélection.lbm");
 OuvreGadget(&gadget[15],"img_pers.lbm");
 OuvreGadget(&gadget[16],"Fin.lbm");
 OuvreGadget(&gadget[17],"Menu.lbm");
 OuvreGadget(&gadget[18],"img_plat.lbm");
 OuvreGadget(&gadget[19],"selection.lbm");
 OuvreGadget(&gadget[20],"poubelle.lbm");
 OuvreGadget(&gadget[21],"Choix_Consult_Produit.lbm");
 OuvreGadget(&gadget[22],"Choix_Impress_Produit.lbm");
 OuvreGadget(&gadget[23],"img_produit.lbm");
 OuvreGadget(&gadget[24],"bliss_produit.lbm");
 OuvreGadget(&gadget[26],"Quantite.lbm");
}

void TrierThemes(themes, lng)
ttheme_pere_fils themes[];
short lng;
{
 int i;
 ttheme_pere_fils *thm;
 thm=(ttheme_pere_fils *)malloc(lng*sizeof(ttheme_pere_fils));
 memcpy(thm,themes,lng*sizeof(ttheme_pere_fils));
 for(i=0;i<lng;themes[thm[i++].priorite]=thm[i]);
 free(thm);
}

PRIVATE void Ouvrir_fichiers()
{
 fdtheme=ouvrir_fichier("theme",R,disk_name);
 fdplat=ouvrir_fichier("plat",R,disk_name);
 fdproduit=ouvrir_fichier("produit",R,disk_name);
```

```
 fdcomposplat=ouvrir_fichier("compos_plat",R,disk_name);
 fdptrcomposplat=ouvrir_fichier("ptr_compos_plat",R,disk_name);
 fdapp=ouvrir_fichier("appartenance",R,disk_name);
 fdptrapp=ouvrir_fichier("ptr_appartenance",R,disk_name);
 fdliensplats=ouvrir_fichier("liens_plats",R,disk_name);
 fdmenu=ouvrir_fichier("menu",RW,disk_name);
 fdptrmenuconst=ouvrir_fichier("ptr_menu_const",RW,disk_name);
 fdconst=ouvrir_fichier("constitution",RW,disk_name);
 fdptrliensplats=ouvrir_fichier("ptr_liens_plats",R,disk_name);
}

void Initialisation()
{
 struct NewScreen NewScreen =
 {
  0,0,320,256,5,0,1,NULL,CUSTOMSCREEN,NULL,NULL,NULL,NULL
 };
 tdate *jc;
 OuvrirLib();
 if((Screen=(struct Screen *) OpenScreen(&NewScreen))==NULL)
  {
   printf("erreur sur l'ecran\n");
   FermerLib();
   exit(False);
  }
 initialiser_tab_nommois();
 ChargeFond("entete.lbm");
 AfficheFond();
 ChargerGadgets();
 ChargerMessages();
 ChargerParams();
 VOIX=param.son;
 Ouvrir_fichiers();
 fournir_themes_et_fils(Tab_themes, &themes_long);
/* TrierThemes(Tab_themes, themes_long);*/
 fournir_menu_prep_repas(tab_menupreprep,&nbrmenupreprep);
 calculer_jour_courant();
 GarnirTab_unite();
 GarnirTab_dizaine();
 initialiser_tab_nomjours();
 calculer_calendrier();
 jc=calendrier;
 supprimer_menu_complet(*jc);
 initialiser_tabart(Tab_Article);
 initialiser_tab_cond(Tab_Cond);
 FermeFond();
}
```

```
/*
 * ---> Module  : FermetureCoordinateur
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "stdio.h"
#include "H_Gadget.h"
#include "H_Ecran.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_PrimBD.h"

/*--------------------------------*/
/* Variables globales au programme */
/*--------------------------------*/
PUBLIC struct Screen *Screen;
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
            *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
            *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit, *fdapp,
            *fdptrcomposplat, *fdproduit, *fdcomposplat;

PUBLIC char *disk_name;

void Fermeture()
{
 int i;
 FermeFond();
 for(i=0;i<max_gadgets;i++)
    FermeGadget(&gadget[i]);
 CloseScreen(Screen);
 fermer_fichier(fdtheme, disk_name);
 fermer_fichier(fdplat, disk_name);
 fermer_fichier(fdapp, disk_name);
 fermer_fichier(fdptrapp, disk_name);
 fermer_fichier(fdliensplats, disk_name);
 fermer_fichier(fdptrliensplats, disk_name);
 fermer_fichier(fdproduit, disk_name);
 fermer_fichier(fdcomposplat, disk_name);
 fermer_fichier(fdptrcomposplat, disk_name);
 FermerLib();
}
```

```
/*
 * Module   : CréationMenu
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ecran.h"
PUBLIC short calendrier_long;
PUBLIC short themes_long;
PUBLIC short plat_long;
PUBLIC tparamutil param;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC ttheme_pere_fils Tab_themes[maxthemes];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme*(maxfilstheme+1)];
PUBLIC tmenu_enreg Menu_enreg;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC st_msg tab_msg[nbre_msg];

PRIVATE UBYTE CouleurTheme;

/************************************************/
PRIVATE void jour_suivant_a_traiter(jour_suivant,fin_jour)
/************************************************/

tdatenom *jour_suivant;
boolean *fin_jour;

{
   boolean trouve;
   short i;

   *fin_jour=True;
   trouve=False;
   i=0;

   while ((i<calendrier_long) && (trouve==False))
         {
            if (calendrier[i].traite==False)
               {
                  jour_suivant->annee=calendrier[i].annee;
                  jour_suivant->mois=calendrier[i].mois;
                  jour_suivant->jour=calendrier[i].jour;
                  jour_suivant->nom=calendrier[i].nom;
                  *fin_jour=False;
                  trouve=True;
               }
               else ++i;
         }
}
```

```
/***************************/
PRIVATE void marquer_jour_traite(jour)
/***************************/

tdate jour;
{
   tdate date_temp;
   boolean marque;
   short i;

   i=0;
   marque=False;
   while (marque==False)
         {
            date_temp.annee=calendrier[i].annee;
            date_temp.mois=calendrier[i].mois;
            date_temp.jour=calendrier[i].jour;
            if (compare_date(date_temp,jour,'=')==True)
               {
                  calendrier[i].traite=True;
                  marque=True;
               }
               else ++i;
         }
}

PRIVATE void InitMenu_enreg(void)
{
 int i;
 for(i=0;i<maxthememenu;i++)
    Menu_enreg.constituants[i].plat=-1;
}

PRIVATE USHORT tab_ptr(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes_b[i].no_theme!=no_theme)
        &&(i<themes_long)) i++;
 return i;
}

PRIVATE void TraitePlat(num_theme)
USHORT num_theme;
{
 int i;
 for(i=0;i<maxplatstheme;i++)
    strcpy(Tab_plats[i].nom,"\0");
 fournir_plats_et_fils_sur_cle1(num_theme,Tab_plats,&plat_long);
 ecran6(num_theme, CouleurTheme);
}

PRIVATE void TraiteMenu()
{
 int i;
 ecran7(True);
 for(i=0;i<maxthemes;i++)
  Tab_themes_b[i].select=eval(Tab_Plat_Theme[i].nom,"=","") ? False : True;
}
```

```
PRIVATE boolean MenuVide()
{
 int i;
 boolean retval=True;
 for(i=0;i<maxthemes;i++)
    retval=(Tab_Plat_Theme[i].num_plat==-1) ? retval : False;
 return retval;
}

PRIVATE void EnregistreMenu(date, nb_pers)
tdate date;
USHORT nb_pers;
{
 int i,num_theme, j=0;
 tconstituants_enreg tab_const[maxthemes];
 Menu_enreg.nbrpers=nb_pers;
 Menu_enreg.annee=date.annee;
 Menu_enreg.mois=date.mois;
 Menu_enreg.jour=date.jour;
 Menu_enreg.repas=0;
 for(num_theme=0;num_theme<maxthemes;num_theme++)
 {
  Menu_enreg.constituants[Tab_themes[tab_ptr(num_theme)].priorite].theme=
        Tab_Plat_Theme[Tab_themes[tab_ptr(num_theme)].priorite].num_theme;
  Menu_enreg.constituants[Tab_themes[tab_ptr(num_theme)].priorite].plat=
        Tab_Plat_Theme[Tab_themes[tab_ptr(num_theme)].priorite].num_plat;
 }

 printf("Date %d/%d/%d\trepas %d\tpersonnes %d\n",
                                Menu_enreg.annee,Menu_enreg.mois,
                                Menu_enreg.jour,Menu_enreg.repas,
                                Menu_enreg.nbrpers);
 for(i=0;i<maxthememenu;i++)
 {
    printf("Constituant %d : %d\n",i,Menu_enreg.constituants[i].plat);
    if(Menu_enreg.constituants[i].plat!=-1)
      tab_const[j++]=Menu_enreg.constituants[i];
 }
 if(j==0)
   Menu_enreg.nbrpers=0;
 enregistrer_menu_complet(date,Menu_enreg.repas,Menu_enreg.nbrpers,
                          tab_const, j);
}

USHORT CreationMenu(void)
{
 USHORT nb_theme, nb_pers, i;
 USHORT sel=2, choix;
 tdatenom jour;
 tdate *date;
 boolean fin_jour, vide=True;
 Menu_enreg.nbrconst=0;
 date=(tdate *) &jour;
 InitMenu_enreg();
 jour_suivant_a_traiter(&jour, &fin_jour);
 for(i=0;i<maxthemes;i++)
 {
  strcpy(Tab_Plat_Theme[i].nom,"");
```

```
 strcpy(Tab_Plat_Theme[i].son,"");
 Tab_Plat_Theme[i].decomp=False;
 Tab_Plat_Theme[i].num_plat=-1;
 Tab_Plat_Theme[i].num_theme=0;
 Tab_Plat_Theme[i].couleur_theme=0;
}
for(i=0;i<themes_long;i++)
{
 strcpy(Tab_themes_b[i].nom,Tab_themes[i].nom);
 strcpy(Tab_themes_b[i].son,Tab_themes[i].son);
 Tab_themes_b[i].no_theme=Tab_themes[i].no_theme;
 Tab_themes_b[i].article=Tab_themes[i].article;
 Tab_themes_b[i].priorite=Tab_themes[i].priorite;
 Tab_themes_b[i].couleur=Tab_themes[i].couleur;
 Tab_themes_b[i].nbr_fils=Tab_themes[i].nbr_fils;
 Tab_themes_b[i].select=False;
}
if(param.scmenus==1)
  while((sel==2)&&(!fin_jour))
  {
   sel=ecran3();
   if(sel==2)
   {
    marquer_jour_traite(*date);
    EnregistreMenu(*date, 0);
    jour_suivant_a_traiter(&jour, &fin_jour);
   }
  }
else
{
 sel=ecran3b();
 date=&calendrier[sel-1];
}
if((sel!=0)&&(!fin_jour))
{
 nb_pers=ecran4(0,True);
 while(vide)
 {
  while((nb_theme=ecran5(&CouleurTheme))!=9999)
     if(nb_theme<10)
       TraitePlat(nb_theme);
     else
       TraiteMenu();
  if(MenuVide())
    vide=!ConfirmationOuiNon(tab_msg[32]);
  else vide=False;
  RaffraichitEGadget();
 }
 marquer_jour_traite(*date);
 EnregistreMenu(*date,nb_pers);
 while((choix=ecran8(True))!=0)
      switch(choix)
      {
       case(1) : ConsultationMenu(date);break;
       case(2) : ImpressionMenu(date);break;
       case(3) : ConsultationProduit(Tab_Plat_Theme);break;
       case(4) : ImpressionProduit(Tab_Plat_Theme, *date);break;
      }
}
```

```
if((fin_jour)||(sel==0))
 return 0;
else
 return sel;
}
```

```
/*
 * Module   : ConsultationMenu
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ecran.h"
PUBLIC short calendrier_long;
PUBLIC short themes_long;
PUBLIC short plat_long;
PUBLIC tparamutil param;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC ttheme_pere_fils Tab_themes[maxthemes];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme*(maxfilstheme+1)];
PUBLIC tmenu_enreg Menu_enreg;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC st_msg tab_msg[nbre_msg];
PRIVATE UBYTE CouleurTheme;
PRIVATE tplatstheme Tab_Plat_Theme_bis[maxthemes];
PRIVATE tconstituants_ext tab_const[maxthemes];

PRIVATE USHORT tab_ptr(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes_b[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return i;
}

PRIVATE void ChargeMenu(date,nbr_pers)
tdate date;
short *nbr_pers;
{
 int j=0, k, i, l;
 short nbr_const;
 boolean existe;
 for(i=0;i<maxthemes;i++)
    Tab_Plat_Theme_bis[i]=Tab_Plat_Theme[i];
 fournir_menu_complet_sur_clel(date,0,nbr_pers,tab_const,&nbr_const,
                               Tab_plats,&plat_long,&existe);
 for(i=0;i<nbr_const;i++)
 {
  j=tab_const[i].ptr_plat;
  l=tab_const[i].priorite;
  Tab_themes_b[l].select=True;
  strcpy(Tab_Plat_Theme[l].fichier,"plat");
  strcat(Tab_Plat_Theme[l].fichier,itoa(Tab_plats[j].no_plat));
  strcat(Tab_Plat_Theme[l].fichier,".pic");
  strcpy(Tab_Plat_Theme[l].nom,"");
  strcpy(Tab_Plat_Theme[l].son,"");
  Tab_Plat_Theme[l].num_plat=Tab_plats[j].no_plat;
  Tab_Plat_Theme[l].couleur_theme=tab_const[i].couleur;
  Tab_Plat_Theme[l].num_theme=tab_const[i].no_theme;
```

```
  Tab_Plat_Theme[l].decomp= (Tab_plats[j].nbr_fils!=0) ? True : False;
  if(!Tab_Plat_Theme[l].decomp)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j].nom);
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j].son);
  }
  for(k=0;k<Tab_plats[j].nbr_fils;k++)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j+k+1].nom);
   strcat(Tab_Plat_Theme[l].nom,"#");
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j+k+1].son);
   strcat(Tab_Plat_Theme[l].son,",");
  }
  Tab_Plat_Theme_bis[l]=Tab_Plat_Theme[l];
 }
}

PRIVATE void TraiteMenu()
{
 int i;
 ecran7(False);
 for(i=0;i<maxthemes;i++)
  Tab_themes_b[i].select=eval(Tab_Plat_Theme[i].nom,"=","") ? False : True;
}

PRIVATE boolean MenuVide()
{
 int i;
 boolean retval=True;
 for(i=0;i<maxthemes;i++)
    retval=(Tab_Plat_Theme[i].num_plat==-1) ? retval : False;
 return retval;
}

USHORT ConsultationMenu(daterec)
tdate *daterec;
{
 USHORT nb_pers, i, choix;
 USHORT sel=2;
 tdatenom jour;
 tdate *date;
 boolean OK=False;
 Menu_enreg.nbrconst=0;
 date=(tdate *) &jour;
 for(i=0;i<calendrier_long;i++)
    if(calendrier[i].traite) OK=True;
 if(OK)
 {
  for(i=0;i<maxthemes;i++)
  {
   strcpy(Tab_Plat_Theme[i].nom,"");
   strcpy(Tab_Plat_Theme[i].son,"");
   Tab_Plat_Theme[i].decomp=False;
   Tab_Plat_Theme[i].num_plat=-1;
   Tab_Plat_Theme[i].num_theme=0;
   Tab_Plat_Theme[i].couleur_theme=0;
  }
  for(i=0;i<themes_long;i++)
  {
```

```
   strcpy(Tab_themes_b[i].nom,Tab_themes[i].nom);
   strcpy(Tab_themes_b[i].son,Tab_themes[i].son);
   Tab_themes_b[i].no_theme=Tab_themes[i].no_theme;
   Tab_themes_b[i].article=Tab_themes[i].article;
   Tab_themes_b[i].priorite=Tab_themes[i].priorite;
   Tab_themes_b[i].couleur=Tab_themes[i].couleur;
   Tab_themes_b[i].nbr_fils=Tab_themes[i].nbr_fils;
   Tab_themes_b[i].select=False;
  }
  if(daterec==NULL)
    sel=ecran3c();
  if(sel!=0)
  {
   date=(daterec==NULL) ? &calendrier[sel-1] : daterec;
   ChargeMenu(*date,&nb_pers);
   if(nb_pers==0)
     ConfirmationOK(tab_msg[26]);
   else
   {
    ecran4(nb_pers,False);
    TraiteMenu();
   }
  while((choix=ecran8(True))!=0)
       switch(choix)
       {
        case(1) : ConsultationMenu(date);break;
        case(2) : ImpressionMenu(date);break;
        case(3) : ConsultationProduit(Tab_Plat_Theme);break;
        case(4) : ImpressionProduit(Tab_Plat_Theme, *date);break;
       }
  }
  if(sel==0)
   return 0;
  else
   return sel;
 }
 else
 {
  ConfirmationOK(tab_msg[25]);
  return 0;
 }
}
```

```
/*
 * Module   : ModificationMenu
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */


#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ecran.h"
PUBLIC short calendrier_long;
PUBLIC short themes_long;
PUBLIC short plat_long;
PUBLIC tparamutil param;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC ttheme_pere_fils Tab_themes[maxthemes];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme*(maxfilstheme+1)];
PUBLIC tmenu_enreg Menu_enreg;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC st_msg tab_msg[nbre_msg];
PRIVATE UBYTE CouleurTheme;
PRIVATE tplatstheme Tab_Plat_Theme_bis[maxthemes];
PRIVATE tconstituants_ext tab_const[maxthemes];

/***********************************************/
PRIVATE void jour_suivant_a_traiter(jour_suivant,fin_jour)
/***********************************************/

tdatenom *jour_suivant;
boolean *fin_jour;

{
   boolean trouve;
   short i;

   *fin_jour=True;
   trouve=False;
   i=0;

   while ((i<calendrier_long) && (trouve==False))
        {
          if (calendrier[i].traite==False)
             {
                jour_suivant->annee=calendrier[i].annee;
                jour_suivant->mois=calendrier[i].mois;
                jour_suivant->jour=calendrier[i].jour;
                jour_suivant->nom=calendrier[i].nom;
                *fin_jour=False;
                trouve=True;
             }
             else ++i;
        }
}

/*************************************/
PRIVATE void marquer_jour_traite(jour)
```

```
/************************************/

tdate jour;

{
   tdate date_temp;
   boolean marque;
   short i;

   i=0;
   marque=False;
   while (marque==False)
        {
           date_temp.annee=calendrier[i].annee;
           date_temp.mois=calendrier[i].mois;
           date_temp.jour=calendrier[i].jour;
           if (compare_date(date_temp,jour,'=')==True)
              {
                 calendrier[i].traite=True;
                 marque=True;
              }
              else ++i;
        }
}

PRIVATE void InitMenu_enreg(void)
{
 int i;
 for(i=0;i<maxthememenu;i++)
    Menu_enreg.constituants[i].plat=-1;
}

PRIVATE USHORT tab_ptr(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes_b[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return i;
}

PRIVATE void ChargeMenu(date,nbr_pers)
tdate date;
short *nbr_pers;
{
 int j=0, k, i, l;
 short nbr_const;
 boolean existe;
 for(i=0;i<maxthemes;i++)
    Tab_Plat_Theme_bis[i]=Tab_Plat_Theme[i];
 fournir_menu_complet_sur_clel(date,0,nbr_pers,tab_const,&nbr_const,
                               Tab_plats,&plat_long,&existe);
 for(i=0;i<nbr_const;i++)
 {
  j=tab_const[i].ptr_plat;
  l=tab_const[i].priorite;
  Tab_themes_b[l].select=True;
  strcpy(Tab_Plat_Theme[l].fichier,"plat");
  strcat(Tab_Plat_Theme[l].fichier,itoa(Tab_plats[j].no_plat));
```

```
  strcat(Tab_Plat_Theme[l].fichier,".pic");
  strcpy(Tab_Plat_Theme[l].nom,"");
  strcpy(Tab_Plat_Theme[l].son,"");
  Tab_Plat_Theme[l].num_plat=Tab_plats[j].no_plat;
  Tab_Plat_Theme[l].couleur_theme=tab_const[i].couleur;
  Tab_Plat_Theme[l].num_theme=tab_const[i].no_theme;
  Tab_Plat_Theme[l].decomp= (Tab_plats[j].nbr_fils!=0) ? True : False;
  if(!Tab_Plat_Theme[l].decomp)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j].nom);
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j].son);
  }
  for(k=0;k<Tab_plats[j].nbr_fils;k++)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j+k+1].nom);
   strcat(Tab_Plat_Theme[l].nom,"#");
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j+k+1].son);
   strcat(Tab_Plat_Theme[l].son,",");
  }
  Tab_Plat_Theme_bis[l]=Tab_Plat_Theme[l];
 }
}

PRIVATE void TraitePlat(num_theme)
USHORT num_theme;
{
 int i;
 for(i=0;i<maxplatstheme;i++)
    strcpy(Tab_plats[i].nom,"\0");
 fournir_plats_et_fils_sur_cle1(num_theme,Tab_plats,&plat_long);
 if(Menu_enreg.constituants[num_theme].plat==-1)
   ++Menu_enreg.nbrconst;
 ecran6(num_theme, CouleurTheme);
}

PRIVATE void TraiteMenu()
{
 int i;
 ecran7(True);
 for(i=0;i<maxthemes;i++)
  Tab_themes_b[i].select=eval(Tab_Plat_Theme[i].nom,"=","") ? False : True;
}

PRIVATE void CreerListeModif(liste_ajoute, liste_enleve,
                             size_ajoute, size_enleve)
tconstituants_enreg liste_ajoute[], liste_enleve[];
short *size_ajoute, *size_enleve;
{
 int i;
 *size_enleve=0;
 *size_ajoute=0;
 for(i=0;i<maxthemes;i++)
     if(Tab_Plat_Theme[i].num_plat!=Tab_Plat_Theme_bis[i].num_plat)
     {
      if(Tab_Plat_Theme[i].num_plat!=-1)
      {
       liste_ajoute[*size_ajoute].plat=Tab_Plat_Theme[i].num_plat;
       liste_ajoute[*size_ajoute].theme=Tab_Plat_Theme[i].num_theme;
       ++*size_ajoute;
```

```
      }
      if(Tab_Plat_Theme_bis[i].num_plat!=-1)
      {
       liste_enleve[*size_enleve].plat=Tab_Plat_Theme_bis[i].num_plat;
       liste_enleve[*size_enleve].theme=Tab_Plat_Theme_bis[i].num_theme;
       ++*size_enleve;
      }
     }
 printf("nbre ajoute %d nombre supprime %d\n",*size_ajoute, *size_enleve);
 for(i=0;i<*size_ajoute;i++)
    printf("plat ajoute no %d plat no %d theme no %d\n",i,
            liste_ajoute[i].plat,liste_ajoute[i].theme);
 for(i=0;i<*size_enleve;i++)
    printf("plat enleve no %d plat no %d theme no %d\n",i,
            liste_enleve[i].plat,liste_enleve[i].theme);
}

PRIVATE void EnregistreModifMenu(date, nb_pers)
tdate date;
USHORT nb_pers;
{
 tconstituants_enreg liste_ajoute[maxthemes], liste_enleve[maxthemes];
 short size_ajoute, size_enleve;
 CreerListeModif(liste_ajoute, liste_enleve, &size_ajoute, &size_enleve);
 modifier_menu_complet(date,0,nb_pers,
                       liste_enleve, size_enleve,
                       liste_ajoute, size_ajoute);
}

PRIVATE boolean MenuVide()
{
 int i;
 boolean retval=True;
 for(i=0;i<maxthemes;i++)
    retval=(Tab_Plat_Theme[i].num_plat==-1) ? retval : False;
 return retval;
}

USHORT ModificationMenu(void)
{
 USHORT nb_theme, nb_pers, i;
 USHORT sel=2, choix;
 tdatenom jour;
 tdate *date;
 boolean OK=False, vide=True;
 Menu_enreg.nbrconst=0;
 date=(tdate *) &jour;
 for(i=0;i<calendrier_long;i++)
    if(calendrier[i].traite) OK=True;
 if(OK)
 {
  InitMenu_enreg();
  for(i=0;i<maxthemes;i++)
  {
   strcpy(Tab_Plat_Theme[i].nom,"");
   strcpy(Tab_Plat_Theme[i].son,"");
   Tab_Plat_Theme[i].decomp=False;
   Tab_Plat_Theme[i].num_plat=-1;
   Tab_Plat_Theme[i].num_theme=0;
```

```
 Tab_Plat_Theme[i].couleur_theme=0;
}
for(i=0;i<themes_long;i++)
{
 strcpy(Tab_themes_b[i].nom,Tab_themes[i].nom);
 strcpy(Tab_themes_b[i].son,Tab_themes[i].son);
 Tab_themes_b[i].no_theme=Tab_themes[i].no_theme;
 Tab_themes_b[i].article=Tab_themes[i].article;
 Tab_themes_b[i].priorite=Tab_themes[i].priorite;
 Tab_themes_b[i].couleur=Tab_themes[i].couleur;
 Tab_themes_b[i].nbr_fils=Tab_themes[i].nbr_fils;
 Tab_themes_b[i].select=False;
}
sel=ecran3c();
if(sel!=0)
{
 date=&calendrier[sel-1];
 ChargeMenu(*date,&nb_pers);
 nb_pers=ecran4(nb_pers,True);
 TraiteMenu();
 while(vide)
 {
  while((nb_theme=ecran5(&CouleurTheme))!=9999)
     if(nb_theme<10)
       TraitePlat(nb_theme);
     else
       TraiteMenu();
  if(MenuVide())
    vide=!ConfirmationOuiNon(tab_msg[32]);
  else vide=False;
  RaffraichitEGadget();
 }
 marquer_jour_traite(*date);
 EnregistreModifMenu(*date,nb_pers);
 while((choix=ecran8(True))!=0)
      switch(choix)
      {
       case(1) : ConsultationMenu(date);break;
       case(2) : ImpressionMenu(date);break;
       case(3) : ConsultationProduit(Tab_Plat_Theme);break;
       case(4) : ImpressionProduit(Tab_Plat_Theme, *date);break;
      }
}
if(sel==0)
 return 0;
else
 return sel;
}
else
{
 ConfirmationOK(tab_msg[25]);
 return 0;
}
}
```

```
/*
 * Module   : ImpressionMenu
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include <exec/types.h>
#include <exec/memory.h>
#include "H_String.h"
#include "H_Gadget.h"
#include "H_Types.h"
#include "H_CTypes.h"
#include "H_Types2.h"
#include "H_Ecran.h"
PUBLIC short calendrier_long;
PUBLIC short themes_long;
PUBLIC short plat_long;
PUBLIC tparamutil param;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC ttheme_pere_fils Tab_themes[maxthemes];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme*(maxfilstheme+1)];
PUBLIC tmenu_enreg Menu_enreg;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct TextAttr TextAttr;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC struct TextFont *TextFont;

PRIVATE UBYTE CouleurTheme;
PRIVATE tplatstheme Tab_Plat_Theme_bis[maxthemes];
PRIVATE tconstituants_ext tab_const[maxthemes];
PRIVATE struct RastPort rport;
PRIVATE struct BitMap Bitmap;
PRIVATE UBYTE *plans[5];
PRIVATE char sentence[256], *token;

PRIVATE USHORT tab_ptr(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes_b[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return i;
}

PRIVATE void ChargeMenu(date,nbr_pers)
tdate date;
short *nbr_pers;
{
 int j=0, k, i, l;
 short nbr_const;
 boolean existe;
 for(i=0;i<maxthemes;i++)
    Tab_Plat_Theme_bis[i]=Tab_Plat_Theme[i];
 fournir_menu_complet_sur_clel(date,0,nbr_pers,tab_const,&nbr_const,
                               Tab_plats,&plat_long,&existe);
```

```
 for(i=0;i<nbr_const;i++)
 {
  j=tab_const[i].ptr_plat;
  l=tab_const[i].priorite;
  Tab_themes_b[l].select=True;
  strcpy(Tab_Plat_Theme[l].fichier,"plat");
  strcat(Tab_Plat_Theme[l].fichier,itoa(Tab_plats[j].no_plat));
  strcat(Tab_Plat_Theme[l].fichier,".pic");
  strcpy(Tab_Plat_Theme[l].nom,"");
  strcpy(Tab_Plat_Theme[l].son,"");
  Tab_Plat_Theme[l].num_plat=Tab_plats[j].no_plat;
  Tab_Plat_Theme[l].couleur_theme=tab_const[i].couleur;
  Tab_Plat_Theme[l].num_theme=tab_const[i].no_theme;
  Tab_Plat_Theme[l].decomp= (Tab_plats[j].nbr_fils!=0) ? True : False;
  if(!Tab_Plat_Theme[l].decomp)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j].nom);
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j].son);
  }
  for(k=0;k<Tab_plats[j].nbr_fils;k++)
  {
   strcat(Tab_Plat_Theme[l].nom,Tab_plats[j+k+1].nom);
   strcat(Tab_Plat_Theme[l].nom,"#");
   strcat(Tab_Plat_Theme[l].son,Tab_plats[j+k+1].son);
   strcat(Tab_Plat_Theme[l].son,",");
  }
  Tab_Plat_Theme_bis[l]=Tab_Plat_Theme[l];
 }
}

PRIVATE void viderbitmap()
{
 int i,j;
 for(i=0;i<5;i++)
 {
  for(j=0;j<600;j++)
     Bitmap.Planes[i][j]=((i==4)||(i==3)||(i==1)) ? 0xFF : 0;
 }
}

PRIVATE short JourDate(date)
tdate date;
{
 int calend;
 calend=((date.annee-1)*365.25)+date.jour;
 calend+=(date.mois<8) ? (((date.mois+1)/2)*31)+((date.mois/2)*30)
                       : (((date.mois+2)/2)*31)+(((date.mois-1)/2)*30);
 calend+=(((date.annee%4)==0)&&(date.mois>1)) ? 1 : 0;
 calend-=(date.mois<2) ? 0 : 2;
 return ((short) (calend-1)%7);
}

PRIVATE void PrintTexte(rport,phrase,x,y,textfont)
struct RastPort *rport;
char *phrase;
USHORT x, y;
struct TextFont *textfont;
{
 struct TextFont *oldfont=rport->Font;
```

```
 if(textfont!=NULL)
    SetFont(rport,textfont);
 Move(rport,x,y+rport->TxBaseline);
 SetAPen(rport,1);
 SetBPen(rport,26);
 SetDrMd(rport,JAM1);
 if(!eval(phrase,"=",""))
   Text(rport,phrase,strlen(phrase));
 print(rport,0,0,320,15);
 SetFont(rport,oldfont);
}

PRIVATE void PrintLigne(phrase, x, y, textfont)
char *phrase;
USHORT x, y;
struct TextFont *textfont;
{
 viderbitmap();
 PrintTexte(&rport,phrase,x,y,textfont);
}

PRIVATE void TraiteMenu(npers, date)
short npers;
tdate date;
{
 int i, lng;
 for(i=0;i<maxthemes;i++)
  Tab_themes_b[i].select=eval(Tab_Plat_Theme[i].nom,"=","") ? False : True;
 strcpy(sentence,"MENU du ");
 strcat(sentence,tab_nomjours[JourDate(date)].nom);
 strcat(sentence," ");
 strcat(sentence,itoa(date.jour));
 strcat(sentence," ");
 strcat(sentence,tab_nommois[date.mois].nom);
 PrintLigne(sentence,50,0,NULL);
 lng=strlen(sentence);
 for(i=0;i<lng;i++)
    sentence[i]='-';
 PrintLigne(sentence,50,0,NULL);
 PrintLigne("",0,0,NULL);
 strcpy(sentence,"Nombre de personnes : ");
 strcat(sentence,itoa(npers));
 strcat(sentence,"     ");
 for(i=0;i<npers;i++)
    strcat(sentence,"| ");
 PrintLigne(sentence,0,0,NULL);
 for(i=0;i<maxthemes;i++)
 {
  if(Tab_themes_b[i].select)
  {
   PrintLigne("",0,0,NULL);
   strcpy(sentence,Tab_themes_b[i].nom);
   PrintLigne(sentence,0,0,NULL);
   if(!Tab_Plat_Theme[Tab_themes_b[i].priorite].decomp)
   {
    token=(char *)strtok(Tab_Plat_Theme[Tab_themes_b[i].priorite].nom,"#");
    while(token!=NULL)
    {
     PrintLigne(token,20,0,TextFont);
```

```
     token=(char *)strtok(NULL,"#");
    }
   }
   else
   {
    strcpy(sentence,Tab_Plat_Theme[Tab_themes_b[i].priorite].nom);
    PrintLigne(sentence,20,0,TextFont);
   }
  }
 }
 PrintLigne("",0,0,NULL);
}

PRIVATE boolean MenuVide()
{
 int i;
 boolean retval=True;
 for(i=0;i<maxthemes;i++)
    retval=(Tab_Plat_Theme[i].num_plat==-1) ? retval : False;
 return retval;
}

USHORT ImpressionMenu(daterec)
tdate *daterec;
{
 USHORT nb_pers, i;
 USHORT sel=2;
 struct RastPort *rp;
 struct BitMap *bm;
 tdatenom jour;
 tdate *date;
 boolean OK=False;
 rp=Window->RPort;
 bm=rp->BitMap;
 rport = *rp;
 Bitmap = *bm;
 Bitmap.BytesPerRow=40;
 Bitmap.Rows=15;
 Bitmap.Depth=5;
 rport.BitMap=&Bitmap;
 Menu_enreg.nbrconst=0;
 date=(tdate *) &jour;
 for(i=0;i<5;i++)
 {
  plans[i]=(UBYTE *) AllocMem(600,MEMF_CHIP|MEMF_CLEAR);
  Bitmap.Planes[i]=plans[i];
 }
 for(i=0;i<calendrier_long;i++)
    if(calendrier[i].traite) OK=True;
 if(OK)
 {
  for(i=0;i<maxthemes;i++)
  {
   strcpy(Tab_Plat_Theme[i].nom,"");
   strcpy(Tab_Plat_Theme[i].son,"");
   Tab_Plat_Theme[i].decomp=False;
   Tab_Plat_Theme[i].num_plat=-1;
   Tab_Plat_Theme[i].num_theme=0;
   Tab_Plat_Theme[i].couleur_theme=0;
```

```
 }
 for(i=0;i<themes_long;i++)
 {
  strcpy(Tab_themes_b[i].nom,Tab_themes[i].nom);
  strcpy(Tab_themes_b[i].son,Tab_themes[i].son);
  Tab_themes_b[i].no_theme=Tab_themes[i].no_theme;
  Tab_themes_b[i].article=Tab_themes[i].article;
  Tab_themes_b[i].priorite=Tab_themes[i].priorite;
  Tab_themes_b[i].couleur=Tab_themes[i].couleur;
  Tab_themes_b[i].nbr_fils=Tab_themes[i].nbr_fils;
  Tab_themes_b[i].select=False;
 }
 if(daterec==NULL)
   sel=ecran3c();
 if(sel!=0)
 {
  date=(daterec==NULL) ? &calendrier[sel-1] : daterec;
  ChargeMenu(*date,&nb_pers);
  if(nb_pers==0)
    ConfirmationOK(tab_msg[26]);
  else
  {
   TraiteMenu(nb_pers, *date);
  }
 }
 if(sel==0)
  return 0;
 else
  return sel;
}
else
{
 ConfirmationOK(tab_msg[25]);
 return 0;
}
for(i=0;i<5;i++)
    FreeMem(plans[i],600);

}
```

```
/*
 * Module   : ConsultationProduit
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "stdio.h"
#include "H_Types2.h"
PUBLIC tproduitcompos Tab_Produit[maxthememenu*maxproduitsplat];

void ConsultationProduit(liste_plats)
tplatstheme liste_plats[];
{
 short i, nbre_plats=0, tab_plats[maxthememenu], nbre_produits;
 for(i=0;i<maxthemes;i++)
 {
  if(liste_plats[i].num_plat!=-1)
     tab_plats[nbre_plats++]=liste_plats[i].num_plat;
 }
 fournir_produits_menu(tab_plats,nbre_plats,Tab_Produit,&nbre_produits);
 ecran9(nbre_produits);
}
```

```
/*
 * Module   : ImpressionProduit
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include <exec/types.h>
#include <exec/memory.h>
#include "H_String.h"
#include "stdio.h"
#include "H_Types2.h"
#include "H_Image.h"
PUBLIC tproduitcompos Tab_Produit[maxthememenu*maxproduitsplat];
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];

PRIVATE struct RastPort rport;
PRIVATE struct BitMap Bitmap;
PRIVATE UBYTE *plans[5];
PRIVATE char sentence[256];
PRIVATE struct Image bliss;

PRIVATE void viderbitmap()
{
 int i,j;
 for(i=0;i<5;i++)
 {
  for(j=0;j<2400;j++)
     Bitmap.Planes[i][j]=((i==4)||(i==3)||(i==1)) ? 0xFF : 0;
 }
}

PRIVATE short JourDate(date)
tdate date;
{
 int calend;
 calend=((date.annee-1)*365.25)+date.jour;
 calend+=(date.mois<8) ? (((date.mois+1)/2)*31)+((date.mois/2)*30)
                       : (((date.mois+2)/2)*31)+(((date.mois-1)/2)*30);
 calend+=(((date.annee%4)==0)&&(date.mois>1)) ? 1 : 0;
 calend-=(date.mois<2) ? 0 : 2;
 return ((short) (calend-1)%7);
}

PRIVATE void PrintTexte(rport,phrase,x,y,textfont)
struct RastPort *rport;
char *phrase;
USHORT x, y;
struct TextFont *textfont;
{
 struct TextFont *oldfont=rport->Font;
 if(textfont!=NULL)
    SetFont(rport,textfont);
 Move(rport,x,y+rport->TxBaseline);
 SetAPen(rport,1);
 SetBPen(rport,26);
 SetDrMd(rport,JAM1);
 if(!eval(phrase,"=",""))
```

```
   Text(rport,phrase,strlen(phrase));
 SetFont(rport,oldfont);
}

PRIVATE void AfficheBliss(picto)
struct Image *picto;
{
 Move(&rport,0,0);
 DrawImage(&rport,picto,0,0);
 print(&rport,0,0,320,60);
}

PRIVATE void PrintLigne(phrase, x, y, textfont, doprint)
char *phrase;
USHORT x, y;
struct TextFont *textfont;
boolean doprint;
{
 if(doprint)
   viderbitmap();
 PrintTexte(&rport,phrase,x,y,textfont);
 if(doprint)
   print(&rport,0,0,320,12);
}

void ImpressionProduit(liste_plats,date)
tplatstheme liste_plats[];
tdate date;
{
 short i, j, nbre_plats=0, tab_plats[maxthememenu], nbre_produits, lng;
 struct RastPort *rp;
 struct BitMap *bm;
 rp=Window->RPort;
 bm=rp->BitMap;
 rport = *rp;
 Bitmap = *bm;
 Bitmap.BytesPerRow=40;
 Bitmap.Rows=60;
 Bitmap.Depth=5;
 rport.BitMap=&Bitmap;
 for(i=0;i<5;i++)
 {
  plans[i]=(UBYTE *) AllocMem(2400,MEMF_CHIP|MEMF_CLEAR);
  Bitmap.Planes[i]=plans[i];
 }
 for(i=0;i<maxthemes;i++)
 {
  if(liste_plats[i].num_plat!=-1)
     tab_plats[nbre_plats++]=liste_plats[i].num_plat;
 }
 fournir_produits_menu(tab_plats,nbre_plats,Tab_Produit,&nbre_produits);
 if(nbre_produits>0)
 {
  strcpy(sentence,"Menu du ");
  strcat(sentence,tab_nomjours[JourDate(date)].nom);
  strcat(sentence," ");
  strcat(sentence,itoa(date.jour));
  strcat(sentence," ");
  strcat(sentence,tab_nommois[date.mois].nom);
```

```
  PrintLigne(sentence,45,0,NULL,True);
  lng=strlen(sentence);
  strcpy(sentence,"(liste des ingrédients)");
  PrintLigne(sentence,45,0,NULL,True);
  for(i=0;i<lng;i++)
     sentence[i]='-';
  sentence[i]='\0';
  PrintLigne(sentence,45,0,NULL,True);
  PrintLigne("",0,0,NULL,True);
  for(i=0;i<nbre_produits;i++)
  {
   viderbitmap();
   strcpy(sentence,Tab_Produit[i].nom);
   PrintLigne(sentence,100,25,NULL,False);
   strcpy(sentence,"Produit");
   strcat(sentence,itoa(Tab_Produit[i].no_produit));
   strcat(sentence,".bliss");
   ChargeImage(sentence,&bliss);
   AfficheBliss(&bliss);
   LibereImage(&bliss);
   if((i%8==0)&&(i!=8)&&(i!=0))
     for(j=0;j<4;j++)
        PrintLigne("",0,0,NULL,True);
  }
 }
 for(i=0;i<5;i++)
     FreeMem(plans[i],2400);

}
```

```
/*
 * Module   : ListeDeCourses
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include <exec/types.h>
#include <exec/memory.h>
#include "H_String.h"
#include "H_Gadget.h"
#include "H_Types.h"
#include "H_CTypes.h"
#include "H_Types2.h"
#include <math.h>

typedef struct
{
 short qte;
 short num_produit;
 tnom_produit nom;
 short article;
 tnom_produit bliss;
 short cond_vente;
 short prix;
} tproduitliste;

PUBLIC short calendrier_long;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct TextFont *TextFont;
PUBLIC tarticle Tab_Article[12];
PUBLIC tconditionnement Tab_Cond[10];
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];

PRIVATE struct RastPort rport;
PRIVATE struct BitMap Bitmap;
PRIVATE UBYTE *plans[5];
PRIVATE char sentence[256], *token;
PRIVATE struct Image bliss;

static tproduitliste liste_produit_a_acheter[maxproduits];
static short nbre_prod;

PRIVATE short TriSurClasseStock(liste_produit, nbre_elem)
tproduitcompos *liste_produit;
short nbre_elem;
{
 short i,j,up,down;
 boolean sens=True;
 tproduitcompos produit;
 for(i=0;((liste_produit[i].classe_stock==0)&&(i<nbre_elem));i++);
 produit=liste_produit[i];
 down=i+1;
 up=nbre_elem-1;
 while(down<=up)
    if(sens)
      for(j=up;(down<=up)&&(sens);j--)
```

```
      {
       if(liste_produit[j].classe_stock<produit.classe_stock)
       {
        liste_produit[i]=liste_produit[j];
        i=j;
        sens=False;
       }
       --up;
      }
    else
      for(j=down;(down<=up)&&(!sens);j++)
      {
       if(liste_produit[j].classe_stock>=produit.classe_stock)
       {
        liste_produit[i]=liste_produit[j];
        i=j;
        sens=True;
       }
       ++down;
      }
 liste_produit[i]=produit;
 return i;
}

PRIVATE void viderbitmap()
{
 int i,j;
 for(i=0;i<5;i++)
 {
  for(j=0;j<2400;j++)
     Bitmap.Planes[i][j]=((i==4)||(i==3)||(i==1)) ? 0xFF : 0;
 }
}

PRIVATE short JourDate(date)
tdate date;
{
 int calend;
 calend=((date.annee-1)*365.25)+date.jour;
 calend+=(date.mois<8) ? (((date.mois+1)/2)*31)+((date.mois/2)*30)
                       : (((date.mois+2)/2)*31)+(((date.mois-1)/2)*30);
 calend+=(((date.annee%4)==0)&&(date.mois>1)) ? 1 : 0;
 calend-=(date.mois<2) ? 0 : 2;
 return ((short) (calend-1)%7);
}

PRIVATE void PrintTexte(rport,phrase,x,y,textfont)
struct RastPort *rport;
char *phrase;
USHORT x, y;
struct TextFont *textfont;
{
 struct TextFont *oldfont=rport->Font;
 if(textfont!=NULL)
    SetFont(rport,textfont);
 Move(rport,x,y+rport->TxBaseline);
 SetAPen(rport,1);
 SetBPen(rport,26);
 SetDrMd(rport,JAM1);
```

```
 if(!eval(phrase,"=",""))
   Text(rport,phrase,strlen(phrase));
 SetFont(rport,oldfont);
}

PRIVATE void AfficheBliss(picto)
struct Image *picto;
{
 Move(&rport,0,0);
 DrawImage(&rport,picto,0,0);
 print(&rport,0,0,320,60);
}

PRIVATE void PrintLigne(phrase, x, y, textfont, doprint)
char *phrase;
USHORT x, y;
struct TextFont *textfont;
boolean doprint;
{
 if(doprint)
   viderbitmap();
 PrintTexte(&rport,phrase,x,y,textfont);
 if(doprint)
   print(&rport,0,0,320,12);
}

PRIVATE void AjouteProduitAAcheter(produit,quantite)
tproduitcompos produit;
short quantite;
{
 liste_produit_a_acheter[nbre_prod].qte=
   ((produit.qte_cond_util_cond_vente!=0)&&(produit.classe_stock==0)) ?
     ((quantite+produit.qte_cond_util_cond_vente-1)/produit.qte_cond_util_cond_vente) :
          ((produit.classe_stock==1) ? 1 : quantite);
 liste_produit_a_acheter[nbre_prod].num_produit=produit.no_produit;
 liste_produit_a_acheter[nbre_prod].article=produit.article;
 strcpy(liste_produit_a_acheter[nbre_prod].nom,produit.nom);
 liste_produit_a_acheter[nbre_prod].cond_vente=produit.cond_vente;
 liste_produit_a_acheter[nbre_prod].prix=produit.prix;
 strcpy(liste_produit_a_acheter[nbre_prod].bliss,"Produit");
 strcat(liste_produit_a_acheter[nbre_prod].bliss,itoa(produit.no_produit));
 strcat(liste_produit_a_acheter[nbre_prod].bliss,".bliss");
 nbre_prod++;
}

PRIVATE void ListeProduitsAAcheter(liste_produit,nbre_elem,periode)
tproduitcompos *liste_produit;
short nbre_elem;
tperiode_achat periode[max_periode_a_traiter];
{
 short i, j, premier_qte=TriSurClasseStock(liste_produit, nbre_elem);
 short qteenstock, qtenec, lng;
 char string[128];
 if(premier_qte!=nbre_elem)
   for(i=premier_qte;i<nbre_elem;i++)
     if(!ecran11(liste_produit[i]))
       AjouteProduitAAcheter(liste_produit[i],1);
 if(premier_qte!=0)
   for(i=0;i<premier_qte;i++)
```

```
  {
   qtenec=dtoi(liste_produit[i].qtecompos);
   if(liste_produit[i].qtecompos!=dtoi(liste_produit[i].qtecompos))
     qtenec+=1.;
   if((qteenstock=ecran10(liste_produit[i]))<qtenec)
     AjouteProduitAAcheter(liste_produit[i],qtenec-qteenstock);
   printf("quantites %d quantiten %d produit %s\n",qteenstock,qtenec,liste_produit[i].nom);
  }
if(nbre_prod!=0)
{
 strcpy(sentence,"Liste du ");
 strcat(sentence,tab_nomjours[JourDate(periode[0].debut)].nom);
 strcat(sentence," ");
 strcat(sentence,itoa(periode[0].debut.jour));
 strcat(sentence," ");
 strcat(sentence,tab_nommois[periode[0].debut.mois].nom);
 PrintLigne(sentence,45,0,NULL,True);
 lng=strlen(sentence);
 strcpy(sentence,"au ");
 strcat(sentence,tab_nomjours[JourDate(periode[0].fin)].nom);
 strcat(sentence," ");
 strcat(sentence,itoa(periode[0].fin.jour));
 strcat(sentence," ");
 strcat(sentence,tab_nommois[periode[0].fin.mois].nom);
 PrintLigne(sentence,93,0,NULL,True);
 for(i=0;i<lng;i++)
    sentence[i]='-';
 sentence[i]='\0';
 PrintLigne(sentence,45,0,NULL,True);
 PrintLigne("",0,0,NULL,True);
 for(i=0;i<nbre_prod;i++)
 {
  viderbitmap();
  printf("%s : %d --> %d",liste_produit_a_acheter[i].nom,
                          liste_produit_a_acheter[i].num_produit,
                          dtoi(liste_produit_a_acheter[i].qte,3));
  strcpy(sentence,itoa(liste_produit_a_acheter[i].qte));
  strcat(sentence," ");
  if(liste_produit_a_acheter[i].cond_vente!=0)
  {
   if(liste_produit_a_acheter[i].qte>1)
     strcat(sentence,
            pluriel(string,
                    Tab_Cond[liste_produit_a_acheter[i].cond_vente].nom));
   else
      strcat(sentence,Tab_Cond[liste_produit_a_acheter[i].cond_vente].nom);
   strcat(sentence," ");
   strcat(sentence,Tab_Article[liste_produit_a_acheter[i].article].nom);
   strcat(sentence," ");
  }
  if((liste_produit_a_acheter[i].qte>1)&&
     (liste_produit_a_acheter[i].cond_vente==0))
    strcat(sentence,pluriel(string,liste_produit_a_acheter[i].nom));
  else
    strcat(sentence,liste_produit_a_acheter[i].nom);
  PrintLigne(sentence,100,10,NULL,False);
  printf("%s",liste_produit_a_acheter[i].bliss);
  strcpy(sentence,"");
  if(liste_produit_a_acheter[i].qte<max_quantite_affiche)
```

```
    for(j=0;j<liste_produit_a_acheter[i].qte;j++)
       strcat(sentence,"| ");
   ChargeImage(liste_produit_a_acheter[i].bliss,&bliss);
   PrintLigne(sentence,100,30,NULL,False);
   AfficheBliss(&bliss);
   LibereImage(&bliss);
   if((i%8==0)&&(i!=8)&&(i!=0))
     for(j=0;j<4;j++)
        PrintLigne("",0,0,NULL,True);
  }
 }
}

USHORT ListeCourses()
{
 short i,nbre_produit;
 boolean OK=False;
 tperiode_achat periode[max_periode_a_traiter];
 tproduitcompos liste_produit[maxproduitsplat*maxthememenu*max_periode_a_traiter];
 struct RastPort *rp;
 struct BitMap *bm;
 rp=Window->RPort;
 bm=rp->BitMap;
 rport = *rp;
 Bitmap = *bm;
 Bitmap.BytesPerRow=40;
 Bitmap.Rows=60;
 Bitmap.Depth=5;
 rport.BitMap=&Bitmap;
 for(i=0;i<5;i++)
 {
  plans[i]=(UBYTE *) AllocMem(2400,MEMF_CHIP|MEMF_CLEAR);
  Bitmap.Planes[i]=plans[i];
 }
 for(i=0;i<calendrier_long;i++)
    if(calendrier[i].traite) OK=True;
 if(OK)
 {
  nbre_prod=0;
  ecran3d(periode);
  if(periode[0].debut.annee!=-1)
  {
   deter_prod_qte_nec_menu_glo(periode,1,liste_produit,&nbre_produit);
   ListeProduitsAAcheter(liste_produit,nbre_produit,periode);
  }
 }
 else
 {
  ConfirmationOK(tab_msg[25]);
  return 0;
 }
 for(i=0;i<5;i++)
    FreeMem(plans[i],2400);

}
```

```
/*
 * ---> Module  : Ecran0
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
PRIVATE struct Gadget G_E_OK, G_E_Stop, G_E_Nom, G_E_But;

PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[9];
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(!strcmp(id,"G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[0].son);
 }
 else if(!strcmp(id,"G_E_OK"))
 {
  choix=ConfirmationOuiNon(tab_msg[3]);
  if(choix==True)
  {
   retval=True;
   selection=1;
  }
  else
  {
   DeselectionneGadget(&G_E_OK);
   RaffraichitEGadget();
  }
 }
 else if(!strcmp(id,"G_E_Stop"))
 {
  choix=ConfirmationOuiNon(tab_msg[3]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
```

```
 else
 {
  DeselectionneGadget(&G_E_Stop);
  RaffraichitEGadget();
 }
}
if(retval!=True)
  SelectionneEStrGadget(&G_E_Nom);
return(retval);
}

/*-----------------------------------------*/
/* Gestion de l'écran 0 : attente d'un évènement */
/*-----------------------------------------*/
USHORT ecran0(prenom)
char *prenom;
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;

 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_Stop, gadget[2], "G_E_Stop", 249, 212);
 AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 212);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[0].nom,&G_E_But);
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("PRENOM");
 AjouteEStrGadget(&G_E_Nom, "G_E_Nom", 96, 80, 15, prenom);
 RaffraichitEGadget();
 Prononce(tab_msg[0].son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
 SelectionneEStrGadget(&G_E_Nom);
 do
 {
  Wait(1L << Window->UserPort->mp_SigBit);
  while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
   {
    class=msg->Class;
    if(class==GADGETUP)
     Sortie=GestionEvent(msg);
    else
     SelectionneEStrGadget(&G_E_Nom);
    ReplyMsg((struct Message *) msg);
   }
 } while(Sortie==False);
 retval = selection;
```

```
 /*---------------------------------------*/
 /* Désinstanciation des boutons de l'écran */
 /*---------------------------------------*/
 SupprimeEGadget(&G_E_Stop);
 SupprimeEGadget(&G_E_But);
 SupprimeEGadget(&G_E_Nom);
 SupprimeEGadget(&G_E_OK);
 return(retval);
}
```

```
/*
 * ---> Module  : Ecran1
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   lng_msg -> longueur du message
   CH_MENU -> choix menu
   CH_LISTE -> choix liste de courses
   gadget -> tableau avec les différents gadgets
   phrase -> phrase affichée dans le philactère
   G_E_... -> Gadget apparaissant sur l'écran
*/
#define  CH_MENU    1
#define  CH_LISTE   2

PRIVATE struct Gadget G_E_OK, G_E_Stop, G_E_Ch_M, G_E_Ch_L, G_E_But;

PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[9];
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;

/*-------------------------------------------------*/
/* Gestion de l'exclusion des deux boutons de choix */
/*-------------------------------------------------*/
PRIVATE void BoutonsExclusifs(g)
struct Gadget *g;
{
 SelectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_Ch_M"))
 {
  DeselectionneGadget(&G_E_Ch_L);
  selection=CH_MENU;
  RaffraichitEGadget();
  Prononce(tab_msg[1].son);
 }
 else
 {
  DeselectionneGadget(&G_E_Ch_M);
  selection=CH_LISTE;
  RaffraichitEGadget();
  Prononce(tab_msg[2].son);
 }
 if(is_OK_visible==False)
 {
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  is_OK_visible=True;
  RaffraichitEGadget();
 }
}
```

```
/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(!strcmp(id,"G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[0].son);
 }
 else if((!strcmp(id,"G_E_Ch_M"))||(!strcmp(id,"G_E_Ch_L")))
  BoutonsExclusifs(g);
 else if(!strcmp(id,"G_E_OK"))
  retval=True;
 else if(!strcmp(id,"G_E_Stop"))
 {
  choix=ConfirmationOuiNon(tab_msg[3]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
  {
   DeselectionneGadget(&G_E_Stop);
   RaffraichitEGadget();
  }
 }
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du premier écran : attente d'un évènement */
/*-------------------------------------------------*/
USHORT ecran1()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 /*-----------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-----------------------------------*/
 AjouteEGadget(&G_E_Ch_M, gadget[0], "G_E_Ch_M", 40, 51);
 AjouteEGadget(&G_E_Ch_L, gadget[1], "G_E_Ch_L", 40, 104);
 AjouteEGadget(&G_E_Stop, gadget[2], "G_E_Stop", 249, 210);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[0].nom,&G_E_But);
 /*--------------------*/
 /* Affichage de l'écran */
 /*--------------------*/
 AfficheFond();
```

```
AfficheTitre("Je Fais Ma Liste de Courses");
RaffraichitEGadget();
Prononce(tab_msg[0].son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
do
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
} while(Sortie==False);
retval = selection;
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SupprimeEGadget(&G_E_Ch_M);
SupprimeEGadget(&G_E_Ch_L);
SupprimeEGadget(&G_E_Stop);
SupprimeEGadget(&G_E_But);
if(is_OK_visible == True)
{
  SupprimeEGadget(&G_E_OK);
  is_OK_visible=False;
}
return(retval);
}
```

```
/*
 * ---> Module  : Ecran2
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   lng_msg -> longueur du message
   CH_CRE_MENU -> choix création menu
   CH_MOD_MENU -> choix modification menu
   CH_CON_MENU -> choix consultation menu
   CH_IMP_MENU -> choix impression menu
   gadget -> tableau avec les différents gadgets
   phrase -> phrase affichée dans le philactère
   G_E_... -> Gadget apparaissant sur l'écran
*/

#define  CH_CRE_MENU    1
#define  CH_MOD_MENU    2
#define  CH_CON_MENU    3
#define  CH_IMP_MENU    4

PRIVATE struct Gadget G_E_OK, G_E_Ret, G_E_CRE_M, G_E_MOD_M, G_E_CON_M,
                      G_E_IMP_M, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;

/*------------------------------------------------*/
/* Gestion de l'exclusion des quatre boutons de choix */
/*------------------------------------------------*/
PRIVATE void BoutonsExclusifs(g)
struct Gadget *g;
{
 SelectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_CRE_M"))
 {
  DeselectionneGadget(&G_E_MOD_M);
  DeselectionneGadget(&G_E_CON_M);
  DeselectionneGadget(&G_E_IMP_M);
  selection=CH_CRE_MENU;
  RaffraichitEGadget();
  Prononce(tab_msg[5].son);
 }
 if(!strcmp(IdGadget(g),"G_E_MOD_M"))
 {
  DeselectionneGadget(&G_E_CRE_M);
  DeselectionneGadget(&G_E_CON_M);
  DeselectionneGadget(&G_E_IMP_M);
  selection=CH_MOD_MENU;
  RaffraichitEGadget();
```

```
 Prononce(tab_msg[6].son);
}
if(!strcmp(IdGadget(g),"G_E_CON_M"))
{
 DeselectionneGadget(&G_E_MOD_M);
 DeselectionneGadget(&G_E_CRE_M);
 DeselectionneGadget(&G_E_IMP_M);
 selection=CH_CON_MENU;
 RaffraichitEGadget();
 Prononce(tab_msg[7].son);
}
if(!strcmp(IdGadget(g),"G_E_IMP_M"))
{
 DeselectionneGadget(&G_E_MOD_M);
 DeselectionneGadget(&G_E_CON_M);
 DeselectionneGadget(&G_E_CRE_M);
 selection=CH_IMP_MENU;
 RaffraichitEGadget();
 Prononce(tab_msg[8].son);
}
if(is_OK_visible==False)
{
 AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
 is_OK_visible=True;
 RaffraichitEGadget();
}
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(strcmp(id,"G_E_But")==0)
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[4].son);
 }
 if((strcmp(id,"G_E_CRE_M")==0)||(strcmp(id,"G_E_MOD_M")==0)
    ||(strcmp(id,"G_E_CON_M")==0)||(strcmp(id,"G_E_IMP_M")==0))
  BoutonsExclusifs(g);
 if(strcmp(id,"G_E_OK")==0)
  retval=True;
 if(strcmp(id,"G_E_Ret")==0)
 {
  choix=ConfirmationOuiNon(tab_msg[9]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
```

```
  {
   DeselectionneGadget(&G_E_Ret);
   RaffraichitEGadget();
  }
 }
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du premier écran : attente d'un évènement */
/*-------------------------------------------------*/
USHORT ecran2()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 /*-----------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-----------------------------------*/
 AjouteEGadget(&G_E_CRE_M, gadget[8], "G_E_CRE_M", 0, 35);
 AjouteEGadget(&G_E_MOD_M, gadget[9], "G_E_MOD_M", 0, 78);
 AjouteEGadget(&G_E_CON_M, gadget[10], "G_E_CON_M", 0, 121);
 AjouteEGadget(&G_E_IMP_M, gadget[11], "G_E_IMP_M", 0, 164);
 AjouteEGadget(&G_E_Ret, gadget[7], "G_E_Ret", 249, 210);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[4].nom,&G_E_But);
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("M E N U");
 RaffraichitEGadget();
 Prononce(tab_msg[4].son);
 /*-------------------------------------------------*/
 /* Répeter                                         */
 /*    Attente d'un évènement                       */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                           */
 /*-------------------------------------------------*/
 do
 {
  Wait(1L << Window->UserPort->mp_SigBit);
  while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
   {
    class=msg->Class;
    if(class==GADGETUP)
     Sortie=GestionEvent(msg);
    ReplyMsg((struct Message *) msg);
   }
 } while(Sortie==False);
 retval = selection;
 /*---------------------------------------*/
 /* Désinstanciation des boutons de l'écran */
 /*---------------------------------------*/
 SupprimeEGadget(&G_E_CRE_M);
 SupprimeEGadget(&G_E_MOD_M);
 SupprimeEGadget(&G_E_CON_M);
 SupprimeEGadget(&G_E_IMP_M);
```

```
  SupprimeEGadget(&G_E_Ret);
  SupprimeEGadget(&G_E_But);
  if(is_OK_visible == True)
  {
    SupprimeEGadget(&G_E_OK);
    is_OK_visible=False;
  }
  return(retval);
}
```

```
/*
 * ---> Module  : Ecran3
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ctypes.h"
#define COUL_A_TRAITER  6
#define COUL_TRAITE    11
#define CHOIX_OUI       1
#define CHOIX_NON       2

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   gadget -> tableau avec les différents gadgets
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_Ret, G_E_Sel, G_E_Oui, G_E_Non, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC boolean CONFIRM_OK;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC struct TextAttr TextAttr;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC short calendrier_long;

PRIVATE struct Image image;
PRIVATE USHORT selection, jour_traite=0;
PRIVATE boolean is_OK_visible=False;
PRIVATE st_msg Message_But;
PRIVATE struct madate
{
   char nom[22];
   char son[50];
   USHORT couleur;
} tab_date[max_eloignement];
PRIVATE char son_jour[50]="";

/*
   son du jour
*/

PRIVATE void calcul_son(jour)
short jour;
{
      if((jour>1)&&(jour<17))
        strcpy(son_jour,tab_unite[jour]);
      else if(jour==1)
        strcpy(son_jour,tab_dizaine[0]);
```

```
      else if((jour==20)||(jour==30))
        strcpy(son_jour,tab_dizaine[jour/10]);
      else if((jour==21)||(jour==31))
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,"/HEY/H");
            strcat(son_jour,tab_unite[1]);
        }
      else
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,tab_unite[jour%10]);
        }
}
/*
 * remplissage du tableau des dates
 */
PRIVATE void RemplirCalendrier(void)
{
   int i;
   boolean premier_non_traite = False;
   for(i=0;i<calendrier_long;i++)
   {
      strcpy(tab_date[i].nom,tab_nomjours[(calendrier[i].nom)].nom);
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,itoa(calendrier[i].jour));
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,tab_nommois[(calendrier[i].mois)].nom);
      strcpy(tab_date[i].son,tab_nomjours[(calendrier[i].nom)].son);
      strcat(tab_date[i].son,"/H");
      calcul_son(calendrier[i].jour);
      strcat(tab_date[i].son,son_jour);
      strcat(tab_date[i].son,"/H");
      strcat(tab_date[i].son,tab_nommois[(calendrier[i].mois)].son);
      if(calendrier[i].traite)
        tab_date[i].couleur = COUL_TRAITE;
      else if(!premier_non_traite)
      {
         premier_non_traite = True;
         tab_date[i].couleur = COUL_A_TRAITER;
         jour_traite = i;
      }
      else
        tab_date[i].couleur = NO_COUL_FOND;
   }
}
/*
  Affichage du Calendrier
*/
PRIVATE void AfficheCalendrier(void)
{
   int ligne=56, i;
   AfficheImage(&image,0,50);
   for(i=0;i<calendrier_long;i++)
   {
      AfficheTexte(tab_date[i].nom,7,ligne,1,tab_date[i].couleur,&TextAttr,15);
      ligne+=15;
   }
}
```

```
/*
  Gestion de l'exclusion des boutons Oui - Non
*/

PRIVATE void BoutonsExclusifs(g)
struct Gadget *g;
{
 SelectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_Oui"))
 {
  DeselectionneGadget(&G_E_Non);
  selection=CHOIX_OUI;
  RaffraichitEGadget();
  Prononce("UWIX");
 }
 if(eval(IdGadget(g),"=","G_E_Non"))
 {
  DeselectionneGadget(&G_E_Oui);
  selection=CHOIX_NON;
  RaffraichitEGadget();
  Prononce("NOHN");
 }
 if((is_OK_visible==False)&&(CONFIRM_OK))
 {
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  is_OK_visible=True;
  RaffraichitEGadget();
 }
}
/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(eval(IdGadget(g),"=","G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(Message_But.son);
 }
 if(eval(IdGadget(g),"=","G_E_Sel"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_date[jour_traite].son);
 }
 if((strcmp(id,"G_E_Oui")==0)||(strcmp(id,"G_E_Non")==0))
 {
  BoutonsExclusifs(g);
  if(!CONFIRM_OK)
    retval=True;
 }
```

```
 if(strcmp(id,"G_E_OK")==0)
  retval=True;
 if(strcmp(id,"G_E_Ret")==0)
 {
  choix=ConfirmationOuiNon(tab_msg[9]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
  {
   DeselectionneGadget(&G_E_Ret);
   RaffraichitEGadget();
  }
 }
 return(retval);
}
/*-------------------------------------------------------------------*/
/* Gestion du troisième écran (1er Scénario) : attente d'un évènement */
/*-------------------------------------------------------------------*/
USHORT ecran3()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 ChargeImage("Deroul_fond.pic",&image);
 RemplirCalendrier();
 /*-----------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-----------------------------------*/
 AjouteEGadget(&G_E_Ret, gadget[7], "G_E_Ret", 249, 210);
 AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, 50+(15*jour_traite));
 AfficheInGadget(tab_date[jour_traite].nom, &G_E_Sel,5, 3, 1,
                 COUL_A_TRAITER ,&TextAttr,10);
 AjouteEGadget(&G_E_Oui, gadget[5], "G_E_Oui", 176, 63);
 AjouteEGadget(&G_E_Non, gadget[6], "G_E_Non", 244, 63);
 strcpy(Message_But.nom,tab_msg[17].nom);
 strcat(Message_But.nom,tab_date[jour_traite].nom);
 strcpy(Message_But.son,tab_msg[17].son);
 strcat(Message_But.son,tab_date[jour_traite].son);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(Message_But.nom,&G_E_But);
 selection=0;
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("CALENDRIER");
 AfficheCalendrier();
 RaffraichitEGadget();
 Prononce(Message_But.son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
```

```
do
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
} while(Sortie==False);
retval = selection;
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SupprimeEGadget(&G_E_Ret);
SupprimeEGadget(&G_E_Sel);
SupprimeEGadget(&G_E_Oui);
SupprimeEGadget(&G_E_Non);
SupprimeEGadget(&G_E_But);
if(is_OK_visible == True)
{
  SupprimeEGadget(&G_E_OK);
  is_OK_visible=False;
}
LibereImage(&image);
return(retval);
}
```

```
/*
 * ---> Module  : Ecran3b
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ctypes.h"
#define COUL_A_TRAITER  6
#define COUL_TRAITE    11
#define CHOIX_OUI       1
#define CHOIX_NON       2

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   gadget -> tableau avec les différents gadgets
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_Ret, G_E_Sel, G_E_FL_H, G_E_FL_B, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC boolean CONFIRM_OK;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC struct TextAttr TextAttr;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC short calendrier_long;

PRIVATE struct Image image;
PRIVATE USHORT selection, jour_traite=0;
PRIVATE boolean is_OK_visible=False;
PRIVATE st_msg Message_But;
PRIVATE struct madate
{
   char nom[22];
   char son[50];
   USHORT couleur;
} tab_date[max_eloignement];
PRIVATE USHORT POSY_Sel;
PRIVATE char son_jour[50]="";

/*
   son du jour
*/

PRIVATE void calcul_son(jour)
short jour;
{
     if((jour>1)&&(jour<17))
       strcpy(son_jour,tab_unite[jour]);
     else if(jour==1)
```

```
        strcpy(son_jour,tab_dizaine[0]);
      else if((jour==20)||(jour==30))
        strcpy(son_jour,tab_dizaine[jour/10]);
      else if((jour==21)||(jour==31))
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,"/HEY/H");
            strcat(son_jour,tab_unite[1]);
        }
      else
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,tab_unite[jour%10]);
        }
}

/*
 * remplissage du tableau des dates
 */
PRIVATE void RemplirCalendrier(void)
{
   int i;
   boolean premier_non_traite = False;
   for(i=0;i<calendrier_long;i++)
   {
      strcpy(tab_date[i].nom,tab_nomjours[(calendrier[i].nom)].nom);
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,itoa(calendrier[i].jour));
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,tab_nommois[(calendrier[i].mois)].nom);
      strcpy(tab_date[i].son,tab_nomjours[(calendrier[i].nom)].son);
      strcat(tab_date[i].son,"/H");
      calcul_son(calendrier[i].jour);
      strcat(tab_date[i].son,son_jour);
      strcat(tab_date[i].son,"/H");
      strcat(tab_date[i].son,tab_nommois[(calendrier[i].mois)].son);
      if(calendrier[i].traite)
        tab_date[i].couleur = COUL_TRAITE;
      else if(!premier_non_traite)
      {
         premier_non_traite = True;
         tab_date[i].couleur = NO_COUL_FOND;
         jour_traite = i;
      }
      else
        tab_date[i].couleur = NO_COUL_FOND;
   }
}

/*
  Affichage du Calendrier
*/
PRIVATE void AfficheCalendrier(void)
{
   int ligne=56, i;
   AfficheImage(&image,0,50);
   for(i=0;i<calendrier_long;i++)
   {
      AfficheTexte(tab_date[i].nom,7,ligne,1,tab_date[i].couleur,&TextAttr,15);
```

```
      ligne+=15;
    }
}

PRIVATE boolean existe_pre_non_traite()
{
 boolean retval=False;
 int i;
 for(i=1;i<selection;i++)
    if(!calendrier[i-1].traite)
      retval=True;
 return retval;
}

PRIVATE boolean existe_post_non_traite()
{
 boolean retval=False;
 int i;
 for(i=selection+1;i<=calendrier_long;i++)
    if(!calendrier[i-1].traite)
      retval=True;
 return retval;
}

/*---------------------
   Gestion du Dérouleur
  ---------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
  if((selection>1)&&(existe_pre_non_traite()))
  {
   selection--;
   POSY_Sel-=15;
   while(calendrier[selection-1].traite)
   {
    POSY_Sel-=15;
    selection--;
   }
  }
  else;
 else
  if((selection<calendrier_long)&&(selection>0)
      &&(existe_post_non_traite()))
  {
   selection++;
   POSY_Sel+=15;
   while(calendrier[selection-1].traite)
   {
    POSY_Sel+=15;
    selection++;
   }
  }
  else;
 if(selection==0)
 {
  selection=jour_traite+1;
```

```
  POSY_Sel=51+15*jour_traite;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
  is_OK_visible=True;
 }
 ChangePosGadget(&G_E_Sel, 2, POSY_Sel);
 AfficheCalendrier();
 AfficheInGadget(tab_date[selection-1].nom,&G_E_Sel,5,3,1,NO_COUL_FOND,&TextAttr,10);
 RaffraichitEGadget();
 Prononce(tab_date[selection-1].son);
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(eval(id,"=","G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[20].son);
 }
 if(eval(id,"=","G_E_FL_H")||eval(id,"=","G_E_FL_B"))
  GestionDeroul(g);
 if(eval(id,"=","G_E_OK"))
  retval=True;
 if(eval(id,"=","G_E_Sel"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_date[selection-1].son);
 }
 if(eval(id,"=","G_E_Ret"))
 {
  choix=ConfirmationOuiNon(tab_msg[9]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
  {
   DeselectionneGadget(&G_E_Ret);
   RaffraichitEGadget();
  }
 }
 return(retval);
}

/*-------------------------------------------------------------------*/
/* Gestion du troisième écran (2eme Scénario) : attente d'un évènement */
/*-------------------------------------------------------------------*/
```

```
USHORT ecran3b()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 ChargeImage("Deroul_fond.pic",&image);
 RemplirCalendrier();
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_Ret, gadget[7], "G_E_Ret", 249, 210);
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[20].nom,&G_E_But);
 selection=0;
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("CALENDRIER");
 AfficheCalendrier();
 RaffraichitEGadget();
 Prononce(tab_msg[20].son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
 do
 {
  Wait(1L << Window->UserPort->mp_SigBit);
  while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
   {
    class=msg->Class;
    if(class==GADGETUP)
     Sortie=GestionEvent(msg);
    ReplyMsg((struct Message *) msg);
   }
 } while(Sortie==False);
 retval = selection;
 /*---------------------------------------*/
 /* Désinstanciation des boutons de l'écran */
 /*---------------------------------------*/
 SupprimeEGadget(&G_E_Ret);
 SupprimeEGadget(&G_E_FL_H);
 SupprimeEGadget(&G_E_FL_B);
 SupprimeEGadget(&G_E_But);
 if(is_OK_visible == True)
 {
   SupprimeEGadget(&G_E_OK);
   SupprimeEGadget(&G_E_Sel);
   is_OK_visible=False;
 }
 LibereImage(&image);
 return(retval);
}
```

```
/*
 * ---> Module  : Ecran3c
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_Ctypes.h"
#define COUL_A_TRAITER  6
#define COUL_TRAITE    11
#define CHOIX_OUI       1
#define CHOIX_NON       2

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   gadget -> tableau avec les différents gadgets
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_Ret, G_E_Sel, G_E_FL_H, G_E_FL_B, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC boolean CONFIRM_OK;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC struct TextAttr TextAttr;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC short calendrier_long;

PRIVATE struct Image image;
PRIVATE USHORT selection, jour_traite=0;
PRIVATE boolean is_OK_visible=False;
PRIVATE st_msg Message_But;
PRIVATE struct madate
{
   char nom[22];
   char son[50];
   USHORT couleur;
} tab_date[max_eloignement];
PRIVATE USHORT POSY_Sel;
PRIVATE char son_jour[50]="";

/*
   son du jour
*/

PRIVATE void calcul_son(jour)
short jour;
{
     if((jour>1)&&(jour<17))
       strcpy(son_jour,tab_unite[jour]);
     else if(jour==1)
```

```
        strcpy(son_jour,tab_dizaine[0]);
      else if((jour==20)||(jour==30))
        strcpy(son_jour,tab_dizaine[jour/10]);
      else if((jour==21)||(jour==31))
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,"/HEY/H");
            strcat(son_jour,tab_unite[1]);
        }
      else
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,tab_unite[jour%10]);
        }
}

/*
 * remplissage du tableau des dates
 */
PRIVATE void RemplirCalendrier(void)
{
   int i;
   boolean premier_traite = False;
   for(i=0;i<calendrier_long;i++)
   {
      strcpy(tab_date[i].nom,tab_nomjours[(calendrier[i].nom)].nom);
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,itoa(calendrier[i].jour));
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,tab_nommois[(calendrier[i].mois)].nom);
      strcpy(tab_date[i].son,tab_nomjours[(calendrier[i].nom)].son);
      strcat(tab_date[i].son,"/H");
      calcul_son(calendrier[i].jour);
      strcat(tab_date[i].son,son_jour);
      strcat(tab_date[i].son,"/H");
      strcat(tab_date[i].son,tab_nommois[(calendrier[i].mois)].son);
      if(calendrier[i].traite)
      {
       tab_date[i].couleur = COUL_TRAITE;
       if(!premier_traite)
       {
        jour_traite = i;
        premier_traite = True;
       }
      }
      else
       tab_date[i].couleur = NO_COUL_FOND;
   }
}

/*
  Affichage du Calendrier
*/
PRIVATE void AfficheCalendrier(void)
{
   int ligne=56, i;
   AfficheImage(&image,0,50);
   for(i=0;i<calendrier_long;i++)
   {
```

```
      AfficheTexte(tab_date[i].nom,7,ligne,1,tab_date[i].couleur,&TextAttr,15);
      ligne+=15;
   }
}

PRIVATE boolean existe_pre_traite()
{
 boolean retval=False;
 int i;
 for(i=1;i<selection;i++)
    if(calendrier[i-1].traite)
      retval=True;
 return retval;
}

PRIVATE boolean existe_post_traite()
{
 boolean retval=False;
 int i;
 for(i=selection+1;i<=calendrier_long;i++)
    if(calendrier[i-1].traite)
      retval=True;
 return retval;
}

/*---------------------
   Gestion du Dérouleur
  ---------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
  if((selection>1)&&(existe_pre_traite()))
  {
   selection--;
   POSY_Sel-=15;
   while(!calendrier[selection-1].traite)
   {
    POSY_Sel-=15;
    selection--;
   }
  }
  else;
 else
  if((selection<calendrier_long)&&(selection>0)
      &&(existe_post_traite()))
  {
   selection++;
   POSY_Sel+=15;
   while(!calendrier[selection-1].traite)
   {
    POSY_Sel+=15;
    selection++;
   }
  }
  else;
 if(selection==0)
 {
```

```
  selection=jour_traite+1;
  POSY_Sel=51+15*jour_traite;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
  is_OK_visible=True;
 }
 ChangePosGadget(&G_E_Sel, 2, POSY_Sel);
 AfficheCalendrier();
 AfficheInGadget(tab_date[selection-1].nom,&G_E_Sel,5,3,1,COUL_TRAITE,&TextAttr,10);
 RaffraichitEGadget();
 Prononce(tab_date[selection-1].son);
}
/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(eval(id,"=","G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[20].son);
 }
 if(eval(id,"=","G_E_FL_H")||eval(id,"=","G_E_FL_B"))
  GestionDeroul(g);
 if(eval(id,"=","G_E_OK"))
  retval=True;
 if(eval(id,"=","G_E_Sel"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_date[selection-1].son);
 }
 if(eval(id,"=","G_E_Ret"))
 {
  choix=ConfirmationOuiNon(tab_msg[9]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
  {
   DeselectionneGadget(&G_E_Ret);
   RaffraichitEGadget();
  }
 }
 return(retval);
}

/*-------------------------------------------------------------------*/
/* Gestion du troisième écran (jours traités) : attente d'un évènement */
/*-------------------------------------------------------------------*/
```

```
USHORT ecran3c()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 ChargeImage("Deroul_fond.pic",&image);
 RemplirCalendrier();
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_Ret, gadget[7], "G_E_Ret", 249, 210);
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[20].nom,&G_E_But);
 selection=0;
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("CALENDRIER");
 AfficheCalendrier();
 RaffraichitEGadget();
 Prononce(tab_msg[20].son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
 do
 {
  Wait(1L << Window->UserPort->mp_SigBit);
  while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
   {
    class=msg->Class;
    if(class==GADGETUP)
     Sortie=GestionEvent(msg);
    ReplyMsg((struct Message *) msg);
   }
 } while(Sortie==False);
 retval = selection;
 /*---------------------------------------*/
 /* Désinstanciation des boutons de l'écran */
 /*---------------------------------------*/
 SupprimeEGadget(&G_E_Ret);
 SupprimeEGadget(&G_E_FL_H);
 SupprimeEGadget(&G_E_FL_B);
 SupprimeEGadget(&G_E_But);
 if(is_OK_visible == True)
 {
   SupprimeEGadget(&G_E_OK);
   SupprimeEGadget(&G_E_Sel);
   is_OK_visible=False;
 }
 LibereImage(&image);
 return(retval);
}
```

```
/*
 * ---> Module  : Ecran3d
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types2.h"
#include "H_Types.h"
#include "H_Ctypes.h"
#define COUL_A_TRAITER  6
#define COUL_TRAITE    11
#define COUL_PERIODE   30
#define CHOIX_OUI       1
#define CHOIX_NON       2
#define MAXVALUE    65535

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   gadget -> tableau avec les différents gadgets
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_Ret, G_E_Sel, G_E_FL_H, G_E_FL_B,
                      G_E_Date, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC boolean CONFIRM_OK;
PUBLIC tcalendrier calendrier[max_eloignement];
PUBLIC struct TextAttr TextAttr;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[12];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC short calendrier_long;

PRIVATE struct Image image;
PRIVATE USHORT selection, jour_traite=0, select;
PRIVATE boolean is_OK_visible=False;
PRIVATE st_msg Message_But;
PRIVATE struct madate
{
   char nom[22];
   char son[50];
   USHORT couleur;
} tab_date[max_eloignement];
PRIVATE USHORT POSY_Sel;
PRIVATE char son_jour[50]="";

/*
   son du jour
*/

PRIVATE void calcul_son(jour)
short jour;
{
```

```
      if((jour>1)&&(jour<17))
        strcpy(son_jour,tab_unite[jour]);
      else if(jour==1)
        strcpy(son_jour,tab_dizaine[0]);
      else if((jour==20)||(jour==30))
        strcpy(son_jour,tab_dizaine[jour/10]);
      else if((jour==21)||(jour==31))
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,"/HEY/H");
            strcat(son_jour,tab_unite[1]);
        }
      else
        {
            strcpy(son_jour,tab_dizaine[jour/10]);
            strcat(son_jour,tab_unite[jour%10]);
        }
}

/*
 * remplissage du tableau des dates
 */
PRIVATE void RemplirCalendrier(void)
{
   int i;
   boolean premier_traite = False;
   for(i=0;i<calendrier_long;i++)
   {
      strcpy(tab_date[i].nom,tab_nomjours[(calendrier[i].nom)].nom);
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,itoa(calendrier[i].jour));
      strcat(tab_date[i].nom," ");
      strcat(tab_date[i].nom,tab_nommois[(calendrier[i].mois)].nom);
      strcpy(tab_date[i].son,tab_nomjours[(calendrier[i].nom)].son);
      strcat(tab_date[i].son,"/H");
      calcul_son(calendrier[i].jour);
      strcat(tab_date[i].son,son_jour);
      strcat(tab_date[i].son,"/H");
      strcat(tab_date[i].son,tab_nommois[(calendrier[i].mois)].son);
      if(calendrier[i].traite)
      {
       tab_date[i].couleur = COUL_TRAITE;
       if(!premier_traite)
       {
        premier_traite = True;
        jour_traite = i;
       }
      }
      else
        tab_date[i].couleur = NO_COUL_FOND;
   }
}

/*
  Affichage du Calendrier
*/
PRIVATE void AfficheCalendrier(void)
{
   int ligne=56, i;
```

```
   AfficheImage(&image,0,50);
   for(i=0;i<calendrier_long;i++)
   {
      if(((select<selection)&&(i>=(select-1))&&(i<selection)&&(selection!=0))
        ||((selection>select)&&(i>=(selection-1))&&(i<select)))
        AfficheTexte(tab_date[i].nom,7,ligne,1,COUL_PERIODE,&TextAttr,15);
      else
        AfficheTexte(tab_date[i].nom,7,ligne,1,tab_date[i].couleur,
                     &TextAttr,15);
      ligne+=15;
   }
}

PRIVATE boolean existe_pre_traite()
{
 boolean retval=False;
 int i;
 for(i=1;i<selection;i++)
    if(calendrier[i-1].traite)
      retval=True;
 return retval;
}

PRIVATE boolean existe_post_traite()
{
 boolean retval=False;
 int i;
 for(i=selection+1;i<=calendrier_long;i++)
    if(calendrier[i-1].traite)
      retval=True;
 return retval;
}

/*----------------------
   Gestion du Dérouleur
  ----------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
  if((selection>1)&&(existe_pre_traite()))
  {
   selection--;
   POSY_Sel-=15;
   while(!calendrier[selection-1].traite)
   {
    POSY_Sel-=15;
    selection--;
   }
  }
  else;
 else
  if((selection<calendrier_long)&&(selection>0)
      &&(existe_post_traite()))
  {
   selection++;
   POSY_Sel+=15;
   while(!calendrier[selection-1].traite)
```

```
   {
    POSY_Sel+=15;
    selection++;
   }
  }
  else;
 if(selection==0)
 {
  selection=jour_traite+1;
  POSY_Sel=51+15*jour_traite;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
  AjouteEGadget(&G_E_Date, gadget[17], "G_E_Date", 30, 210);
  is_OK_visible=True;
 }
 ChangePosGadget(&G_E_Sel, 2, POSY_Sel);
 AfficheCalendrier();
 if(select!=0)
    AfficheInGadget(tab_date[selection-1].nom,&G_E_Sel,5,3,1,
                    COUL_PERIODE,&TextAttr,10);
 else
    AfficheInGadget(tab_date[selection-1].nom,&G_E_Sel,5,3,1,
                    NO_COUL_FOND,&TextAttr,10);
 RaffraichitEGadget();
 Prononce(tab_date[selection-1].son);
}


/*----------------------------------*/
/* Fonction de gestion des évènements */
/*----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(eval(id,"=","G_E_But"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[20].son);
 }
 if(eval(id,"=","G_E_FL_H")||eval(id,"=","G_E_FL_B"))
  GestionDeroul(g);
 if(eval(id,"=","G_E_OK"))
  retval=True;
 if(eval(id,"=","G_E_Sel"))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_date[selection-1].son);
 }
 if(eval(id,"=","G_E_Date"))
 {
  DeselectionneGadget(g);
  AfficheInGadget(tab_date[selection-1].nom,&G_E_Sel,5,3,1,
```

```
                COUL_PERIODE,&TextAttr,10);
 RaffraichitEGadget();
 select=selection;
 Prononce(tab_date[selection-1].son);
 }
 if(eval(id,"=","G_E_Ret"))
 {
  choix=ConfirmationOuiNon(tab_msg[9]);
  if(choix==True)
  {
   retval=True;
   selection=0;
  }
  else
  {
   DeselectionneGadget(&G_E_Ret);
   RaffraichitEGadget();
  }
 }
 return(retval);
}

/*-------------------------------------------------------------------------*/
/* Gestion du troisième écran (Liste de courses) : attente d'un évènement */
/*-------------------------------------------------------------------------*/
void ecran3d(periode)
tperiode_achat *periode;
{
 boolean Sortie=False;
 ULONG class;
 USHORT tamp;
 struct IntuiMessage *msg = NULL;
 ChargeImage("Deroul_fond.pic",&image);
 RemplirCalendrier();
 /*-----------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-----------------------------------*/
 AjouteEGadget(&G_E_Ret, gadget[7], "G_E_Ret", 249, 210);
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[20].nom,&G_E_But);
 selection=0;
 select=MAXVALUE;
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("CALENDRIER");
 AfficheCalendrier();
 RaffraichitEGadget();
 Prononce(tab_msg[20].son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
 do
```

```
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
} while(Sortie==False);
if(selection!=0)
{
 if(select>selection)
 {
  tamp=select;
  select=selection;
  selection=tamp;
 }
 periode[0].debut.jour = calendrier[select-1].jour;
 periode[0].debut.mois = calendrier[select-1].mois;
 periode[0].debut.annee = calendrier[select-1].annee;
 periode[0].fin.jour = calendrier[selection-1].jour;
 periode[0].fin.mois = calendrier[selection-1].mois;
 periode[0].fin.annee = calendrier[selection-1].annee;
 periode[0].achat_fait = False;
}
else
    periode[0].debut.annee = -1;
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SupprimeEGadget(&G_E_Ret);
SupprimeEGadget(&G_E_FL_H);
SupprimeEGadget(&G_E_FL_B);
SupprimeEGadget(&G_E_But);
if(is_OK_visible == True)
{
  SupprimeEGadget(&G_E_OK);
  SupprimeEGadget(&G_E_Date);
  SupprimeEGadget(&G_E_Sel);
  is_OK_visible=False;
}
LibereImage(&image);
}
```

```
/*
 * ---> Module  : Ecran4
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   lng_msg -> longueur du message
   gadget -> tableau avec les différents gadgets
   phrase -> phrase affichée dans le philactère
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_FL_H, G_E_FL_B,
                      G_E_Sel, G_E_Img, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;
PRIVATE struct Image image[7], *OLD_IMAGE;
PRIVATE USHORT POSY_Sel;
PRIVATE char pmsg[6][12] = {"1 Personne","2 Personnes",
                            "3 Personnes","4 Personnes",
                            "5 Personnes","6 Personnes"};

/*---------------------
   Gestion du Dérouleur
  ---------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
  if(selection>1)
  {
   selection--;
   POSY_Sel-=20;
  }
  else;
 else
  if((selection<6)&&(selection>0))
  {
   selection++;
   POSY_Sel+=20;
  }
  else;
 if(selection==0)
 {
  selection++;
  POSY_Sel=63;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
```

```
  AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
 }
 ChangePosGadget(&G_E_Sel, 2, POSY_Sel);
 AfficheImage(&image[0],0,51);
 AfficheInGadget(pmsg[selection-1],&G_E_Sel,40,3,1,NO_COUL_FOND,&TextAttr,10);
 SetImgInGadget(&G_E_Img,&image[selection]);
 RaffraichitEGadget();
 Prononce(tab_msg[10+selection].son);
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(strcmp(id,"G_E_But")==0)
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  Prononce(tab_msg[10].son);
 }
 if((strcmp(id,"G_E_FL_H")==0)||(strcmp(id,"G_E_FL_B")==0))
  GestionDeroul(g);
 if((strcmp(id,"G_E_Sel")==0)||(strcmp(id,"G_E_Img")==0))
 {
  DeselectionneGadget(g);
  RaffraichitEGadget();
  if(selection!=0)
  {
   Prononce(tab_msg[10+selection].son);
  }
  else;
 }
 if(strcmp(id,"G_E_OK")==0)
  retval=True;
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du quatrième écran : attente d'un évènement */
/*-------------------------------------------------*/
USHORT ecran4(pers, affiche)
USHORT pers;
boolean affiche;
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval,i;
 struct IntuiMessage *msg = NULL;
 /*-----------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-----------------------------------*/
 if(affiche)
```

```
{
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
}
AjouteEGadget(&G_E_Img, gadget[15], "G_E_Img", 195, 57);
AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
AfficheBulle(tab_msg[10].nom,&G_E_But);
OLD_IMAGE=GetImgInGadget(&G_E_Img);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeImage("pers.pic",&image[0]);
ChargeImage("Un.pic",&image[1]);
ChargeImage("Deux.pic",&image[2]);
ChargeImage("Trois.pic",&image[3]);
ChargeImage("Quatre.pic",&image[4]);
ChargeImage("Cinq.pic",&image[5]);
ChargeImage("Six.pic",&image[6]);
/*---------------------*/
/* Affichage de l'écran */
/*---------------------*/
AfficheFond();
AfficheTitre("Nombre de personnes");
AfficheImage(&image[0],0,51);
RaffraichitEGadget();
Prononce(tab_msg[10].son);
selection=pers;
if(selection!=0)
{
 POSY_Sel=63+20*(selection-1);
 AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
 AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
 AfficheInGadget(pmsg[selection-1],&G_E_Sel,40,4,1,NO_COUL_FOND,&TextAttr,10);
 SetImgInGadget(&G_E_Img,&image[selection]);
 RaffraichitEGadget();
 Prononce(tab_msg[10+selection].son);
}
/*-------------------------------------------------*/
/* Répeter                                         */
/*    Attente d'un évènement                       */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                           */
/*-------------------------------------------------*/
do
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
} while(Sortie==False);
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SetImgInGadget(&G_E_Img,OLD_IMAGE);
```

```
if(affiche)
 {
  SupprimeEGadget(&G_E_FL_H);
  SupprimeEGadget(&G_E_FL_B);
 }
 SupprimeEGadget(&G_E_Sel);
 SupprimeEGadget(&G_E_Img);
 SupprimeEGadget(&G_E_OK);
 SupprimeEGadget(&G_E_But);
 /*----------------------
    Libération des images
   ----------------------*/
 for(i=0;i<7;i++)
   LibereImage(&image[i]);
 retval = selection;
 return(retval);
}
```

```
/*
 * ---> Module  : Ecran5
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   lng_msg -> longueur du message
   gadget -> tableau avec les différents gadgets
   phrase -> phrase affichée dans le philactère
   G_E_... -> Gadget apparaissant sur l'écran
*/

PRIVATE struct Gadget G_E_OK, G_E_FL_H, G_E_FL_B, G_E_Sel, G_E_But,
                      G_E_Fin, G_E_Menu;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC ttheme_pere_fils_bool Tab_themes_b[maxthemes];
PUBLIC short themes_long;
PUBLIC tparamutil param;

PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;
PRIVATE struct Image image;
PRIVATE USHORT POSY_Sel;
PRIVATE ttheme_pere_fils_bool Tab_mes_themes[maxthemes];
PRIVATE short themes_long_bis;
PRIVATE st_msg conf;

/*--------------------
   Gestion des thèmes
  --------------------*/

PRIVATE boolean CalculThemes(void)
{
 int i;
 boolean retval;
 themes_long_bis=0;
 for(i=0;i<themes_long;i++)
  if((Tab_themes_b[i].nbr_fils==0)&&(!Tab_themes_b[i].select))
    Tab_mes_themes[themes_long_bis++]=Tab_themes_b[i];
  else if (!param.themeelem)
    {
     if(!Tab_themes_b[i].select)
       Tab_mes_themes[themes_long_bis++]=Tab_themes_b[i];
     i+=Tab_themes_b[i].nbr_fils;
    }
 return (retval=(themes_long_bis==0) ? False : True);
}
```

```
PRIVATE void AfficheThemes(void)
{
   int ligne=68, i;
   AfficheImage(&image,0,50);
   for(i=0;i<themes_long_bis;i++)
   {
      AfficheTexte(Tab_mes_themes[i].nom,7,ligne,1,
                   Tab_mes_themes[i].couleur,&TextAttr,15);
      ligne+=20;
   }
}

/*---------------------
   Gestion du Dérouleur
  ---------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
  if(selection>1)
  {
   selection--;
   POSY_Sel-=20;
  }
  else;
 else
  if((selection<themes_long_bis)&&(selection>0))
  {
   selection++;
   POSY_Sel+=20;
  }
  else;
 if(selection==0)
 {
  selection++;
  POSY_Sel=63;
  is_OK_visible=True;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, POSY_Sel);
 }
 ChangePosGadget(&G_E_Sel, 2, POSY_Sel);
 AfficheThemes();
 AfficheInGadget(Tab_mes_themes[selection-1].nom,&G_E_Sel,5,3,1,
                 Tab_mes_themes[selection-1].couleur,&TextAttr,10);
 RaffraichitEGadget();
 Prononce(Tab_mes_themes[selection-1].son);
}

/*----------------------------------*/
/* Fonction de gestion des évènements */
/*----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
```

```
id=IdGadget(g);
if(strcmp(id,"G_E_But")==0)
{
 DeselectionneGadget(g);
 RaffraichitEGadget();
 Prononce(tab_msg[18].son);
}
if((strcmp(id,"G_E_FL_H")==0)||(strcmp(id,"G_E_FL_B")==0))
 GestionDeroul(g);
if(strcmp(id,"G_E_Sel")==0)
{
 DeselectionneGadget(g);
 RaffraichitEGadget();
 if(selection!=0)
 {
  Prononce(Tab_mes_themes[selection-1].son);
 }
 else;
}
if(strcmp(id,"G_E_OK")==0)
{
 retval=True;
}
if(strcmp(id,"G_E_Fin")==0)
{
 retval=True;
 selection=0;
}
if(strcmp(id,"G_E_Menu")==0)
{
 retval=True;
 selection+=10;
}
return(retval);
}

PRIVATE USHORT tab_ptr(num)
short num;
{
 USHORT i=0;
 while((Tab_themes_b[i].no_theme!=Tab_mes_themes[num].no_theme)
       &&(i<themes_long)) i++;
 return i;
}

/*-------------------------------------------------*/
/* Gestion du cinquième écran : attente d'un évènement */
/*-------------------------------------------------*/
USHORT ecran5(couleur)
UBYTE *couleur;
{
 ULONG class;
 struct IntuiMessage *msg = NULL;
 boolean is_possible = CalculThemes();
 boolean Sortie = is_possible ? False : True;
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
```

```
AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
AjouteEGadget(&G_E_Fin, gadget[16], "G_E_Fin", 249, 210);
AjouteEGadget(&G_E_Menu, gadget[17], "G_E_Menu", 30, 210);
selection=0;
AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
AfficheBulle(tab_msg[18].nom,&G_E_But);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeImage("deroul_fond.pic",&image);
/*----------------------*/
/* Affichage de l'écran */
/*----------------------*/
AfficheFond();
AfficheTitre("Themes");
AfficheThemes();
RaffraichitEGadget();
Prononce(tab_msg[18].son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
while(Sortie==False)
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
}
if(!is_possible)
{
 AfficheFond();
 AfficheTitre("Themes");
 ConfirmationOK(tab_msg[22]);
}
/*----------------------
   Libération des images
  ----------------------*/
  LibereImage(&image);
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SupprimeEGadget(&G_E_Fin);
SupprimeEGadget(&G_E_FL_H);
SupprimeEGadget(&G_E_FL_B);
SupprimeEGadget(&G_E_But);
SupprimeEGadget(&G_E_Menu);
if(is_OK_visible)
{
 SupprimeEGadget(&G_E_Sel);
 SupprimeEGadget(&G_E_OK);
 is_OK_visible=False;
}
```

```
if(selection==0)
   return 9999;
else if(selection>=10)
   return selection;
else
  {
   *couleur=Tab_mes_themes[selection-1].couleur;
   Tab_themes_b[tab_ptr(selection-1)].select=True;
   return (USHORT) Tab_mes_themes[selection-1].no_theme;
  }
}
```

```
/*
 * ---> Module  : Ecran6
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_CTypes.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/

PRIVATE struct Gadget G_E_OK, G_E_FL_H, G_E_FL_B,
                      G_E_Sel, G_E_Img, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC ttheme_pere_fils Tab_themes[maxplatstheme];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme];
PUBLIC short plat_long;
PUBLIC short themes_long;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];

PRIVATE UBYTE couleur_theme;
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;
PRIVATE struct Image image[maxplatstheme+1], *OLD_IMAGE;
PRIVATE short POS_DEBUT;
PRIVATE USHORT POSY_Sel;
PRIVATE boolean is_Decomp;
PRIVATE short NBRE_PLATS;
PRIVATE tplatstheme Tab_mes_plats[maxplatstheme];
PRIVATE short numero_theme;
PRIVATE st_msg messagebut;

/*
   Gestion des plats
 */

PRIVATE boolean TestDecomp(void)
{
 if(Tab_plats[0].nbr_fils!=0)
   return True;
 else
   return False;
}

PRIVATE void ChargePlats(void)
{
 int j=0, k;
 for(NBRE_PLATS=0;j<plat_long;NBRE_PLATS++)
 {
  strcpy(Tab_mes_plats[NBRE_PLATS].fichier,"plat");
  strcat(Tab_mes_plats[NBRE_PLATS].fichier,itoa(Tab_plats[j].no_plat));
  strcat(Tab_mes_plats[NBRE_PLATS].fichier,".pic");
```

```
 strcpy(Tab_mes_plats[NBRE_PLATS].nom,"");
 strcpy(Tab_mes_plats[NBRE_PLATS].son,"");
 Tab_mes_plats[NBRE_PLATS].num_plat=Tab_plats[j].no_plat;
 Tab_mes_plats[NBRE_PLATS].couleur_theme=couleur_theme;
 Tab_mes_plats[NBRE_PLATS].num_theme=numero_theme;
 Tab_mes_plats[NBRE_PLATS].decomp= (Tab_plats[j].nbr_fils!=0) ? True : False;
 ChargeImage(Tab_mes_plats[NBRE_PLATS].fichier,&image[NBRE_PLATS+1]);
 if(!Tab_mes_plats[NBRE_PLATS].decomp)
 {
  strcat(Tab_mes_plats[NBRE_PLATS].nom,Tab_plats[j].nom);
  strcat(Tab_mes_plats[NBRE_PLATS].son,Tab_plats[j].son);
 }
 for(k=0;k<Tab_plats[j].nbr_fils;k++)
 {
  strcat(Tab_mes_plats[NBRE_PLATS].nom,Tab_plats[j+k+1].nom);
  strcat(Tab_mes_plats[NBRE_PLATS].nom,"#");
  strcat(Tab_mes_plats[NBRE_PLATS].son,Tab_plats[j+k+1].son);
  strcat(Tab_mes_plats[NBRE_PLATS].son,",");
 }
 j+=Tab_plats[j].nbr_fils;
 ++j;
 }
}

PRIVATE void AfficheNonDecPlats(void)
{
 int i, j;
 USHORT pos_ligne=56,max;
 max=(selection==0) ? 4 : 5;
 for(i=0;i<max-selection;i++)pos_ligne+=15;
 for(i=4;i>0;i--)
  if((j=selection-i)>0)
  {
   AfficheTexte(Tab_mes_plats[j-1].nom,7,pos_ligne,1,couleur_theme,
                &TextAttr,15);
   pos_ligne+=15;
  }
 if(selection!=0)
   AfficheTexte(Tab_mes_plats[selection-1].nom,7,pos_ligne,1,couleur_theme,
                &TextAttr,15);
  pos_ligne+=15;
 for(i=0;i<4;i++)
  if((j=selection+i)<NBRE_PLATS)
  {
   AfficheTexte(Tab_mes_plats[j].nom,7,pos_ligne,1,couleur_theme,
                &TextAttr,15);
   pos_ligne+=15;
  }
}

PRIVATE void AfficheDecPlats(void)
{
 USHORT pos_ligne=56;
 if(selection>1)
  AfficheTexte(Tab_mes_plats[selection-2].nom,7,pos_ligne,1,couleur_theme,
               &TextAttr,15);
 pos_ligne+=45;
 if(selection!=0)
  AfficheTexte(Tab_mes_plats[selection-1].nom,7,pos_ligne,1,couleur_theme,
```

```
                &TextAttr,15);
 pos_ligne+=45;
 if(selection<NBRE_PLATS)
   AfficheTexte(Tab_mes_plats[selection].nom,7,pos_ligne,1,couleur_theme,
                &TextAttr,15);
}

PRIVATE void AffichePlats(void)
{
 AfficheImage(&image[0],0,50);
 if(is_Decomp)
   AfficheDecPlats();
 else
   AfficheNonDecPlats();
}

/*---------------------
   Gestion du Dérouleur
  ---------------------*/
PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_FL_H"))
   if(selection>1)
     selection--;
   else;
 else
   if((selection<NBRE_PLATS)&&(selection>0))
     selection++;
  else;
 if(selection==0)
 {
  selection++;
  AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
  if(is_Decomp)
    AjouteEGadget(&G_E_Sel, gadget[19], "G_E_Sel", 2, 95);
  else
    AjouteEGadget(&G_E_Sel, gadget[14], "G_E_Sel", 2, 110);
  POS_DEBUT=0;
 }
 AffichePlats();
 AfficheInGadget(Tab_mes_plats[selection-1].nom,&G_E_Sel,5,3,1,
                 couleur_theme,&TextAttr,15);
 SetImgInGadget(&G_E_Img,&image[selection]);
 RaffraichitEGadget();
 Prononce(Tab_mes_plats[selection-1].son);
}

/*----------------------------------*/
/* Fonction de gestion des évènements */
/*----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
```

```
 id=IdGadget(g);
 if(!strcmp(id,"G_E_But"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    Prononce(messagebut.son);
   }
 else if((!strcmp(id,"G_E_FL_H"))||(!strcmp(id,"G_E_FL_B")))
   GestionDeroul(g);
 else if((!strcmp(id,"G_E_Sel"))||(!strcmp(id,"G_E_Img")))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Tab_mes_plats[selection-1].son);
    else;
    }
 else if(strcmp(id,"G_E_OK")==0)
    retval=True;
 return(retval);
}

PRIVATE short priorite_theme(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return Tab_themes[i].priorite;
}

PRIVATE short pos_theme(no_theme)
short no_theme;
{
 short i=0;
 while((Tab_themes[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return i;
}

/*-----------------------------------------------------*/
/* Gestion du sixième écran : attente d'un évènement */
/*-----------------------------------------------------*/
USHORT ecran6(num_theme, coul_theme)
UBYTE coul_theme;
short num_theme;
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval,i,ptheme=pos_theme(num_theme);
 struct IntuiMessage *msg = NULL;
 couleur_theme=coul_theme;
 numero_theme=num_theme;
 is_Decomp=TestDecomp();
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
```

```
AjouteEGadget(&G_E_Img, gadget[18], "G_E_Img", 158, 34);
AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
messagebut=tab_msg[19];
strcat(messagebut.nom,Tab_themes[ptheme].nom);
strcat(messagebut.son,Tab_themes[ptheme].son);
AfficheBulle(messagebut.nom,&G_E_But);
OLD_IMAGE=GetImgInGadget(&G_E_Img);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeImage("deroul_fond.pic",&image[0]);
ChargePlats();
/*----------------------*/
/* Affichage de l'écran */
/*----------------------*/
AfficheFond();
AfficheTitre("PLATS");
selection=0;
AffichePlats();
RaffraichitEGadget();
Prononce(messagebut.son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
do
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
  {
   class=msg->Class;
   if(class==GADGETUP)
    Sortie=GestionEvent(msg);
   ReplyMsg((struct Message *) msg);
  }
} while(Sortie==False);
/*----------------------
   Libération des images
  ----------------------*/
for(i=0;i<NBRE_PLATS+1;i++)
  LibereImage(&image[i]);
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SetImgInGadget(&G_E_Img,OLD_IMAGE);
SupprimeEGadget(&G_E_FL_H);
SupprimeEGadget(&G_E_FL_B);
SupprimeEGadget(&G_E_Sel);
SupprimeEGadget(&G_E_Img);
SupprimeEGadget(&G_E_OK);
SupprimeEGadget(&G_E_But);
retval = Tab_mes_plats[selection-1].num_plat;
Tab_Plat_Theme[priorite_theme(Tab_mes_plats[selection-1].num_theme)]
       = Tab_mes_plats[selection-1];
return(retval);
}
```

```
/*
 * ---> Module  : Ecran7
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */
#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_CTypes.h"

PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC ttheme_pere_fils Tab_themes[maxplatstheme];
PUBLIC tplat_pere_fils Tab_plats[maxplatstheme];
PUBLIC short plat_long;
PUBLIC short themes_long;
PUBLIC tplatstheme Tab_Plat_Theme[maxthemes];
PUBLIC st_msg tab_msg[nbre_msg];

PRIVATE struct Gadget G_E_FL_H, G_E_FL_B, G_E_Sel, G_E_Poubelle,
                      G_E_But, G_E_Img, G_E_Fin;
PRIVATE tplatstheme Tab_mes_plats[maxthemes];
PRIVATE short idx_tab_plat_theme[maxthemes+1], idx_img[maxthemes+1];
PRIVATE short NBRE_PLATS, NBRE_PLATS_S;
PRIVATE tplatsaffiche Tab_plats_affiche[9];
PRIVATE short selection;
PRIVATE struct Image image[maxthemes+1], *OLD_IMAGE;
PRIVATE USHORT POSY_Sel;
PRIVATE boolean is_Poubelle=False;
PRIVATE boolean is_Sel=False;
PRIVATE First;
PRIVATE st_msg conf;
PRIVATE boolean Affiche;

PRIVATE void SupprimePlat(void)
{
 int i;
 for(i=selection;i<NBRE_PLATS+1;i++)
 {
  idx_tab_plat_theme[i-1]=idx_tab_plat_theme[i];
  idx_img[i-1]=idx_img[i];
 }
 --NBRE_PLATS;
}

PRIVATE void Mise_a_jour_Tab_plats(void)
{
 int i,ptr=0;
 for(i=0;i<maxthemes;i++)
    if(i!=idx_tab_plat_theme[ptr])
    {
     strcpy(Tab_Plat_Theme[i].nom,"");
     Tab_Plat_Theme[i].num_plat=-1;
    }
    else ++ptr;
}

PRIVATE void Init_idx_tab_plat_theme(void)
```

```
{
 int i;
 for(i=0;i<maxthemes+1;idx_tab_plat_theme[i++]=-1);
}

PRIVATE void Garnir_idx_tab_plat_theme(void)
{
 int i;
 NBRE_PLATS=0;
 Init_idx_tab_plat_theme();
 for(i=0;i<maxthemes;i++)
    if(!eval(Tab_Plat_Theme[i].nom,"=",""))
      idx_tab_plat_theme[NBRE_PLATS++]=i;
 NBRE_PLATS_S=NBRE_PLATS;
}

PRIVATE void ChargePlats(void)
{
 int i;
 for(i=0;i<NBRE_PLATS;i++)
 {
  ChargeImage(Tab_Plat_Theme[idx_tab_plat_theme[i]].fichier,&image[i+1]);
  idx_img[i]=i+1;
 }
}

PRIVATE boolean est_Affiche_plat(num)
short num;
{
 int i;
 for(i=0;(i<9)&&(Tab_plats_affiche[i].num_plat!=num);i++)
 return (i==9) ? False : Tab_plats_affiche[i].entier;
}

PRIVATE boolean Premier_affiche(num)
short num;
{
 return (Tab_plats_affiche[0].num_plat==num) ? True : False;
}

PRIVATE boolean Dernier_affiche(num)
short num;
{
 return (Tab_plats_affiche[8].num_plat==num) ? True : False;
}

PRIVATE void AffichePlatsSimple_1(void)
{
 int i, ligne=56;
 AfficheImage(&image[0],0,50);
 for(i=0;i<NBRE_PLATS;i++)
 {
  AfficheTexte(Tab_Plat_Theme[idx_tab_plat_theme[i]].nom,7,ligne,1,
               Tab_Plat_Theme[idx_tab_plat_theme[i]].couleur_theme,
               &TextAttr,15);
  if(Tab_Plat_Theme[idx_tab_plat_theme[i]].decomp)
    ligne+=45;
  else
    ligne+=15;
```

```
 }
}

PRIVATE void AffichePlatsSimple_2(void)
{
 int i, ligne=56,num_index=(POSY_Sel-51)/15;
 AfficheImage(&image[0],0,50);
 for(i=1;i<=num_index;i++)
    AfficheTexte(Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].nom,7,
                 ligne+(15*i),1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].couleur_theme,
                 &TextAttr,15);
 AfficheTexte(Tab_Plat_Theme[idx_tab_plat_theme[i-1]].nom,7,ligne+(15*i),
              1,Tab_Plat_Theme[idx_tab_plat_theme[i-1]].couleur_theme,
              &TextAttr,15);
 for(i=0;(i<9-num_index)&&(i+selection<NBRE_PLATS);i++);
    AfficheTexte(Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].nom,7,
                 ligne+(15*i),1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].couleur_theme,
                 &TextAttr,15);
}

PRIVATE void decompose_plat(ch1, tch2)
char ch1;
tnom_plat tch2[];
{
 int i=0;
 *tch2[i]=strtok(ch1,"#");
 while(tch2[i++]!=NULL)
      *tch2[i]=strtok(NULL,"#");
}

PRIVATE void init_tab(void)
{
 int i;
 for(i=0;i<9;strcpy(Tab_plats_affiche[i++].nom,""));
}

PRIVATE void AffichePlatsComplexe(void)
{
 int i,ligne=56,num_index=(POSY_Sel-51)/15,k,j=num_index-1;
 tnom_plat nomplat[3];
 for(i=1;j>=0;i++)
 {
  init_tab();
  if(Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].decomp)
  {
   decompose_plat(Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].nom,
                  nomplat);
   for(k=2;(k<=0)&&(j>=0);k--)
   {
    strcpy(Tab_plats_affiche[j].nom,nomplat[k]);
    Tab_plats_affiche[j].entier=((j-2<0)||(j-1<0)) ? False : True;
    Tab_plats_affiche[j].num_plat=
         Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].num_plat;
    Tab_plats_affiche[j].couleur=
         Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].couleur_theme;
    --j;
   }
```

```
  }
  else
  {
   strcpy(Tab_plats_affiche[j].nom,
          Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].nom);
   Tab_plats_affiche[j].num_plat=
          Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].num_plat;
   Tab_plats_affiche[j].entier=True;
   Tab_plats_affiche[j].couleur=
          Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].couleur_theme;
    --j;
  }
 }
 strcpy(Tab_plats_affiche[num_index].nom,
          Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].nom);
 Tab_plats_affiche[num_index].num_plat=
          Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].num_plat;
 Tab_plats_affiche[num_index].entier=True;
 Tab_plats_affiche[num_index].couleur=
           Tab_Plat_Theme[idx_tab_plat_theme[selection-(i+1)]].couleur_theme;
 j=num_index+1;
 for(i=0;j>9-num_index;i++)
 {
  init_tab();
  if(Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].decomp)
  {
   decompose_plat(Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].nom,
                  nomplat);
   for(k=0;(k<=2)&&(j>=0);k++)
   {
    strcpy(Tab_plats_affiche[j].nom,nomplat[k]);
    Tab_plats_affiche[j].entier=((j+2>9)||(j+1>9)) ? False : True;
    Tab_plats_affiche[j].num_plat=
          Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].num_plat;
    Tab_plats_affiche[j].couleur=
          Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].couleur_theme;
    ++j;
   }
  }
  else
  {
   strcpy(Tab_plats_affiche[j].nom,
          Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].nom);
   Tab_plats_affiche[j].num_plat=
          Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].num_plat;
   Tab_plats_affiche[j].entier=True;
   Tab_plats_affiche[j].couleur=
          Tab_Plat_Theme[idx_tab_plat_theme[selection+i]].couleur_theme;
    ++j;
  }
 }
 for(i=0;i<9;i++)
    AfficheTexte(Tab_plats_affiche[i].nom,7,ligne,1,
                 Tab_plats_affiche[i].couleur,&TextAttr,15);
}

PRIVATE void Gestion_simple_1(g)
struct Gadget *g;
{
```

```
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PLATS)&&(selection>0))
   {
    ++selection;
    POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
  if(Affiche)
  {
  AjouteEGadget(&G_E_Poubelle,gadget[20],"G_E_Poubelle",140,210);
  is_Poubelle=True;
  }
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 ChangePosGadget(&G_E_Sel,0,POSY_Sel);
 AffichePlatsSimple_1();
 AfficheInGadget(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].nom,
                 &G_E_Sel,5,3,1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].couleur_theme,
                 &TextAttr,1);
 SetImgInGadget(&G_E_Img,&image[idx_img[selection-1]]);
 RaffraichitEGadget();
 Prononce(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].son);
}

PRIVATE void Gestion_simple_2(g)
struct Gaddget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    selection--;
    if(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].decomp)
      POSY_Sel-=45;
    else
      POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PLATS)&&(selection>0))
   {
    selection++;
    if(Tab_Plat_Theme[idx_tab_plat_theme[selection-2]].decomp)
      POSY_Sel+=45;
```

```
    else
      POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
  if(Affiche)
  {
  AjouteEGadget(&G_E_Poubelle,gadget[20],"G_E_Poubelle",140,210);
  is_Poubelle=True;
  }
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 if(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].decomp)
   AjouteEGadget(&G_E_Sel,gadget[19],"G_E_Sel",2,POSY_Sel);
 else
   AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 AffichePlatsSimple_1();
 AfficheInGadget(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].nom,
                 &G_E_Sel,5,3,1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].couleur_theme,
                 &TextAttr,15);
 SetImgInGadget(&G_E_Img,&image[idx_img[selection-1]]);
 RaffraichitEGadget();
 Prononce(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].son);
}

PRIVATE void Gestion_simple_3(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    if(POSY_Sel>71)
      POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PLATS)&&(selection>0))
   {
    selection++;
    if(POSY_Sel<146)
      POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
  if(Affiche)
  {
  AjouteEGadget(&G_E_Poubelle,gadget[20],"G_E_Poubelle",140,210);
  is_Poubelle=True;
  }
 }
```

```
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 AffichePlatsSimple_2();
 AfficheInGadget(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].nom,
                 &G_E_Sel,5,3,1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].couleur_theme,
                 &TextAttr,15);
 SetImgInGadget(&G_E_Img,&image[idx_img[selection-1]]);
 RaffraichitEGadget();
 Prononce(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].son);
}

PRIVATE void Gestion_complexe(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    if((est_Affiche_plat(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].num_plat))
       &&(!Premier_affiche(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].num_plat)))
      if(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].decomp)
        POSY_Sel-=45;
      else
        POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PLATS)&&(selection>0))
   {
    selection++;
    if((est_Affiche_plat(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].num_plat))
       &&(!Dernier_affiche(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].num_plat)))
      if(Tab_Plat_Theme[idx_tab_plat_theme[selection-2]].decomp)
        POSY_Sel+=45;
      else
        POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
  if(Affiche)
  {
  AjouteEGadget(&G_E_Poubelle,gadget[20],"G_E_Poubelle",140,210);
  is_Poubelle=True;
  }
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 if(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].decomp)
   AjouteEGadget(&G_E_Sel,gadget[19],"G_E_Sel",2,POSY_Sel);
 else
   AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 AffichePlatsComplexe();
```

```
 AfficheInGadget(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].nom,
                 &G_E_Sel,5,3,1,
                 Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].couleur_theme,
                 &TextAttr,15);
 SetImgInGadget(&G_E_Img,&image[idx_img[selection-1]]);
 RaffraichitEGadget();
 Prononce(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].son);
}

PRIVATE USHORT MaxPlatsAffiche(void)
{
 USHORT retval=0, i;
 for(i=0;i<NBRE_PLATS;i++)
    if(Tab_Plat_Theme[idx_tab_plat_theme[i]].decomp)
      retval+=3;
    else
      retval++;
 return retval;
}

PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 if((NBRE_PLATS==MaxPlatsAffiche())&&(MaxPlatsAffiche()<=9))
   Gestion_simple_1(g);
 else if(MaxPlatsAffiche()<=9)
   Gestion_simple_2(g);
 else if(NBRE_PLATS==MaxPlatsAffiche())
   Gestion_simple_3(g);
 else
   Gestion_complexe(g);
 RaffraichitEGadget();
}

PRIVATE void AffichePlats(void)
{
 if(First)
   selection++;
 if(MaxPlatsAffiche()<9)
   AffichePlatsSimple_1();
 else if(NBRE_PLATS==MaxPlatsAffiche())
   AffichePlatsSimple_2();
 else
   AffichePlatsComplexe();
 if(First)
 {
  selection--;
  First=False;
 }
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
```

```
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(!strcmp(id,"G_E_But"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    Prononce(tab_msg[21].son);
   }
 else if((!strcmp(id,"G_E_FL_H"))||(!strcmp(id,"G_E_FL_B")))
   GestionDeroul(g);
 else if((!strcmp(id,"G_E_Sel"))||(!strcmp(id,"G_E_Img")))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Tab_Plat_Theme[idx_tab_plat_theme[selection-1]].son);
    else;
    }
 else if(strcmp(id,"G_E_Poubelle")==0)
   {
    DeselectionneGadget(g);
    SupprimePlat();
    if(selection>1)
      GestionDeroul(&G_E_FL_H);
    else if(idx_tab_plat_theme[selection-1]!=-1)
      GestionDeroul(&G_E_FL_B);
    RaffraichitEGadget();
   }
 else if(strcmp(id,"G_E_Fin")==0)
    retval=True;
 return(retval);
}

PRIVATE short priorite_theme(no_theme)
short no_theme;
{
 USHORT i=0;
 while((Tab_themes[i].no_theme!=no_theme)
       &&(i<themes_long)) i++;
 return Tab_themes[i].priorite;
}

/*-----------------------------------------------------*/
/* Gestion du septième écran : attente d'un évènement */
/*-----------------------------------------------------*/
void ecran7(affiche)
boolean affiche;
{
 boolean Sortie=False;
 ULONG class;
 USHORT i;
 struct IntuiMessage *msg = NULL;
 First=True;
 is_Poubelle=is_Sel=False;
 Affiche=affiche;
 Garnir_idx_tab_plat_theme();
 if(NBRE_PLATS!=0)
 {
```

```
/*-------------------------------------*/
/* Instanciation des boutons de l'écran */
/*-------------------------------------*/
AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
AjouteEGadget(&G_E_Img, gadget[18], "G_E_Img", 158, 34);
AjouteEGadget(&G_E_Fin, gadget[16], "G_E_Fin", 249, 210);
AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
AfficheBulle(tab_msg[21].nom,&G_E_But);
OLD_IMAGE=GetImgInGadget(&G_E_Img);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeImage("deroul_fond.pic",&image[0]);
ChargePlats();
/*---------------------*/
/* Affichage de l'écran */
/*---------------------*/
AfficheFond();
AfficheTitre("VISUALISATION DU MENU");
selection=0;
AffichePlats();
RaffraichitEGadget();
Prononce(tab_msg[21].son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
while((Sortie==False)&&(NBRE_PLATS!=0))
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
 {
  class=msg->Class;
  if(class==GADGETUP)
   Sortie=GestionEvent(msg);
  ReplyMsg((struct Message *) msg);
 }
}
if(NBRE_PLATS==0)
{
 AfficheFond();
 AfficheTitre("VISUALISATION DU MENU");
 if(NBRE_PLATS_S==0)
   ConfirmationOK(tab_msg[23]);
 else
   ConfirmationOK(tab_msg[24]);
}
if(NBRE_PLATS_S!=0)
{
 /*----------------------
    Libération des images
   ----------------------*/
 for(i=0;i<NBRE_PLATS_S+1;i++)
    LibereImage(&image[i]);
 /*-------------------------------------*/
```

```
 /* Désinstanciation des boutons de l'écran */
 /*--------------------------------------*/
 SetImgInGadget(&G_E_Img,OLD_IMAGE);
 SupprimeEGadget(&G_E_FL_H);
 SupprimeEGadget(&G_E_FL_B);
 SupprimeEGadget(&G_E_Img);
 SupprimeEGadget(&G_E_But);
 SupprimeEGadget(&G_E_Fin);
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 if(is_Poubelle&&Affiche)
   SupprimeEGadget(&G_E_Poubelle);
 Mise_a_jour_Tab_plats();
 }
}
```

```
/*
 * ---> Module  : Ecran8
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"

/*------------------------------*/
/* Variables globales de l'écran */
/*------------------------------*/
/*
   gadget -> tableau avec les différents gadgets
   phrase -> phrase affichée dans le philactère
   G_E_... -> Gadget apparaissant sur l'écran
*/

#define  CH_CON_MENU    1
#define  CH_IMP_MENU    2
#define  CH_CON_PRODUIT 3
#define  CH_IMP_PRODUIT 4

PRIVATE struct Gadget G_E_OK, G_E_Fin, G_E_CON_M, G_E_IMP_M, G_E_CON_P,
                      G_E_IMP_P, G_E_But;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC struct Gadget gadget[max_gadgets];
PRIVATE USHORT selection;
PRIVATE boolean is_OK_visible=False;

/*-------------------------------------------------*/
/* Gestion de l'exclusion des quatre boutons de choix */
/*-------------------------------------------------*/
PRIVATE void BoutonsExclusifs(g)
struct Gadget *g;
{
 SelectionneGadget(g);
 if(!strcmp(IdGadget(g),"G_E_CON_M"))
 {
  DeselectionneGadget(&G_E_IMP_M);
  DeselectionneGadget(&G_E_CON_P);
  DeselectionneGadget(&G_E_IMP_P);
  selection=CH_CON_MENU;
  RaffraichitEGadget();
  Prononce(tab_msg[27].son);
 }
 if(!strcmp(IdGadget(g),"G_E_IMP_M"))
 {
  DeselectionneGadget(&G_E_CON_M);
  DeselectionneGadget(&G_E_CON_P);
  DeselectionneGadget(&G_E_IMP_P);
  selection=CH_IMP_MENU;
  RaffraichitEGadget();
  Prononce(tab_msg[28].son);
 }
 if(!strcmp(IdGadget(g),"G_E_CON_P"))
 {
  DeselectionneGadget(&G_E_CON_M);
```

```
 DeselectionneGadget(&G_E_IMP_M);
 DeselectionneGadget(&G_E_IMP_P);
 selection=CH_CON_PRODUIT;
 RaffraichitEGadget();
 Prononce(tab_msg[29].son);
 }
 if(!strcmp(IdGadget(g),"G_E_IMP_P"))
 {
 DeselectionneGadget(&G_E_CON_M);
 DeselectionneGadget(&G_E_IMP_M);
 DeselectionneGadget(&G_E_CON_P);
 selection=CH_IMP_PRODUIT;
 RaffraichitEGadget();
 Prononce(tab_msg[30].son);
 }
 if(is_OK_visible==False)
 {
 AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
 is_OK_visible=True;
 RaffraichitEGadget();
 }
}

/*----------------------------------*/
/* Fonction de gestion des évènements */
/*----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False, choix;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(strcmp(id,"G_E_But")==0)
 {
 DeselectionneGadget(g);
 RaffraichitEGadget();
 Prononce(tab_msg[4].son);
 }
 if((strcmp(id,"G_E_IMP_M")==0)||(strcmp(id,"G_E_CON_M")==0)
    ||(strcmp(id,"G_E_IMP_P")==0)||(strcmp(id,"G_E_CON_P")==0))
 BoutonsExclusifs(g);
 if(strcmp(id,"G_E_OK")==0)
 retval=True;
 if(strcmp(id,"G_E_Fin")==0)
 {
 choix=ConfirmationOuiNon(tab_msg[31]);
 if(choix==True)
 {
 retval=True;
 selection=0;
 }
 else
 {
 DeselectionneGadget(&G_E_Fin);
 RaffraichitEGadget();
 }
 }
```

```
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du huitième écran : attente d'un évènement */
/*-------------------------------------------------*/
USHORT ecran8()
{
 boolean Sortie=False;
 ULONG class;
 USHORT retval;
 struct IntuiMessage *msg = NULL;
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_CON_M, gadget[10], "G_E_CON_M", 0, 35);
 AjouteEGadget(&G_E_IMP_M, gadget[11], "G_E_IMP_M", 0, 78);
 AjouteEGadget(&G_E_CON_P, gadget[21], "G_E_CON_P", 0, 121);
 AjouteEGadget(&G_E_IMP_P, gadget[22], "G_E_IMP_P", 0, 164);
 AjouteEGadget(&G_E_Fin, gadget[16], "G_E_Fin", 249, 210);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AfficheBulle(tab_msg[4].nom,&G_E_But);
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("M E N U");
 RaffraichitEGadget();
 Prononce(tab_msg[4].son);
 /*-----------------------------------------------*/
 /* Répeter                                       */
 /*    Attente d'un évènement                     */
 /*    envoi à la routine de gestion des évènements */
 /* Tant que Sortie=False                         */
 /*-----------------------------------------------*/
 do
 {
  Wait(1L << Window->UserPort->mp_SigBit);
  while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
   {
    class=msg->Class;
    if(class==GADGETUP)
     Sortie=GestionEvent(msg);
    ReplyMsg((struct Message *) msg);
   }
 } while(Sortie==False);
 retval = selection;
 /*---------------------------------------*/
 /* Désinstanciation des boutons de l'écran */
 /*---------------------------------------*/
 SupprimeEGadget(&G_E_CON_M);
 SupprimeEGadget(&G_E_IMP_M);
 SupprimeEGadget(&G_E_CON_P);
 SupprimeEGadget(&G_E_IMP_P);
 SupprimeEGadget(&G_E_Fin);
 SupprimeEGadget(&G_E_But);
 if(is_OK_visible == True)
 {
   SupprimeEGadget(&G_E_OK);
```

```
    is_OK_visible=False;
  }
  return(retval);
}
```

```
/*
 * ---> Module  : Ecran9
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */
#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_CTypes.h"

PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC tproduitcompos Tab_Produit[maxproduitsplat*maxthememenu];
PUBLIC st_msg tab_msg[nbre_msg];

PRIVATE struct Gadget G_E_FL_H, G_E_FL_B, G_E_Sel, G_E_Bliss,
                      G_E_But, G_E_Img, G_E_Fin;
PRIVATE short NBRE_PRODUITS;
PRIVATE short selection;
PRIVATE struct Image image[maxproduitsplat*maxthememenu+1],
                     bliss[maxproduitsplat*maxthememenu+1],
                     *OLD_IMAGE, *OLD_BLISS;
PRIVATE USHORT POSY_Sel;
PRIVATE boolean is_Sel=False;
PRIVATE First;

PRIVATE void ChargeProduits(void)
{
 int i;
 char nom_img[15], nom_bliss[20];
 for(i=0;i<NBRE_PRODUITS;i++)
 {
  strcpy(nom_img,"produit");
  strcat(nom_img,itoa(Tab_Produit[i].no_produit));
  strcpy(nom_bliss,nom_img);
  strcat(nom_img,".pic");
  strcat(nom_bliss,".bliss");
  ChargeImage(nom_img,&image[i+1]);
  ChargeImage(nom_bliss,&bliss[i+1]);
 }
}

PRIVATE void AfficheProduitsSimple_1(void)
{
 int i, ligne=56;
 AfficheImage(&image[0],0,50);
 for(i=0;i<NBRE_PRODUITS;i++)
 {
  AfficheTexte(Tab_Produit[i].nom,7,ligne,1,NO_COUL_FOND,&TextAttr,15);
  ligne+=15;
 }
}

PRIVATE void AfficheProduitsSimple_2(void)
{
 int i, ligne=56,num_index=(POSY_Sel-51)/15;
 AfficheImage(&image[0],0,50);
 for(i=1;i<=num_index;i++)
```

```
    AfficheTexte(Tab_Produit[selection-(i+1)].nom,7,ligne+15*(num_index-i),1,
                 NO_COUL_FOND,&TextAttr,15);
 AfficheTexte(Tab_Produit[selection-1].nom,7,ligne+(num_index*15),1,
              NO_COUL_FOND,&TextAttr,15);
 for(i=0;(i<(9-num_index))&&((i+selection)<NBRE_PRODUITS);i++)
    AfficheTexte(Tab_Produit[selection+i].nom,7,ligne+(15*(num_index+i+1)),1,
                 NO_COUL_FOND,&TextAttr,15);
}

PRIVATE void Gestion_simple_1(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PRODUITS)&&(selection>0))
   {
    ++selection;
    POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 ChangePosGadget(&G_E_Sel,0,POSY_Sel);
 AfficheProduitsSimple_1();
 AfficheInGadget(Tab_Produit[selection-1].nom,&G_E_Sel,5,3,1,NO_COUL_FOND,
                 &TextAttr,1);
 SetImgInGadget(&G_E_Img,&image[selection]);
 SetImgInGadget(&G_E_Bliss,&bliss[selection]);
 RaffraichitEGadget();
 Prononce(Tab_Produit[selection-1].son_nom);
}

PRIVATE void Gestion_simple_2(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    if((POSY_Sel>71)||(selection==1))
      POSY_Sel-=15;
   }
   else;
 else
   if((selection<NBRE_PRODUITS)&&(selection>0))
```

```
   {
    selection++;
    if((POSY_Sel<146)||(selection==NBRE_PRODUITS))
      POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection++;
  POSY_Sel=51;
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 AfficheProduitsSimple_2();
 AfficheInGadget(Tab_Produit[selection-1].nom, &G_E_Sel,5,3,1,NO_COUL_FOND,
                 &TextAttr,15);
 SetImgInGadget(&G_E_Img,&image[selection]);
 SetImgInGadget(&G_E_Bliss,&bliss[selection]);
 RaffraichitEGadget();
 Prononce(Tab_Produit[selection-1].son_nom);
}

PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 if(NBRE_PRODUITS<=9)
   Gestion_simple_1(g);
 else
   Gestion_simple_2(g);
 RaffraichitEGadget();
}

PRIVATE void AfficheProduits(void)
{
 if(First)
   selection++;
 if(NBRE_PRODUITS<=9)
   AfficheProduitsSimple_1();
 else
   AfficheProduitsSimple_2();
 if(First)
 {
  selection--;
  First=False;
 }
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
```

```
 if(!strcmp(id,"G_E_But"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    Prononce(tab_msg[33].son);
   }
 else if((!strcmp(id,"G_E_FL_H"))||(!strcmp(id,"G_E_FL_B")))
   GestionDeroul(g);
 else if((!strcmp(id,"G_E_Bliss"))||(!strcmp(id,"G_E_Img"))
        |(!strcmp(id,"G_E_Sel")))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Tab_Produit[selection-1].son_nom);
    else;
    }
 else if(strcmp(id,"G_E_Fin")==0)
    retval=True;
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du neuvième écran : attente d'un évènement */
/*-------------------------------------------------*/
void ecran9(nbre_prod)
short nbre_prod;
{
 boolean Sortie=False;
 ULONG class;
 USHORT i;
 struct IntuiMessage *msg = NULL;
 First=True;
 is_Sel=0;
 NBRE_PRODUITS=nbre_prod;
 POSY_Sel=51;
 selection=0;
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
 AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
 AjouteEGadget(&G_E_Img, gadget[23], "G_E_Img", 150, 34);
 AjouteEGadget(&G_E_Fin, gadget[16], "G_E_Fin", 249, 212);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AjouteEGadget(&G_E_Bliss, gadget[24], "G_E_Bliss", 230, 44);
 AfficheBulle(tab_msg[33].nom,&G_E_But);
 OLD_IMAGE=GetImgInGadget(&G_E_Img);
 OLD_BLISS=GetImgInGadget(&G_E_Bliss);
 /*----------------------
    Chargement des images
   ----------------------*/
 ChargeImage("deroul_fond.pic",&image[0]);
 ChargeProduits();
 /*---------------------*/
 /* Affichage de l'écran */
 /*---------------------*/
 AfficheFond();
 AfficheTitre("INGREDIENTS");
```

```
selection=0;
AfficheProduits();
RaffraichitEGadget();
Prononce(tab_msg[33].son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
while((Sortie==False)&&(NBRE_PRODUITS!=0))
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
 {
  class=msg->Class;
  if(class==GADGETUP)
   Sortie=GestionEvent(msg);
  ReplyMsg((struct Message *) msg);
 }
}
/*----------------------
   Libération des images
  ----------------------*/
for(i=0;i<NBRE_PRODUITS+1;i++)
{
   LibereImage(&image[i]);
   LibereImage(&bliss[i]);
}
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SetImgInGadget(&G_E_Img,OLD_IMAGE);
SetImgInGadget(&G_E_Bliss,OLD_BLISS);
SupprimeEGadget(&G_E_Bliss);
SupprimeEGadget(&G_E_FL_H);
SupprimeEGadget(&G_E_FL_B);
SupprimeEGadget(&G_E_Img);
SupprimeEGadget(&G_E_But);
SupprimeEGadget(&G_E_Fin);
if(is_Sel) SupprimeEGadget(&G_E_Sel);
}
```

```
/*
 * ---> Module  : Ecran10
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_CTypes.h"

typedef struct
{
 char nom[13],son[20];
} ttabquantite;

PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
PUBLIC tarticle Tab_Article[12];
PUBLIC tconditionnement Tab_Cond[10];

PRIVATE struct Gadget G_E_FL_H, G_E_FL_B, G_E_Sel, G_E_Bliss,
                      G_E_But, G_E_Img, G_E_OK, G_E_Qte;
PRIVATE short NBRE_QUANTITES;
PRIVATE short selection;
PRIVATE struct Image image, bliss, imagefond, img_qte[max_quantite_affiche],
                     *OLD_IMAGE, *OLD_BLISS, *OLD_QTE;
PRIVATE USHORT POSY_Sel;
PRIVATE boolean is_Sel=False;
PRIVATE First;
PRIVATE tproduitcompos Produit;
PRIVATE ttabquantite Tab_Quantite[max_quantite];
PRIVATE char son_nombre[31];
PRIVATE st_msg msg_but;

PRIVATE void ChargeProduit(void)
{
 char nom_img[15], nom_bliss[20];
 strcpy(nom_img,"produit");
 strcat(nom_img,itoa(Produit.no_produit));
 strcpy(nom_bliss,nom_img);
 strcat(nom_img,".pic");
 strcat(nom_bliss,".bliss");
 ChargeImage(nom_img,&image);
 ChargeImage(nom_bliss,&bliss);
}

PRIVATE void calcul_son(nombre)
short nombre;
{
      if((nombre>1)&&(nombre<17))
        strcpy(son_nombre,tab_unite[nombre]);
      else if(nombre==1)
        strcpy(son_nombre,tab_dizaine[0]);
      else if((nombre%20)==0)
```

```
        strcpy(son_nombre,tab_dizaine[nombre/10]);
      else if((((nombre-1)%10)==0)&&(nombre<100))
        {
            strcpy(son_nombre,tab_dizaine[nombre/10]);
            strcat(son_nombre,"/HEY/H");
            strcat(son_nombre,tab_unite[1]);
        }
      else
        {
            strcpy(son_nombre,tab_dizaine[nombre/10]);
            strcat(son_nombre,tab_unite[nombre%10]);
        }
}

PRIVATE void ChargeQuantites(void)
{
 int i;
 char nom_img[15], buffer[128];
 for(i=0;i<=NBRE_QUANTITES;i++)
 {
  if(i<max_quantite_affiche)
  {
   strcpy(nom_img,"quantite");
   strcat(nom_img,itoa(i));
   strcat(nom_img,".pic");
   ChargeImage(nom_img,&img_qte[i]);
  }
  strcpy(Tab_Quantite[i].nom,itoa(i));
  strcat(Tab_Quantite[i].nom," ");
  if(i<2)
    if(Produit.cond_util!=0)
      strcat(Tab_Quantite[i].nom,Tab_Cond[Produit.cond_util].nom);
    else
      strcat(Tab_Quantite[i].nom,Produit.nom);
  else
    if(Produit.cond_util!=0)
      strcat(Tab_Quantite[i].nom,pluriel(buffer,
                                         Tab_Cond[Produit.cond_util].nom));
    else
      strcat(Tab_Quantite[i].nom,pluriel(buffer,Produit.nom));
  calcul_son(i);
  strcat(Tab_Quantite[i].son,son_nombre);
  strcat(Tab_Quantite[i].son," ");
  if(Produit.cond_util!=0)
    strcat(Tab_Quantite[i].son,Tab_Cond[Produit.cond_util].son);
  else
    strcat(Tab_Quantite[i].son,Produit.son_nom);
 }
}

PRIVATE void AfficheQuantitesSimple_1(void)
{
 int i, ligne=56;
 AfficheImage(&imagefond,0,50);
 for(i=0;i<=NBRE_QUANTITES;i++)
 {
  AfficheTexte(Tab_Quantite[i].nom,7,ligne,1,NO_COUL_FOND,&TextAttr,15);
  ligne+=15;
 }
```

```
}

PRIVATE void AfficheQuantitesSimple_2(void)
{
 int i, ligne=56,num_index=(POSY_Sel-51)/15;
 AfficheImage(&imagefond,0,50);
 for(i=1;i<=num_index;i++)
    AfficheTexte(Tab_Quantite[selection-(i+1)].nom,7,
                 ligne+15*(num_index-i),1,NO_COUL_FOND,&TextAttr,15);
 AfficheTexte(Tab_Quantite[selection-1].nom,7,ligne+(num_index*15),1,
              NO_COUL_FOND,&TextAttr,15);
 for(i=0;(i<(9-num_index))&&((i+selection)<=NBRE_QUANTITES);i++)
    AfficheTexte(Tab_Quantite[selection+i].nom,7,
                 ligne+(15*(num_index+i+1)),1,NO_COUL_FOND,&TextAttr,15);
}

PRIVATE void Gestion_simple_1(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    POSY_Sel-=15;
   }
   else;
 else
   if((selection<=NBRE_QUANTITES)&&(selection>0))
   {
    ++selection;
    POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection=NBRE_QUANTITES+1;
  POSY_Sel=51+NBRE_QUANTITES*15;
  AjouteEGadget(&G_E_OK,gadget[4],"G_E_OK",140,210);
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 ChangePosGadget(&G_E_Sel,0,POSY_Sel);
 AfficheQuantitesSimple_1();
 AfficheInGadget(Tab_Quantite[selection-1].nom,&G_E_Sel,5,3,1,NO_COUL_FOND,
                 &TextAttr,1);
 if(selection<max_quantite_affiche)
    SetImgInGadget(&G_E_Qte,&img_qte[selection-1]);
 else
    SetImgInGadget(&G_E_Qte,OLD_QTE);
 RaffraichitEGadget();
 Prononce(Tab_Quantite[selection-1].son);
}

PRIVATE void Gestion_simple_2(g)
struct Gadget *g;
{
 DeselectionneGadget(g);
```

```
 if(eval(IdGadget(g),"=","G_E_FL_H"))
   if(selection>1)
   {
    --selection;
    if((POSY_Sel>71)||(selection==1))
      POSY_Sel-=15;
   }
   else;
 else
   if((selection<=NBRE_QUANTITES)&&(selection>0))
   {
    selection++;
    if((POSY_Sel<146)||(selection==NBRE_QUANTITES+1))
      POSY_Sel+=15;
   }
 if(selection==0)
 {
  selection=NBRE_QUANTITES+1;
  POSY_Sel=161;
  AjouteEGadget(&G_E_OK,gadget[4],"G_E_OK",140,210);
 }
 if(is_Sel)
   SupprimeEGadget(&G_E_Sel);
 AjouteEGadget(&G_E_Sel,gadget[14],"G_E_Sel",2,POSY_Sel);
 is_Sel=True;
 AfficheQuantitesSimple_2();
 AfficheInGadget(Tab_Quantite[selection-1].nom, &G_E_Sel,5,3,1,NO_COUL_FOND,
                 &TextAttr,15);
 if(selection<max_quantite_affiche)
    SetImgInGadget(&G_E_Qte,&img_qte[selection-1]);
 else
    SetImgInGadget(&G_E_Qte,OLD_QTE);
 RaffraichitEGadget();
 Prononce(Tab_Quantite[selection-1].son);
}

PRIVATE void GestionDeroul(g)
struct Gadget *g;
{
 if(NBRE_QUANTITES<9)
   Gestion_simple_1(g);
 else
   Gestion_simple_2(g);
 RaffraichitEGadget();
}

PRIVATE void AfficheQuantites(void)
{
 if(First)
   selection++;
 if(NBRE_QUANTITES<9)
   AfficheQuantitesSimple_1();
 else
   AfficheQuantitesSimple_2();
 if(First)
 {
  selection--;
  First=False;
 }
```

```
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(!strcmp(id,"G_E_But"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    Prononce(msg_but.son);
   }
 else if((!strcmp(id,"G_E_FL_H"))||(!strcmp(id,"G_E_FL_B")))
   GestionDeroul(g);
 else if((!strcmp(id,"G_E_Sel"))||(!strcmp(id,"G_E_Qte")))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Tab_Quantite[selection-1].son);
    }
 else if((!strcmp(id,"G_E_Bliss"))||(!strcmp(id,"G_E_Img")))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Produit.son_nom);
    }
 else if(strcmp(id,"G_E_OK")==0)
    retval=True;
 return(retval);
}

/*-------------------------------------------------*/
/* Gestion du dixième écran : attente d'un évènement */
/*-------------------------------------------------*/
void ecran10(produit)
tproduitcompos produit;
{
 boolean Sortie=False;
 ULONG class;
 USHORT i;
 struct IntuiMessage *msg = NULL;
 First=True;
 is_Sel=0;
 Produit=produit;
 NBRE_QUANTITES=(short) Produit.qtecompos;
 if(NBRE_QUANTITES!=Produit.qtecompos)
   ++NBRE_QUANTITES;
 POSY_Sel=51;
 selection=0;
 ChargeProduit();
```

```
ChargeQuantites();
/*-------------------------------------*/
/* Instanciation des boutons de l'écran */
/*-------------------------------------*/
AjouteEGadget(&G_E_FL_H, gadget[12], "G_E_FL_H", 6, 32);
AjouteEGadget(&G_E_FL_B, gadget[13], "G_E_FL_B", 6, 190);
AjouteEGadget(&G_E_Img, gadget[23], "G_E_Img", 150, 34);
AjouteEGadget(&G_E_Qte, gadget[26], "G_E_Qte", 150, 128);
AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150,162);
AjouteEGadget(&G_E_Bliss, gadget[24], "G_E_Bliss", 226, 44);
strcpy(msg_but.nom,tab_msg[35].nom);
strcat(msg_but.nom,"#");
strcat(msg_but.nom,Tab_Quantite[NBRE_QUANTITES].nom);
if(Produit.cond_util!=0)
{
 strcat(msg_but.nom,"#");
 strcat(msg_but.nom,Tab_Article[Produit.article].nom);
 strcat(msg_but.nom," ");
 strcat(msg_but.nom,Produit.nom);
}
strcpy(msg_but.son,tab_msg[35].son);
strcat(msg_but.son," ");
strcat(msg_but.son,Tab_Quantite[NBRE_QUANTITES].son);
if(Produit.cond_util!=0)
{
 strcat(msg_but.nom," ");
 strcat(msg_but.son,Tab_Article[Produit.article].son);
 strcat(msg_but.nom," ");
 strcat(msg_but.son,Produit.son_nom);
}
AfficheBulle(msg_but.nom,&G_E_But);
OLD_IMAGE=GetImgInGadget(&G_E_Img);
OLD_BLISS=GetImgInGadget(&G_E_Bliss);
OLD_QTE=GetImgInGadget(&G_E_Qte);
SetImgInGadget(&G_E_Img,&image);
SetImgInGadget(&G_E_Bliss,&bliss);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeImage("deroul_fond.pic",&imagefond);
/*----------------------*/
/* Affichage de l'écran */
/*----------------------*/
AfficheFond();
AfficheTitre("QUANTITES");
selection=0;
AfficheQuantites();
RaffraichitEGadget();
Prononce(msg_but.son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
while((Sortie==False)&&(NBRE_QUANTITES!=0))
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
```

```
 {
  class=msg->Class;
  if(class==GADGETUP)
   Sortie=GestionEvent(msg);
  ReplyMsg((struct Message *) msg);
 }
}
/*----------------------
   Libération des images
  ----------------------*/
LibereImage(&image);
LibereImage(&bliss);
for(i=0;i<NBRE_QUANTITES+1;i++)
{
 LibereImage(&img_qte[i]);
}
/*---------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*---------------------------------------*/
SetImgInGadget(&G_E_Img,OLD_IMAGE);
SetImgInGadget(&G_E_Bliss,OLD_BLISS);
SetImgInGadget(&G_E_Bliss,OLD_QTE);
SupprimeEGadget(&G_E_Bliss);
SupprimeEGadget(&G_E_FL_H);
SupprimeEGadget(&G_E_FL_B);
SupprimeEGadget(&G_E_Img);
SupprimeEGadget(&G_E_But);
SupprimeEGadget(&G_E_OK);
SupprimeEGadget(&G_E_Qte);
if(is_Sel) SupprimeEGadget(&G_E_Sel);
}
```

```
/*
 * ---> Module  : Ecran11
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Gadget.h"
#include "H_Types.h"
#include "H_Types2.h"
#include "H_CTypes.h"

PUBLIC struct Gadget gadget[max_gadgets];
PUBLIC struct TextAttr TextAttr;
PUBLIC st_msg tab_msg[nbre_msg];
PUBLIC tarticle Tab_Article[12];

PRIVATE struct Gadget G_E_OK, G_E_Oui, G_E_Non, G_E_Bliss, G_E_But, G_E_Img;
PRIVATE boolean selection, is_OK;
PRIVATE struct Image image, bliss, *OLD_IMAGE, *OLD_BLISS;
PRIVATE st_msg msg_but;
PRIVATE tproduitcompos Produit;

PRIVATE void ChargeProduit(void)
{
 char nom_img[15], nom_bliss[20];
 strcpy(nom_img,"produit");
 strcat(nom_img,itoa(Produit.no_produit));
 strcpy(nom_bliss,nom_img);
 strcat(nom_img,".pic");
 strcat(nom_bliss,".bliss");
 ChargeImage(nom_img,&image);
 ChargeImage(nom_bliss,&bliss);
}

/*-----------------------------------*/
/* Fonction de gestion des évènements */
/*-----------------------------------*/
PRIVATE boolean GestionEvent(m)
struct IntuiMessage *m;
{
 struct Gadget *g;
 char *id;
 boolean retval=False;
 g=(struct Gadget *) m->IAddress;
 id=IdGadget(g);
 if(eval(id,"=","G_E_But"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    Prononce(msg_but.son);
   }
 else if(eval(id,"=","G_E_Oui"))
 {
  DeselectionneGadget(&G_E_Non);
  selection=True;
  if(!is_OK)
  {
   AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
```

```
   is_OK=True;
  }
  RaffraichitEGadget();
  Prononce("UWIX");
 }
 else if(eval(id,"=","G_E_Non"))
 {
  DeselectionneGadget(&G_E_Oui);
  selection=False;
  if(!is_OK)
  {
   AjouteEGadget(&G_E_OK, gadget[4], "G_E_OK", 140, 210);
   is_OK=True;
  }
  RaffraichitEGadget();
  Prononce("NOHN");
 }
 else if(eval(id,"=","G_E_Bliss")||eval(id,"=","G_E_Img"))
   {
    DeselectionneGadget(g);
    RaffraichitEGadget();
    if(selection!=0)
      Prononce(Produit.son_nom);
    else;
    }
 else if(eval(id,"=","G_E_OK"))
    retval=True;
 return(retval);
}

/*-----------------------------------------------------*/
/* Gestion du onzième écran : attente d'un évènement */
/*-----------------------------------------------------*/
boolean ecran11(produit)
tproduitcompos produit;
{
 boolean Sortie=False;
 ULONG class;
 struct IntuiMessage *msg = NULL;
 Produit=produit;
 is_OK=False;
 /*-------------------------------------*/
 /* Instanciation des boutons de l'écran */
 /*-------------------------------------*/
 AjouteEGadget(&G_E_Oui, gadget[5], "G_E_Oui", 6, 63);
 AjouteEGadget(&G_E_Non, gadget[6], "G_E_Non", 74, 63);
 AjouteEGadget(&G_E_Img, gadget[23], "G_E_Img", 150, 34);
 AjouteEGadget(&G_E_But, gadget[3], "G_E_But", 150, 162);
 AjouteEGadget(&G_E_Bliss, gadget[24], "G_E_Bliss", 230, 44);
 strcpy(msg_but.nom,tab_msg[34].nom);
 strcpy(msg_but.son,tab_msg[34].son);
 strcat(msg_but.nom," ");
 strcat(msg_but.nom,Tab_Article[Produit.article].nom);
 strcat(msg_but.son,Tab_Article[Produit.article].son);
 strcat(msg_but.nom," ");
 strcat(msg_but.son," ");
 strcat(msg_but.nom,produit.nom);
 strcat(msg_but.son,produit.son_nom);
 AfficheBulle(msg_but.nom,&G_E_But);
```

```
OLD_IMAGE=GetImgInGadget(&G_E_Img);
OLD_BLISS=GetImgInGadget(&G_E_Bliss);
/*----------------------
   Chargement des images
  ----------------------*/
ChargeProduit();
SetImgInGadget(&G_E_Img,&image);
SetImgInGadget(&G_E_Bliss,&bliss);
/*---------------------*/
/* Affichage de l'écran */
/*---------------------*/
AfficheFond();
AfficheTitre("LISTE D'ACHATS");
selection=0;
RaffraichitEGadget();
Prononce(tab_msg[33].son);
/*-----------------------------------------------*/
/* Répeter                                       */
/*    Attente d'un évènement                     */
/*    envoi à la routine de gestion des évènements */
/* Tant que Sortie=False                         */
/*-----------------------------------------------*/
while(Sortie==False)
{
 Wait(1L << Window->UserPort->mp_SigBit);
 while(msg=(struct InutiMessage *)GetMsg(Window->UserPort))
 {
  class=msg->Class;
  if(class==GADGETUP)
   Sortie=GestionEvent(msg);
  ReplyMsg((struct Message *) msg);
 }
}
/*----------------------
   Libération des images
  ----------------------*/
 LibereImage(&image);
 LibereImage(&bliss);
/*--------------------------------------*/
/* Désinstanciation des boutons de l'écran */
/*--------------------------------------*/
SetImgInGadget(&G_E_Img,OLD_IMAGE);
SetImgInGadget(&G_E_Bliss,OLD_BLISS);
SupprimeEGadget(&G_E_Bliss);
SupprimeEGadget(&G_E_Img);
SupprimeEGadget(&G_E_But);
SupprimeEGadget(&G_E_Oui);
SupprimeEGadget(&G_E_Non);
SupprimeEGadget(&G_E_OK);
return(selection);
}
```

```
/*
 * Module  : Filtrage collection
 * Type    : Source complète
 * Auteurs : Topet Léopold
 *           Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*********************************/
/*  MODULE FILTRAGE COLLECTION  */
/*********************************/

/************************************************
void fournir_preparations(tab_preparations,nombre)
************************************************

tpreparation tab_preparations[];
short *nombre;

{
 tpreparation  art_preparation;
 short statutbd;

 *nombre=0;
 statutbd=premier_seq_preparation(&art_preparation);
 while (statutbd==F_OK)
       {
       copier_article((char *)&tab_preparations[*nombre],
                 (char *)&art_preparation,sizeof(tpreparation));
       ++ *nombre;
        statutbd=suivant_seq_preparation(&art_preparation);
       }
 return;
}

/***********************************/
void fournir_repas(tab_repas,nombre)
/***********************************/

trepas tab_repas[];
short *nombre;

{
 trepas art_repas;
 short statutbd;

 *nombre=0;
 statutbd=premier_seq_repas(&art_repas);
 while (statutbd==F_OK)
       {
   copier_article((char *)&tab_repas[*nombre],
                 (char *)&art_repas,sizeof(trepas));
   ++ *nombre;
   statutbd=suivant_seq_repas(&art_repas);
       }
 return;
}
```

```
/*****************************************/
void fournir_themes_et_fils(tab_themes,nombre)
/*****************************************/

ttheme_pere_fils tab_themes[];
short *nombre;

{
 ttheme tab_themes_temp[maxfilstheme+1];
 short taille,statutbd;

 *nombre=0;
 statutbd=premier_seq_theme_et_fils(tab_themes_temp,&taille);
 while (statutbd==F_OK)
       {
        copier_theme_et_fils(tab_themes,tab_themes_temp,*nombre,taille);
        tab_themes[*nombre].nbr_fils=taille-1;
       *nombre+=taille;
        statutbd=suivant_seq_theme_et_fils(tab_themes_temp,&taille);
       }
 return;
}

/*************************************************************************/
void fournir_menu_complet_sur_cle1(date,repas,nbr_pers,tab_const,nbr_const,
                                   tab_plats,nbr_plats,existe)
/*************************************************************************/

tdate date;
short repas,*nbr_pers,*nbr_const,*nbr_plats;
tconstituants_ext tab_const[];
tplat_pere_fils tab_plats[];
boolean *existe;

{
 tmenu art_menu;
 ttheme art_theme;
 tplat art_plat,tab_plats_fils[maxfilstheme+1];
 boolean decomposable;
 short nombre,statutbd;

 *nbr_const=*nbr_plats=0;
 if (menu_sur_id1(date,repas,&art_menu)==F_OK)
    {
     *existe=True;
     *nbr_pers=art_menu.nnbr_pers;
     statutbd=premier_themeplat_sur_cle1(date,repas,&art_theme,
                                                    &art_plat);
     while (statutbd==F_OK)
           {
       copier_article((char *)&tab_const[*nbr_const],
                      (char *)&art_theme,sizeof(ttheme));
       verifier_theme_origine_thth(art_theme.no_theme,&decomposable,
                                   statutbd);
       if (decomposable==True)
          {
           fournir_plats_sur_cle2(art_plat.no_plat,
                                  tab_plats_fils,&nombre);
          copier_plats_fils(tab_plats,tab_plats_fils,
```

```
                              (*nbr_plats+1),nombre);
          tab_plats[*nbr_plats].nbr_fils=nombre;
          tab_plats[*nbr_plats].no_plat=art_plat.no_plat;
          tab_const[*nbr_const].ptr_plat=*nbr_plats;
          *nbr_plats+=(nombre+1);
         }
         else
         {
          copier_article((char *)&tab_plats[*nbr_plats],
                    (char *)&art_plat,sizeof(tplat));
          tab_plats[*nbr_plats].nbr_fils=0;
               tab_const[*nbr_const].ptr_plat=*nbr_plats;
          ++ *nbr_plats;
         }
       ++ *nbr_const;
       statutbd=suivant_themeplat_sur_cle1(date,repas,
                                        &art_theme,
                                   &art_plat);
      }
     }
     else *existe=False;
 return;
}

/*************************************************************/
void fournir_plats_et_fils_sur_cle1(theme,tab_plats,nbr_plats)
/*************************************************************/

short theme,*nbr_plats;
tplat_pere_fils tab_plats[];

{
 tplat art_plat,tab_plats_fils[maxfilstheme+1];
 boolean decomposable;
 short nombre,statutbd;

 *nbr_plats=0;
 if (verifier_theme_origine_thth(theme,&decomposable)==F_OK)
    {
     if (premier_plat_sur_cle1(theme,&art_plat)==F_OK)
        {
        statutbd=F_OK;
        while (statutbd==F_OK)
              {
               if (decomposable==True)
                  {
                   fournir_plats_sur_cle2(art_plat.no_plat,
                                          tab_plats_fils,&nombre);
                   copier_plats_fils(tab_plats,tab_plats_fils,
                                     (*nbr_plats+1),nombre);
                  tab_plats[*nbr_plats].nbr_fils=nombre;
                  tab_plats[*nbr_plats].no_plat=art_plat.no_plat;
                   strcpy(tab_plats[*nbr_plats].nom,'\0');
                   strcpy(tab_plats[*nbr_plats].son,'\0');
                   *nbr_plats+=(nombre+1);
                  }
                else
                  {
                   copier_article((char *)&tab_plats[*nbr_plats],
```

```
                                (char *)&art_plat,sizeof(tplat));
                tab_plats[*nbr_plats].nbr_fils=0;
                ++ *nbr_plats;
               }
          statutbd=suivant_plat_sur_cle1(theme,&art_plat);
         }
     }
    }
 return;
}

/*************************************************/
void fournir_menu_prep_repas(tab_menupreprep,nombre)
/*************************************************/

tmenupreprep tab_menupreprep[];
short *nombre;

{
 tmenupreprep art_menupreprep;
 short statutbd;

 *nombre=0;
 statutbd=premier_seq_menupreprep(&art_menupreprep);
 while (statutbd==F_OK)
       {
   copier_article((char *)&tab_menupreprep[*nombre],
                (char *)&art_menupreprep,sizeof(tmenupreprep));
   ++ *nombre;
   statutbd=suivant_seq_menupreprep(&art_menupreprep);
       }
 return;
}

/*****************************************************************/
void fournir_produits_compos_sur_cle1(plat,tab_produits,nombre)
/*****************************************************************/

short plat,*nombre;
tproduitcompos tab_produits[];

{
 tproduitcompos art_produit;
 short statutbd;

 *nombre=0;
 statutbd=premier_produit_compos_sur_cle1(plat,&art_produit);
 while (statutbd==F_OK)
       {
        copier_article((char *)&tab_produits[*nombre],(char *)&art_produit,
                 sizeof(tproduitcompos));
        statutbd=suivant_produit_compos_sur_cle1(plat,&art_produit);
       ++ *nombre;
       }
}
```

```
/*****************************************************************/
void fournir_produits_menu(tab_plats,tab_plats_long,tab_produits,
                           tab_produits_long)
/*****************************************************************/

short tab_plats[],tab_plats_long,*tab_produits_long;
tproduitcompos tab_produits[];

{
 short i,j,temp_long;
 tproduitcompos tab_produits_temp[maxproduitsplat*maxthememenu];

 *tab_produits_long=0;
 for (i=0;i<tab_plats_long;++i)
     {
      fournir_produits_compos_sur_clel(tab_plats[i],tab_produits_temp,
                                       &temp_long);
      for (j=0;j<temp_long;++j)
        inserer_produit_compos(tab_produits,tab_produits_long,&tab_produits_temp[j]);
     }
}
```

```
/*
 * Module  : Filtrage collection (suite)
 * Type    : Source complète
 * Auteurs : Topet Léopold
 *           Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*****************************/
/*  ACCES EXTERNES (SUITE)  */
/*****************************/

PUBLIC FILE *fdptrmenuconst, *fdmenu;
PUBLIC char *disk_name;

/*******************************************/
short premier_seq_menupreprep(art_menupreprep)
/*******************************************/

tmenupreprep *art_menupreprep;

{
 tptr_menu_const art_ptr_menu_const;
 tmenuint art_temp;
 tmenu art_menu;
 short posit_acc;

 posit_acc=POS_DEBUT;
 if (lire_fichier(fdptrmenuconst,disk_name,(char *)&art_ptr_menu_const,
                sizeof(art_ptr_menu_const),&posit_acc)==F_OK)
    {
     posit_acc=art_ptr_menu_const.ptr_menu;
     lire_fichier(fdmenu,disk_name,(char *)&art_temp,
                 sizeof(art_temp),&posit_acc);
     convintprgmenu(&art_temp,&art_menu);
     copier_article((char *)art_menupreprep,(char *)&art_menu,
              sizeof(art_menu));
     art_menupreprep->date=art_ptr_menu_const.date;
     art_menupreprep->no_repas=art_ptr_menu_const.repas;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}

/*******************************************/
short suivant_seq_menupreprep(art_menupreprep)
/*******************************************/

tmenupreprep *art_menupreprep;

{
 tptr_menu_const art_ptr_menu_const;
 tmenuint art_temp;
 tmenu art_menu;
 short posit_acc;

 posit_acc=POS_COURANT;
 if (lire_fichier(fdptrmenuconst,disk_name,(char *)&art_ptr_menu_const,
```

```
                sizeof(art_ptr_menu_const),&posit_acc)==F_OK)
    {
     posit_acc=art_ptr_menu_const.ptr_menu;
     lire_fichier(fdmenu,disk_name,(char *)&art_temp,
                  sizeof(art_temp),&posit_acc);
     convintprgmenu(&art_temp,&art_menu);
     copier_article((char *)art_menupreprep,(char *)&art_menu,
               sizeof(art_menu));
     art_menupreprep->date=art_ptr_menu_const.date;
     art_menupreprep->no_repas=art_ptr_menu_const.repas;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}
```

```
/*
 * Fichier : Sous-filtrage (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_types2.h"
/************************************/
/*  SOUS_FILTRAGE COLLECTION (SUITE)  */
/************************************/

/*******************************************************/
void fournir_plats_sur_cle2(plat_pere,tab_plats,nombre)
/*******************************************************/

short plat_pere,*nombre;
tplat tab_plats[];

{
 tplat art_platfils;
 short statutbd;

 *nombre=0;
 statutbd=premier_plat_sur_cle2(plat_pere,&art_platfils);
 while (statutbd==F_OK)
       {
        copier_article((char *)&tab_plats[*nombre],(char *)&art_platfils,
                 sizeof(art_platfils));
        ++ *nombre;
   statutbd=suivant_plat_sur_cle2(plat_pere,&art_platfils);
        }
 return;
}

/*************************************************************/
void copier_plats_fils(tab_plats,tab_plats_fils,position,nombre)
/*************************************************************/

tplat_pere_fils tab_plats[];
tplat tab_plats_fils[];
short nombre,position;

{
 short i;

 for (i=0;i<nombre;++i)
     copier_article((char *)&tab_plats[position+i],
                    (char *)&tab_plats_fils[i],sizeof(tplat));
 return;
}
```

```
/*************************************************************************/
void copier_theme_et_fils(tab_themes,tab_theme_et_fils,position,nombre)
/*************************************************************************/

ttheme_pere_fils tab_themes[];
ttheme tab_theme_et_fils[];
short position,nombre;

{
 short i;

 for (i=0;i<nombre;++i)
     copier_article((char *)&tab_themes[position+i],
                    (char *)&tab_theme_et_fils[i],sizeof(ttheme));
 return;
}

/*********************************************************/
void inserer_produit_compos(tab_produits,tab_long,produit)
/*********************************************************/

tproduitcompos tab_produits[],*produit;
short *tab_long;

{
 boolean trouve;
 short i;

 trouve=False;
 for (i=0;((i<*tab_long) && (trouve==False));++i)
     {
      if (tab_produits[i].no_produit==produit->no_produit)
         {
          tab_produits[i].qtecompos+=produit->qtecompos;
          trouve=True;
         }
     }
 if (trouve==False)
    {
     copier_article((char *)&tab_produits[i],(char *)produit,
                    sizeof(tproduitcompos));
     ++ (*tab_long);
    }
}
```

```
/*
 * ---> Module  : Courses
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"

PUBLIC tmenupreprep tab_menupreprep[max_eloignement*maxrepas];
PUBLIC short nbrmenupreprep;
/*************/
/*  COURSES  */
/*************/

/*********************************************************************/
void prod_qte_nec_achat_fait(art_menupreprep,tab_prod_nec,tab_long)
/*********************************************************************/

tmenupreprep *art_menupreprep;
tproduitcompos tab_prod_nec[];
short *tab_long;

{
 short i,j,k,tab_prod_temp_long,nbr_pers,nbr_const,nbr_plats,borne_sup,
       diff_pers;
 tproduitcompos tab_prod_temp[maxproduitsplat];
 boolean existe;
 tconstituants_ext tab_const[maxthememenu];
 tplat_pere_fils tab_plats[maxthememenu-maxthemedec+(maxthemedec*
                           maxfilstheme)];
 tconstitution art_constitution;

 if (art_menupreprep->modifie==True)
    {
     fournir_menu_complet_sur_cle1(art_menupreprep->date,
                                   art_menupreprep->no_repas,
                                   &nbr_pers,tab_const,&nbr_const,tab_plats,
                                   &nbr_plats,&existe);
     for (i=0;i<nbr_const;++i)
         {
          borne_sup=tab_plats[tab_const[i].ptr_plat].nbr_fils+1;
          constitution_sur_id1(art_menupreprep->date,
                               art_menupreprep->no_repas,
                               &art_constitution);
          for (j=0;j<borne_sup;j++)
              {
               if (tab_plats[tab_const[i].ptr_plat+j].nbr_fils==0)
                  {
                   if (art_constitution.modifie_plat='A')
                      {
                       fournir_produits_compos_sur_cle1(
                                     tab_plats[i].no_plat,
                                     tab_prod_temp,&tab_prod_temp_long);
                       for (k=0;k<tab_prod_temp_long;k++)
                           {
                            tab_prod_temp[k].qtecompos*=art_menupreprep->
                                                                 nnbr_pers;
                            inserer_produit_compos(tab_prod_nec,tab_long,
```

```
                                                &tab_prod_temp[k]);
                        }
                    }
                    else
                    {
                     if (art_menupreprep->nnbr_pers >
                         art_menupreprep->nbr_pers)
                        {
                         diff_pers=(art_menupreprep->nnbr_pers -
                                    art_menupreprep->nbr_pers);
                         fournir_produits_compos_sur_cle1(
                                     tab_plats[i].no_plat,
                                     tab_prod_temp,
                                     &tab_prod_temp_long);
                         for (k=0;k<tab_prod_temp_long;k++)
                             {
                              tab_prod_temp[k].qtecompos*=diff_pers;
                              inserer_produit_compos(tab_prod_nec,
                                                     tab_long,
                                                     &tab_prod_temp[k]);
                             }
                        }
                    }
                }
            }
        }
    }
}

/*********************************************************************/
void prod_qte_nec_achat_pas_fait(art_menupreprep,tab_prod_nec,tab_long)
/*********************************************************************/

tmenupreprep *art_menupreprep;
tproduitcompos tab_prod_nec[];
short *tab_long;

{
 short i,j,tab_prod_temp_long,nbr_pers,nbr_const,nbr_plats;
 tproduitcompos tab_prod_temp[maxproduitsplat];
 boolean existe;
 tconstituants_ext tab_const[maxthememenu];
 tplat_pere_fils tab_plats[maxthememenu-maxthemedec+(maxthemedec*
                           maxfilstheme+1)];

 fournir_menu_complet_sur_cle1(art_menupreprep->date,
                               art_menupreprep->no_repas,
                               &nbr_pers,tab_const,&nbr_const,tab_plats,
                               &nbr_plats,&existe);
 for (i=0;i<nbr_plats;i++)
     {
      if (tab_plats[i].nbr_fils==0)
         {
          fournir_produits_compos_sur_cle1(tab_plats[i].no_plat,
                                           tab_prod_temp,
                                           &tab_prod_temp_long);
          for (j=0;j<tab_prod_temp_long;j++)
              {
               tab_prod_temp[j].qtecompos*=art_menupreprep->nnbr_pers;
               inserer_produit_compos(tab_prod_nec,tab_long,
```

```
                                        &tab_prod_temp[j]);
                }
            }
        }
}


/*********************************************************************/
void determiner_prod_qte_nec_menu_ss(periode,tab_prod_nec,tab_long)
/*********************************************************************/

tperiode_achat *periode;
tproduitcompos tab_prod_nec[];
short *tab_long;

{
 short i,j;

 *tab_long=0;
 for (i=0;((i<nbrmenupreprep)&&(compare_date(tab_menupreprep[i].date,
                                        periode->debut,'<')==True));++i);
 for (j=i;((j<nbrmenupreprep)&&(compare_date(tab_menupreprep[j].date,
                                        periode->fin,'>')==False));++j)
     {if (periode->achat_fait==True)
          prod_qte_nec_achat_fait(&tab_menupreprep[j],tab_prod_nec,
                                  tab_long);
          else
          prod_qte_nec_achat_pas_fait(&tab_menupreprep[j],tab_prod_nec,
                                        tab_long);
     }
}

/*********************************************************************************/
void deter_prod_qte_nec_menu_glo(tab_ss_periode,
                                 tab_ss_periode_long,tab_prod_nec,
                                 tab_prod_nec_long)
/*********************************************************************************/

tperiode_achat tab_ss_periode[];
short          tab_ss_periode_long,*tab_prod_nec_long;
tproduitcompos    tab_prod_nec[];

{
 short i,j,tab_temp_long;
 tproduitcompos tab_prod_nec_temp[maxproduitsplat*maxrepas*
                                  max_periode_a_traiter*maxthememenu];

 /* fournir_menu_prep_rep*/
 for (i=0;i<tab_ss_periode_long;i++)
     {
      determiner_prod_qte_nec_menu_ss(&tab_ss_periode[i],
                                        tab_prod_nec_temp,
                                        &tab_temp_long);
      for(j=0;j<tab_temp_long;j++)
         inserer_produit_compos(tab_prod_nec,tab_prod_nec_long,
                                &tab_prod_nec_temp[j]);
     }
}
```

```
/*
 * ---> Module  : Macro menu
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*************************************/
/*  MACRO PRIMITIVES DE MODIFICATIONS  */
/*************************************/

PUBLIC short nbrmenupreprep;
PUBLIC tmenupreprep tab_menupreprep[(max_eloignement*maxrepas)];


/*****************************************************************************/
void enregistrer_menu_complet(date,repas,nbr_pers,constituants,nbr_const)
/*****************************************************************************/

tdate date;
short repas,nbr_pers,nbr_const;
tconstituants_enreg constituants[];

{
 short i;

 creer_preparation(date,repas);
 creer_menu(date,repas,nbr_pers);
 for (i=0;(i<nbr_const);++i)
      creer_constitution(date,repas,constituants[i].theme,
                  constituants[i].plat);
 return;
}

/*****************************************************************************/
void modifier_menu_complet(date,repas,nbr_pers,const_a_enlever,
                           nbr_a_enlever,const_a_ajouter,nbr_a_ajouter)
/*****************************************************************************/

tdate date;
short repas,nbr_pers,nbr_a_enlever,nbr_a_ajouter;
tconstituants_enreg const_a_enlever[],const_a_ajouter[];

{
 short i;
 tconstitution art_constitution;

 for (i=0;(i<nbr_a_enlever);++i)
     maj_const_modifieplat_sur_id1(date,repas,
                                       const_a_enlever[i].theme,'E');

 for (i=0;(i<nbr_a_ajouter);++i)
     {
      if (constitution_sur_id1(date,repas,
                                      const_a_ajouter[i].theme,
                                      &art_constitution)==F_OK)
    maj_constitution_plat_sur_id1(const_a_ajouter[i].plat,date,repas,
                                    const_a_ajouter[i].theme,'A');
```

```
    else
        creer_constitution(date,repas,const_a_ajouter[i].theme,
                           const_a_ajouter[i].plat);
     }
 maj_menu_pers_modifie_sur_id1(date,repas,nbr_pers,True);
 return;
}

/*******************************/
void supprimer_menu_complet(date)
/*******************************/

tdate date;

{
 short i;
 boolean poursuite,compare_date();

 poursuite=True;
 i=0;
 while((i<nbrmenupreprep) && (poursuite==True))
 {
  if(compare_date(tab_menupreprep[i].date,date,'<')==True)
  {
   suppression_constitution_sur_cle1(tab_menupreprep[i].date,
                                     tab_menupreprep[i].no_repas);
   suppression_menu_sur_id1(tab_menupreprep[i].date,
                            tab_menupreprep[i].no_repas);
   suppression_preparation_sur_id1(tab_menupreprep[i].date,
                                   tab_menupreprep[i].no_repas);
   i++;
  }
  else poursuite=False;
 }
 return;
}
```

```
/*
 * Module  : Opérations externes
 * Type    : Source complète
 * Auteurs : Topet Léopold
 *           Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*************************/
/*  OPERATIONS EXTERNES  */
/*************************/

/********************/
/*  ACCES EXTERNES  */
/********************/

/************************************/
short menu_sur_id1(date,repas,art_menu)
/************************************/

tdate date;
short repas;
tmenu *art_menu;

{
 tmenuint art_temp;

 if (menu_int_sur_id1(date,repas,&art_temp)==F_OK)
    {
     convintprgmenu(&art_temp,art_menu);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}

/*********************************/
short plat_sur_id1(no_plat,art_plat)
/*********************************/

short no_plat;
tplat *art_plat;

{
 tplatint art_temp;

 if (plat_int_sur_id1(no_plat,&art_temp)==F_OK)
    {
     convintprgplat(&art_temp,art_plat);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/************************************/
short theme_sur_id1(no_theme,art_theme)
/************************************/

short no_theme;
ttheme *art_theme;

{
 tthemeint art_temp;

 if (theme_int_sur_id1(no_theme,&art_temp)==F_OK)
    {
     convintprgtheme(&art_temp,art_theme);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}

/************************************************************/
short constitution_sur_id1(date,repas,theme,art_constitution)
/************************************************************/

tdate date;
short repas,theme;
tconstitution *art_constitution;

{
 tconstitutionint art_temp;

 if (constitution_int_sur_id1(date,repas,theme,&art_temp)==F_OK)
    {
     convintprgconstitution(&art_temp,art_constitution);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}

/*************************************/
short produit_sur_id1(numero,art_produit)
/*************************************/

short numero;
tproduitcompos *art_produit;

{
 tproduitint art_produit_temp;

 if (produit_int_sur_id1(numero,&art_produit_temp)==F_OK)
    {
     convintprgproduit(&art_produit_temp,art_produit);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/************************/
/*  CREATIONS EXTERNES  */
/************************/

/*********************************/
void creer_preparation(date,repas)
/*********************************/

tdate date;
short repas;
{
 tpreparation art_preparation;
 tpreparationint art_temp;

 convprgintpreparation(&art_preparation,&art_temp);
 creer_preparation_int(&art_temp);
 return;
}

/*************************************/
void creer_menu(date,repas,nbr_personnes)
/*************************************/

tdate date;
short repas,nbr_personnes;

{
 tmenu art_menu;
 tmenuint art_temp;

 art_menu.nbr_pers=nbr_personnes;
 art_menu.nnbr_pers=nbr_personnes;
 art_menu.modifie=True;
 art_menu.imprime=False;
 convprgintmenu(&art_menu,&art_temp);
 creer_menu_int(date,repas,&art_temp);
 return;
}

/*****************************************/
void creer_constitution(date,repas,theme,plat)
/*****************************************/

tdate date;
short repas,theme,plat;

{
 tconstitution art_constitution;
 tconstitutionint art_temp;

 art_constitution.modifie_plat='A';
 convprgintconstitution(&art_constitution,theme,plat,infini,&art_temp);
 creer_constitution_int(date,repas,&art_temp);
 return;
}
```

```
/**************************/
/*  MISES-A-JOUR EXTERNES  */
/**************************/

/*****************************************************************/
short maj_menu_pers_modifie_sur_id1(date,repas,nbr_pers,modifie)
/*****************************************************************/

tdate date;
short repas,nbr_pers;
boolean modifie;

{
 return(maj_menu_int_pers_modif_sur_id1(date,repas,nbr_pers,modifie));
}

/***********************************************/
short maj_menu_imprime_sur_id1(date,repas,imprime)
/***********************************************/

tdate date;
short repas;
boolean imprime;

{
 return(maj_menu_int_imprime_sur_id1(date,repas,imprime));
}

/*************************************************************************/
short maj_const_modifieplat_sur_id1(date,repas,theme,modifie_plat)
/*************************************************************************/

tdate date;
short repas,theme;
char modifie_plat;

{
 return(maj_const_int_modifieplat_sur_id1(date,repas,theme,modifie_plat));
}

/*********************************************************************/
short maj_constitution_plat_sur_id1(plat,date,repas,theme,modif_plat)
/*********************************************************************/

short plat,repas,theme;
tdate date;
char modif_plat;

{
 return(maj_const_int_plat_sur_id1(plat,date,repas,theme,modif_plat));
}
```

```
/************************/
/* SUPPRESSIONS EXTERNES */
/************************/

/*********************************************/
short suppression_preparation_sur_id1(date,repas)
/*********************************************/

tdate date;
short repas;

{
 return(effacer_preparation_int_sur_id1(date,repas));
}

/*************************************/
short suppression_menu_sur_id1(date,repas)
/*************************************/

tdate date;
short repas;

{
 return(effacer_menu_int_sur_id1(date,repas));
}

/*********************************************/
short suppression_constitution_sur_cle1(date,repas)
/*********************************************/

tdate date;
short repas;

{
 return(effacer_constitution_int_sur_cle1(date,repas));
}

/*****************************/
/*  VERIFICATIONS DIVERSES  */
/*****************************/

/*********************************************/
short verifier_theme_origine_thth(theme,origine)
/*********************************************/

short theme;
boolean *origine;

{
 tthemeint art_theme;
 boolean conventbool();

 if (theme_int_sur_id1(theme,&art_theme)==F_OK)
    {
     *origine=conventbool(art_theme.a_fils);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/*
 * Module   : Opérations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
           *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
           *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit;
PUBLIC char *disk_name;
/*****************************/
/*  ACCES EXTERNES (SUITE)  */
/*****************************/

static tconstitutionint art_const;

/*************************************************************/
short premier_themeplat_sur_cle1(date,repas,art_theme,art_plat)
/*************************************************************/

tdate date;
short repas;
ttheme *art_theme;
tplat *art_plat;

{
 tptr_menu_const art_ptr_menu_const;
 short posit_acc;
 boolean trouve,fin;

 trouve=False;
 fin=False;
 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     posit_acc=art_ptr_menu_const.ptr_const;
     lire_fichier(fdconst,disk_name,(char *)&art_const,
                  sizeof(tconstitutionint),&posit_acc);
     while ((trouve==False) && (fin==False))
           {
       if (art_const.modifie_plat!='E')
              trouve=True;
              else
              {
               if (art_const.ptr_suivant==infini)
                  fin=True;
                  else
                  {
                   posit_acc=art_const.ptr_suivant;
                   lire_fichier(fdconst,disk_name,(char *)&art_const,
                                sizeof(tconstitutionint),&posit_acc);
                  }

              }
           }
     if (trouve==True)
        {
```

```
        theme_sur_id1(art_const.no_theme,art_theme);
        plat_sur_id1(art_const.no_plat,art_plat);
    return(F_OK);
       }
       else return(F_FIN_FICH);
    }
    else return(F_FIN_FICH);
}

/*************************************************************/
short suivant_themeplat_sur_cle1(date,repas,art_theme,art_plat)
/*************************************************************/

tdate date;
short repas;
ttheme *art_theme;
tplat *art_plat;

{
 short posit_acc;
 boolean trouve,fin;

 trouve=fin=False;
 posit_acc=art_const.ptr_suivant;
 if(posit_acc==infini)
   return(F_FIN_FICH);
 else
 {
 lire_fichier(fdconst,disk_name,(char *)&art_const,
             sizeof(tconstitutionint),&posit_acc);
 while ((trouve==False) && (fin==False))
       {
   if (art_const.modifie_plat!='E')
          trouve=True;
          else
          {
           if (art_const.ptr_suivant==infini)
               fin=True;
               else
               {
                posit_acc=art_const.ptr_suivant;
                lire_fichier(fdconst,disk_name,(char *)&art_const,
                            sizeof(tconstitutionint),&posit_acc);
               }
          }
       }
    if (trouve==True)
       {
        theme_sur_id1(art_const.no_theme,art_theme);
        plat_sur_id1(art_const.no_plat,art_plat);
    return(F_OK);
       }
       else return(F_FIN_FICH);
}
}
```

```
/*
 * Module   : Opérations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Privacy.h"
#include "H_PrimBD.h"
#include "H_Types2.h"
/****************************/
/*  ACCES EXTERNES (SUITE)  */
/****************************/
PUBLIC FILE *fdtheme;
PUBLIC char *disk_name;
static short posit_acc;

/*************************************************/
short premier_seq_theme_et_fils(tab_themes,nombre)
/*************************************************/

ttheme tab_themes[];
short *nombre;

{
 boolean fin_fils;
 tthemeint art_theme_temp;
 short statutbd;

 *nombre=0;
 posit_acc=POS_DEBUT;
 if (lire_fichier(fdtheme,disk_name,(char *)&art_theme_temp,
                 sizeof(tthemeint),&posit_acc)==F_OK)
    {
     convintprgtheme(&art_theme_temp,&tab_themes[*nombre]);
     if (art_theme_temp.a_fils==1)
        {
         statutbd=F_OK;
         fin_fils=False;
         while ((fin_fils==False) && (statutbd==F_OK))
               {
                posit_acc=POS_COURANT;
               statutbd=lire_fichier(fdtheme,disk_name,
                                     (char *)&art_theme_temp,
                                      sizeof(tthemeint),&posit_acc);
                if (statutbd==F_OK)
                   {
                    if (art_theme_temp.no_theme_pere==tab_themes[0].
                                                      no_theme)
                       {
                        ++ *nombre;
                        convintprgtheme(&art_theme_temp,
                                        &tab_themes[*nombre]);
                       }
                       else fin_fils=True;
                   }
               }
        }
       else ++posit_acc;
```

```
     ++ *nombre;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}

/*********************************************/
short suivant_seq_theme_et_fils(tab_themes,nombre)
/*********************************************/

ttheme tab_themes[];
short *nombre;

{
 boolean fin_fils;
 tthemeint art_theme_temp;
 short statutbd;

 *nombre=0;
 if (lire_fichier(fdtheme,disk_name,(char *)&art_theme_temp,
                  sizeof(tthemeint),&posit_acc)==F_OK)
    {
     convintprgtheme(&art_theme_temp,&tab_themes[*nombre]);
     if (art_theme_temp.a_fils==1)
        {
         statutbd=F_OK;
         fin_fils=False;
         while ((fin_fils==False) && (statutbd==F_OK))
               {
                posit_acc=POS_COURANT;
               statutbd=lire_fichier(fdtheme,disk_name,
                                     (char *)&art_theme_temp,
                                     sizeof(tthemeint),&posit_acc);
                if (statutbd==F_OK)
                   {
                    if (art_theme_temp.no_theme_pere==tab_themes[0].
                                                      no_theme)
                       {
                        ++ *nombre;
                        convintprgtheme(&art_theme_temp,
                                        &tab_themes[*nombre]);
                       }
                       else fin_fils=True;
                   }
               }
        }
       else ++posit_acc;
     ++ *nombre;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}
```

```
/*
 * Module   : Opérations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/***************************/
/*  ACCES EXTERNES (SUITE)  */
/***************************/
PUBLIC FILE *fdptrmenuconst;
PUBLIC char *disk_name;

/*****************************************/
short premier_seq_preparation (art_preparation)
/*****************************************/

tpreparation *art_preparation;

{
 tptr_menu_const art_ptr_menu_const;
 short posit_acc;

 posit_acc=POS_DEBUT;
 if (lire_fichier(fdptrmenuconst,disk_name,(char *)&art_ptr_menu_const,
                  sizeof(tptr_menu_const),&posit_acc)==F_OK)
    {
     art_preparation->date=art_ptr_menu_const.date;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}

/*****************************************/
short suivant_seq_preparation (art_preparation)
/*****************************************/

tpreparation *art_preparation;

{
 tptr_menu_const art_ptr_menu_const;
 short posit_acc;

 posit_acc=POS_COURANT;
 if (lire_fichier(fdptrmenuconst,disk_name,(char *)&art_ptr_menu_const,
                  sizeof(tptr_menu_const),&posit_acc)==F_OK)
    {
     art_preparation->date=art_ptr_menu_const.date;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}
```

```
/*
 * Module   : Opérations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/****************************/
/*  ACCES EXTERNES (SUITE)  */
/****************************/
PUBLIC FILE *fdptrcomposplat, *fdproduit, *fdcomposplat;
PUBLIC char *disk_name;

static short i;
static tptr_compos_plat art_ptr_compos_plat;

/*****************************************************/
short premier_produit_compos_sur_cle1(plat,art_produit)
/*****************************************************/

short plat;
tproduitcompos *art_produit;

{
 tcompos_platint art_compos_plat;
 short posit_acc;
 tproduitcompos art_produit_temp;

 i=1;
 if (ptr_compos_plat_sur_id1(plat,&art_ptr_compos_plat)==F_OK)
    {
     posit_acc=art_ptr_compos_plat.ptr_compos_plat;
     lire_fichier(fdcomposplat,disk_name,(char *)&art_compos_plat,
                  sizeof(tcompos_platint),&posit_acc);
     produit_sur_id1(art_compos_plat.no_produit,&art_produit_temp);
     copier_article(art_produit,&art_produit_temp,sizeof(tproduitcompos));
     art_produit->qtecompos=art_compos_plat.qte_produit;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}

/*****************************************************/
short suivant_produit_compos_sur_cle1(plat,art_produit)
/*****************************************************/

short plat;
tproduitcompos *art_produit;

{
 tcompos_platint art_compos_plat;
 short posit_acc;
 tproduitcompos art_produit_temp;

 if (i<art_ptr_compos_plat.nbr_produits)
    {
     posit_acc=POS_COURANT;
```

```
    lire_fichier(fdcomposplat,disk_name,(char *)&art_compos_plat,
                 sizeof(tcompos_platint),&posit_acc);
    produit_sur_id1(art_compos_plat.no_produit,&art_produit_temp);
    copier_article(art_produit,&art_produit_temp,sizeof(tproduitcompos));
    art_produit->qtecompos=art_compos_plat.qte_produit;
    ++ i;
    return(F_OK);
   }
   else return(F_FIN_FICH);
}
```

```
/*
 * Module   : Opérations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_types2.h"
/****************************/
/*  ACCES EXTERNES (SUITE)  */
/****************************/

static short i;
static tptr_appartenance art_ptr_appartenance;
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
            *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
            *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit, *fdapp;
PUBLIC char *disk_name;

/***************************************/
short premier_plat_sur_cle1(theme,art_plat)
/***************************************/

short theme;
tplat *art_plat;

{
 tappartenanceint art_appartenance;
 short posit_acc;

 i=1;
 if (ptr_appartenance_sur_id1(theme,&art_ptr_appartenance)==F_OK)
    {
     posit_acc=art_ptr_appartenance.ptr_app;
     lire_fichier(fdapp,disk_name,(char *)&art_appartenance,
                  sizeof(tappartenanceint),&posit_acc);
     plat_sur_id1(art_appartenance.no_plat,art_plat);
     return(F_OK);
    }
    else return(F_FIN_FICH);
}

/***************************************/
short suivant_plat_sur_cle1(theme,art_plat)
/***************************************/

short theme;
tplat *art_plat;

{
 tappartenanceint art_appartenance;
 short posit_acc;

 if (i<art_ptr_appartenance.nbr_plats)
    {
     posit_acc=POS_COURANT;
     lire_fichier(fdapp,disk_name,(char *)&art_appartenance,
                  sizeof(tappartenanceint),&posit_acc);
```

```
        plat_sur_id1(art_appartenance.no_plat,art_plat);
        ++i;
        return(F_OK);
       }
       else return(F_FIN_FICH);
}
```

```
/*
 * Module   : Operations externes (suite)
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */


#include "H_PrimBD.h"
#include "H_types2.h"
/****************************/
/*  ACCES EXTERNES (SUITE)  */
/****************************/
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
            *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
            *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit;
PUBLIC char *disk_name;

static short i;
static tptr_liens_plats art_ptr_liensplats;

/************************************************/
short premier_plat_sur_cle2(plat_pere,art_platfils)
/************************************************/

short plat_pere;
tplat *art_platfils;

{
 tliens_plats art_liensplats;
 short posit_acc;

 i=1;
 art_ptr_liensplats.no_plat=plat_pere;
 if (ptr_liens_plats_sur_id1(plat_pere,&art_ptr_liensplats)==F_OK)
    {
     posit_acc=art_ptr_liensplats.ptr_liensplats;
     lire_fichier(fdliensplats,disk_name,(char *)&art_liensplats,
                  sizeof(tliens_plats),&posit_acc);
     if (art_liensplats.no_plat_fils!=infini)
        plat_sur_id1(art_liensplats.no_plat_fils,art_platfils);
        else {
              art_platfils->no_plat=infini;
              strcpy(art_platfils->nom,'\0');
              strcpy(art_platfils->son_nom,'\0');
             }
     return(F_OK);
    }
    else return(F_FIN_FICH);
}
```

```
/************************************************/
short suivant_plat_sur_cle2(plat_pere,art_platfils)
/************************************************/

short plat_pere;
tplat *art_platfils;

{
 tliens_plats art_liensplats;
 short posit_acc;

 if (i<art_ptr_liensplats.nbr_fils)
    {
     posit_acc=POS_COURANT;
     lire_fichier(fdliensplats,disk_name,(char *)&art_liensplats,
                  sizeof(tliens_plats),&posit_acc);
     if (art_liensplats.no_plat_fils!=infini)
         plat_sur_id1(art_liensplats.no_plat_fils,art_platfils);
         else {
               art_platfils->no_plat=infini;
               strcpy(art_platfils->nom,'\0');
               strcpy(art_platfils->son_nom,'\0');
              }
     ++i;
     return(F_OK);
    }
    else return(F_FIN_FICH);
}
```

```
/*
 * Fichier  : Operationsint
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *          : Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
           *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
           *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit;
PUBLIC char *disk_name;
/*************************/
/*  OPERATIONS INTERNES  */
/*************************/

/********************/
/*  ACCES INTERNES  */
/********************/

/*********************************************************/
short ptr_menu_const_sur_id1(date,repas,art_ptr_menu_const)
/*********************************************************/

tdate date;
short repas;
tptr_menu_const *art_ptr_menu_const;

{
 short posit_acc;

 art_ptr_menu_const->date=date;
 art_ptr_menu_const->repas=repas;
 return(lire_sur_id(fdptrmenuconst,disk_name,(char *)art_ptr_menu_const,
                  (char *)&art_ptr_menu_const->date,
                   (sizeof(date)+sizeof(repas)),
                  sizeof(*art_ptr_menu_const),&posit_acc));
}

/*************************************************************************/
short constint_posit_sur_id1(date,repas,theme,art_constitution,position)
/*************************************************************************/

tdate date;
short repas,theme,*position;
tconstitutionint *art_constitution;

{
 boolean trouve;
 tptr_menu_const art_ptr_menu_const;
 short statutbd;

 statutbd=ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const);
 if (statutbd==F_OK)
    {
     trouve=False;
     *position=art_ptr_menu_const.ptr_const;
     while ((trouve==False) && (statutbd==F_OK))
```

```
          {
           statutbd=lire_fichier(fdconst,disk_name,
                                 (char *)art_constitution,
                      sizeof(*art_constitution),position);
           if (statutbd==F_OK)
              {
           if (art_constitution->no_theme==theme)
               trouve=True;
               else {
                     if (art_constitution->ptr_suivant==infini)
                         statutbd=F_FIN_FICH;
                    else *position=art_constitution->ptr_suivant;
                    }
           }
         }
    }
 return(statutbd);
}

/*****************************************************************/
short constitution_int_sur_id1(date,repas,theme,art_constitution)
/*****************************************************************/

tdate date;
short repas,theme;
tconstitutionint *art_constitution;

{
 short position;

 return(constint_posit_sur_id1(date,repas,theme,art_constitution,
                               &position));
}

/*****************************************************/
short ptr_liens_plats_sur_id1(plat,art_ptr_liensplats)
/*****************************************************/

short plat;
tptr_liens_plats *art_ptr_liensplats;

{
 short posit_acc;

 art_ptr_liensplats->no_plat=plat;
 return(lire_sur_id(fdptrliensplats,disk_name,(char *)art_ptr_liensplats,
                  (char *)&art_ptr_liensplats->no_plat,sizeof(plat),
                  sizeof(*art_ptr_liensplats),&posit_acc));
}

/*********************************************************/
short ptr_appartenance_sur_id1(theme,art_ptr_appartenance)
/*********************************************************/

short theme;
tptr_appartenance *art_ptr_appartenance;

{
 short posit_acc;
```

```
 art_ptr_appartenance->no_theme=theme;
 return(lire_sur_id(fdptrapp,disk_name,(char *)art_ptr_appartenance,
                    (char *)&art_ptr_appartenance->no_theme,sizeof(theme),
                     sizeof(*art_ptr_appartenance),&posit_acc));
}

/*****************************************/
short theme_int_sur_id1(no_theme,art_theme)
/*****************************************/

short no_theme;
tthemeint *art_theme;

{
 short posit_acc;

 art_theme->no_theme=no_theme;
 return(lire_sur_id(fdtheme,disk_name,(char *)art_theme,
                       (char *)&art_theme->no_theme,sizeof(short),
                       sizeof(*art_theme),&posit_acc));
}

/*************************************/
short plat_int_sur_id1(no_plat,art_plat)
/*************************************/

short no_plat;
tplatint *art_plat;

{
 short posit_acc;

 art_plat->no_plat=no_plat;
 return(lire_sur_id(fdplat,disk_name,(char *)art_plat,
                    (char *)&art_plat->no_plat,
                    sizeof(short),sizeof(*art_plat),&posit_acc));
}

/*****************************************/
short menu_int_sur_id1(date,repas,art_menu)
/*****************************************/

tdate date;
short repas;
tmenuint *art_menu;

{
 tptr_menu_const art_ptr_menu_const;
 short position;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     position=art_ptr_menu_const.ptr_menu;
     return(lire_fichier(fdmenu,disk_name,(char *)art_menu,
                         sizeof(*art_menu),&position));
    }
    else return(F_MAUV_CLE);
}
```

```
/*****************************************************/
short ptr_compos_plat_sur_id1(plat,art_ptr_compos_plat)
/*****************************************************/

short plat;
tptr_compos_plat *art_ptr_compos_plat;

{
 short posit_acc;

 art_ptr_compos_plat->no_plat=plat;
 return(lire_sur_id(fdptrcomposplat,disk_name,(char *)art_ptr_compos_plat,
                    (char *)&art_ptr_compos_plat->no_plat,sizeof(short),
                     sizeof(tptr_compos_plat),&posit_acc));
}

/*****************************************/
short produit_int_sur_id1(numero,art_produit)
/*****************************************/

short numero;
tproduitint *art_produit;

{
 short posit_acc;

 art_produit->no_produit=numero;
 return(lire_sur_id(fdproduit,disk_name,(char *)art_produit,
                    (char *)&art_produit->no_produit,sizeof(short),
                     sizeof(tproduitint),&posit_acc));
}

/************************/
/*  CREATIONS INTERNES  */
/************************/

/*******************************************/
void creer_ptr_menu_const(date,repas,position)
/*******************************************/

tdate date;
short repas,position;

{
 short statutbd,posit_acc;

 boolean trouve;
 FILE *fdtemp;
 tptr_menu_const art_ptr_menu_const,art_temp;

 fdtemp=ouvrir_fichier("PTR_TEMP",W,disk_name);
 statutbd=F_OK;
 trouve=False;
 posit_acc=POS_DEBUT;

 while ((statutbd==F_OK) && (trouve==False))
       {
        statutbd=lire_fichier(fdptrmenuconst,disk_name,
```

```
                        (char *)&art_ptr_menu_const,
                            sizeof(tptr_menu_const),&posit_acc);
       if (statutbd==F_OK)
          {
           if ((compare_date(art_ptr_menu_const.date,date,'>')==True) ||

((compare_date(art_ptr_menu_const.date,date,'=')==True) &&
              (art_ptr_menu_const.repas>repas)))
              {
                trouve=True;
                art_temp.date=date;
                art_temp.repas=repas;
                art_temp.ptr_menu=position;
                art_temp.ptr_const=infini;
                 posit_acc=POS_FIN;
                ecrire_fichier(fdtemp,disk_name,(char *)&art_temp,
                              sizeof(art_temp),&posit_acc);
              }
             posit_acc=POS_FIN;
             ecrire_fichier(fdtemp,disk_name,
                              (char *)&art_ptr_menu_const,
                        sizeof(tptr_menu_const),&posit_acc);
             posit_acc=POS_COURANT;
          }
       }
 if (statutbd==F_FIN_FICH)
    {art_temp.date=date;
     art_temp.repas=repas;
     art_temp.ptr_menu=position;
     art_temp.ptr_const=infini;
     posit_acc=POS_FIN;
     ecrire_fichier(fdtemp,disk_name,(char *)&art_temp,
                  sizeof(tptr_menu_const),&posit_acc);
    }
    else
    {while (statutbd==F_OK)
      {
       posit_acc=POS_COURANT;
       statutbd=lire_fichier(fdptrmenuconst,disk_name,
                        (char *)&art_ptr_menu_const,
                            sizeof(tptr_menu_const),&posit_acc);
       if (statutbd==F_OK)
          {
           posit_acc=POS_FIN;
           ecrire_fichier(fdtemp,disk_name,(char *)&art_ptr_menu_const,
                     sizeof(tptr_menu_const),&posit_acc);
          }
      }
    }
 fermer_fichier(fdtemp,disk_name);
 fermer_fichier(fdptrmenuconst,disk_name);
 supprimer_fichier("PTR_MENU_CONST",disk_name);
 renommer_fichier("PTR_TEMP","PTR_MENU_CONST",disk_name);
 fdptrmenuconst=ouvrir_fichier("PTR_MENU_CONST",RW,disk_name);
 return;
}
```

```
/****************************************/
void creer_preparation_int(art_preparation)
/****************************************/

tpreparationint *art_preparation;

{
 return;
}

/*************************************/
void creer_menu_int(date,repas,art_menu)
/*************************************/

tdate date;
short repas;
tmenuint *art_menu;

{
 short posit_acc;

 posit_acc=POS_FIN;
 ecrire_fichier(fdmenu,disk_name,(char *)art_menu,sizeof(*art_menu),
                &posit_acc);
 creer_ptr_menu_const(date,repas,posit_acc);
 return;
}

/*****************************************************/
void creer_constitution_int(date,repas,art_constitution)
/*****************************************************/

tdate date;
short repas;
tconstitutionint *art_constitution;

{
 short posit_acc;

 posit_acc=POS_FIN;
 ecrire_fichier(fdconst,disk_name,(char *)art_constitution,
                sizeof(*art_constitution),&posit_acc);
 insertion_liste_constitutions(date,repas,art_constitution->no_theme,
                               posit_acc);
 return;
}
```

```
/*************************/
/*  EFFACEMENTS INTERNES  */
/*************************/

/*********************************************************************/
short effacer_ptr_menu_const_majptrmenu_sur_id1(date,repas,position)
/*********************************************************************/

tdate date;
short repas,position;

{
 short statutbd,posit_acc;
 FILE *fdtemp;
 tptr_menu_const art_ptr_menu_const;
 boolean trouve;

 fdtemp=ouvrir_fichier("PTR_MENU_CONST_TEMP",W,disk_name);
 statutbd=F_OK;
 trouve=False;
 posit_acc=POS_DEBUT;

 while ((statutbd==F_OK) && (trouve==False))
       {
        statutbd=lire_fichier(fdptrmenuconst,disk_name,
                              (char *)&art_ptr_menu_const,
                                  sizeof(tptr_menu_const),&posit_acc);
        if (statutbd==F_OK)
           {
            if ((compare_date(art_ptr_menu_const.date,date,'=')==True) &&
                (art_ptr_menu_const.repas==repas))
                trouve=True;
                else {
                     if (art_ptr_menu_const.ptr_menu>position)
                         --art_ptr_menu_const.ptr_menu;
                      posit_acc=POS_COURANT;
                      ecrire_fichier(fdtemp,disk_name,
                                  (char *)&art_ptr_menu_const,
                                   sizeof(tptr_menu_const),&posit_acc);
                     }
           }
           posit_acc=POS_COURANT;
       }

 while (statutbd==F_OK)
       {
        posit_acc=POS_COURANT;
        statutbd=lire_fichier(fdptrmenuconst,disk_name,
                              (char *)&art_ptr_menu_const,
                               sizeof(tptr_menu_const),&posit_acc);
        if (statutbd==F_OK)
           {
            if (art_ptr_menu_const.ptr_menu>position)
                --art_ptr_menu_const.ptr_menu;
                posit_acc=POS_COURANT;
                ecrire_fichier(fdtemp,disk_name,(char *)&art_ptr_menu_const,
                              sizeof(tptr_menu_const),&posit_acc);
           }
       }
```

```
 fermer_fichier(fdtemp,disk_name);
 fermer_fichier(fdptrmenuconst,disk_name);
 supprimer_fichier("PTR_MENU_CONST",disk_name);
 renommer_fichier("PTR_MENU_CONST_TEMP","PTR_MENU_CONST",disk_name);
 fdptrmenuconst=ouvrir_fichier("PTR_MENU_CONST",RW,disk_name);
 if (trouve==True) return (F_OK);
                   else return (F_MAUV_CLE);
}

/*********************************************************************/
short effacer_const_int_majptrsuivant_sur_deplptr(depl_ptr,longueur)
/*********************************************************************/

short depl_ptr[],longueur;

{
 short statutbd,position,posit_acc,nouv_ptr;
 boolean dernier,reponse;
 FILE *fdtemp;
 tconstitutionint art_constitution;

 position=0;
 fdtemp=ouvrir_fichier("CONSTITUTION_TEMP",W,disk_name);
 dernier=False;
 statutbd=F_OK;
 posit_acc=POS_DEBUT;

 while ((statutbd==F_OK) && (dernier==False))
       {
        statutbd=lire_fichier(fdconst,disk_name,(char *)&art_constitution,
                              sizeof(tconstitutionint),&posit_acc);
        if (statutbd==F_OK)
           {
            retrait_art(position,depl_ptr,longueur,&reponse);
            if (reponse==False)
               {
                nouveau_ptr(art_constitution.ptr_suivant,&nouv_ptr,
                            depl_ptr,longueur);
                 art_constitution.ptr_suivant=nouv_ptr;
                 posit_acc=POS_COURANT;
                ecrire_fichier(fdtemp,disk_name,(char *)&art_constitution,
                               sizeof(tconstitutionint),&posit_acc);
               }
               else
               {
                 if (art_constitution.ptr_suivant==infini)
                dernier=True;
               }
               ++position;
           }
          posit_acc=POS_COURANT;
       }
 while (statutbd==F_OK)
       {
         posit_acc=POS_COURANT;
         statutbd=lire_fichier(fdconst,disk_name,(char *)&art_constitution,
                               sizeof(tconstitutionint),&posit_acc);
         if (statutbd==F_OK)
            {
```

```
            nouveau_ptr(art_constitution.ptr_suivant,&nouv_ptr,
                        depl_ptr,longueur);
            art_constitution.ptr_suivant=nouv_ptr;
            posit_acc=POS_COURANT;
            ecrire_fichier(fdtemp,disk_name,(char *)&art_constitution,
                        sizeof(tconstitutionint),&posit_acc);
           }
       }
 fermer_fichier(fdtemp,disk_name);
 fermer_fichier(fdconst,disk_name);
 supprimer_fichier("CONSTITUTION",disk_name);
 renommer_fichier("CONSTITUTION_TEMP","CONSTITUTION",disk_name);
 fdconst=ouvrir_fichier("CONSTITUTION",RW,disk_name);
 if (longueur>0) return (F_OK);
               else return (F_MAUV_CLE);
}

/*********************************************/
short effacer_preparation_int_sur_id1(date,repas)
/*********************************************/

tdate date;
short repas;

{
 tptr_menu_const art_ptr_menu_const;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    return(F_OK);
    else return(F_MAUV_CLE);
}

/*****************************************/
short effacer_menu_int_sur_id1(date,repas)
/*****************************************/

tdate date;
short repas;

{
 tptr_menu_const art_ptr_menu_const;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     effacer_fichier(fdmenu,disk_name,"MENU",RW,sizeof(tmenuint),
                    &art_ptr_menu_const.ptr_menu);
     effacer_ptr_menu_const_majptrmenu_sur_id1(date,repas,
                                          art_ptr_menu_const.ptr_menu);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/*********************************************/
short effacer_constitution_int_sur_cle1(date,repas)
/*********************************************/

tdate date;
short repas;

{
 short depl_ptr[maxthememenu+1],longueur;
 tptr_menu_const art_ptr_menu_const;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     calculer_depl_ptr_const(date,repas,depl_ptr,&longueur);
     if (effacer_const_int_majptrsuivant_sur_deplptr(depl_ptr,longueur)
                                                    ==F_OK)
        majptrmenuconstptrconstsurdeplptr(depl_ptr,longueur);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}

/***************************/
/*  MISES-A-JOUR INTERNES  */
/***************************/

/*************************************************************************/
short maj_menu_int_pers_modif_sur_id1(date,repas,nbr_pers,modifie)
/*************************************************************************/

tdate date;
short repas,nbr_pers;
boolean modifie;

{
 tptr_menu_const art_ptr_menu_const;
 tmenu art_menu;
 tmenuint art_temp;
 short posit_acc;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     posit_acc=art_ptr_menu_const.ptr_menu;
     lire_fichier(fdmenu,disk_name,(char *)&art_temp,sizeof(art_temp),
                  &posit_acc);
     convintprgmenu(&art_temp,&art_menu);
     art_menu.modifie=modifie;
     art_menu.nnbr_pers=nbr_pers;
     convprgintmenu(&art_menu,&art_temp);
     ecrire_fichier(fdmenu,disk_name,(char *)&art_temp,sizeof(art_temp),
                    &posit_acc);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/*************************************************/
short maj_menu_int_imprime_sur_id1(date,repas,imprime)
/*************************************************/

tdate date;
short repas;
boolean imprime;

{
 tptr_menu_const art_ptr_menu_const;
 tmenu art_menu;
 tmenuint art_temp;
 short posit_acc;

 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     posit_acc=art_ptr_menu_const.ptr_menu;
     lire_fichier(fdmenu,disk_name,(char *)&art_temp,sizeof(art_temp),
                  &posit_acc);
     convintprgmenu(&art_temp,&art_menu);
     art_menu.imprime=imprime;
     convprgintmenu(&art_menu,&art_temp);
     ecrire_fichier(fdmenu,disk_name,(char *)&art_temp,sizeof(art_temp),
                    &posit_acc);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}

/*************************************************************************/
short maj_const_int_modifieplat_sur_id1(date,repas,theme,modifie_plat)
/*************************************************************************/

tdate date;
short repas,theme;
char modifie_plat;

{
 tconstitution art_constitution;
 tconstitutionint art_temp;
 short position;

 if (constint_posit_sur_id1(date,repas,theme,&art_temp,&position)==F_OK)
    {
     convintprgconstitution(&art_temp,&art_constitution);
     art_constitution.modifie_plat=modifie_plat;
     convprgintconstitution(&art_constitution,art_temp.no_theme,
                            art_temp.no_plat,art_temp.ptr_suivant,
                  &art_temp);
     ecrire_fichier(fdconst,disk_name,(char *)&art_temp,
                    sizeof(tconstitutionint),&position);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/*************************************************************************/
short maj_const_int_plat_sur_id1(plat,date,repas,theme,modif_plat)
/*************************************************************************/

short plat,repas,theme;
tdate date;
char modif_plat;

{
 tconstitutionint art_temp;
 tconstitution art_constitution;
 short position;

 if (constint_posit_sur_id1(date,repas,theme,&art_temp,&position)==F_OK)
    {
     convintprgconstitution(&art_temp,&art_constitution);
     art_constitution.modifie_plat=modif_plat;
     convprgintconstitution(&art_constitution,art_temp.no_theme,plat,
                            art_temp.ptr_suivant,&art_temp);
     ecrire_fichier(fdconst,disk_name,(char *)&art_temp,sizeof(art_temp),
                    &position);
     return(F_OK);
    }
    else return(F_MAUV_CLE);
}
```

```
/*
 * Module   : Operation sur les dates
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "time.h"
#include "H_PrimBD.h"
#include "H_Types2.h"

PUBLIC tdatecournom jour_courant;
PUBLIC tparamutil param;

/*****************************/
/*  OPERATIONS SUR LES DATES  */
/*****************************/

/*************************/
void calculer_jour_courant()
/*************************/

{
   long t;
   char temps[26];

   time(&t);
   strcpy(temps,ctime(&t));
   calculer_jour_courant_jour(temps);
   calculer_jour_courant_mois(temps);
   calculer_jour_courant_annee(temps);
   calculer_jour_courant_nom(temps);
}

/***********************************/
boolean compare_date(date1,date2,signe)
/***********************************/

tdate date1,date2;
char signe;

{
 boolean rapport;

 rapport=False;
 if (signe=='=')
    {if ((date1.annee==date2.annee) && (date1.mois==date2.mois)
    && (date1.jour==date2.jour))
   rapport=True;
    }
    else {
          if (signe=='<')
             {if (date1.annee<date2.annee)
        rapport=True;
        else {if (date1.annee==date2.annee)
                {if (date1.mois<date2.mois)
                    rapport=True;
                    else {if (date1.mois==date2.mois)
                            {if (date1.jour<date2.jour)
```

```
                        rapport=True;
                    }
                }
            }
        }
    }
        else {
             if (signe=='>')
                {if (date1.annee>date2.annee)
            rapport=True;
            else {if (date1.annee==date2.annee)
                   {if (date1.mois>date2.mois)
                      rapport=True;
                      else {if (date1.mois==date2.mois)
                              {if (date1.jour>date2.jour)
                                 rapport=True;
                              }
                           }
                   }
                    }
        }
    }
    }
 return(rapport);
}

/**********************/
void date_suivante(date)
/**********************/

tdate *date;

{
   short tab_joursmois[12];

   initialiser_tab_joursmois(tab_joursmois);
   ++(date->jour);
   if (date->mois==(short)2)
      {
         if (bissextile(date->annee))
            {
               if (date->jour==tab_joursmois[1]+1)
                  {
                  }
                  else {
                        if (date->jour>tab_joursmois[1]+1)
                           {
                              date->jour=(short)1;
                              ++date->mois;
                           }
                        }
            }
            else
            {
             if (date->jour>tab_joursmois[1])
                {
                 date->jour=(short)1;
                 ++date->mois;
                }
```

```
            }
        }
        else
        {
           if (date->jour>tab_joursmois[date->mois-1])
              {
                 date->jour=(short)1;
                 ++date->mois;
              }
        }
     if (date->mois>(short)12)
        {
           date->mois=(short)1;
           ++date->annee;
        }
}

/************************/
void calculer_calendrier()
/************************/
{
 if(param.scmenus==1)
   calculer_calendrier_sc1();
 else
   calculer_calendrier_sc2();
}
```

```
/*
 * Fichier  : Sousopsurdate
 * Type     : Source complète
 * Auteurs  : Topet Léopold
 *            Vonèche Xavier
 */

#include "H_Privacy.h"
#include "H_Types2.h"
#include <string.h>

PUBLIC short nbrmenupreprep;
PUBLIC tmenupreprep tab_menupreprep[(max_periode_a_traiter*maxrepas)];
PUBLIC tdatecournom jour_courant;
PUBLIC tcalendrier calendrier[max_periode_a_traiter];
PUBLIC short calendrier_long;
PUBLIC short nbrrepas;
PUBLIC tjour tab_nomjours[7];
PUBLIC tmois tab_nommois[13];

/*******************************************/
void initialiser_tab_joursmois (tab_joursmois)
/*******************************************/

short tab_joursmois[];

{
   tab_joursmois[0]=(short)31;
   tab_joursmois[1]=(short)28;
   tab_joursmois[2]=(short)31;
   tab_joursmois[3]=(short)30;
   tab_joursmois[4]=(short)31;
   tab_joursmois[5]=(short)30;
   tab_joursmois[6]=(short)31;
   tab_joursmois[7]=(short)31;
   tab_joursmois[8]=(short)30;
   tab_joursmois[9]=(short)31;
   tab_joursmois[10]=(short)30;
   tab_joursmois[11]=(short)31;
}

/************************/
boolean bissextile(annee)
/************************/

short annee;

{
   if ((annee % (short)400==0) || (annee % (short)4==0))
      return(True);
      else return(False);
}
```

```
/************************************/
void calculer_jour_courant_jour(temps)
/************************************/

char temps[];

{
   char inter[3];

   inter[0]=temps[8];
   inter[1]=temps[9];
   inter[2]='\0';
   jour_courant.jour=(short)(atoi(inter));
}

/************************************/
void calculer_jour_courant_mois(temps)
/************************************/

char temps[];

{

 if (temps[4]=='J')
    {
     if (temps[5]=='a')
          jour_courant.mois=1;
        else {
              if (temps[6]=='n')
                   jour_courant.mois=6;
                 else jour_courant.mois=7;
             }
  }
  else {
       if (temps[4]=='F')
           jour_courant.mois=2;
         else {
               if (temps[4]=='M')
                  {
                   if (temps[5]=='a')
                         jour_courant.mois=3;
                     else jour_courant.mois=5;
                   }
                   else {
                        if (temps[4]=='A')
                           {
                            if (temps[5]=='p')
                                 jour_courant.mois=4;
                                 else jour_courant.mois=8;
                            }
                           else {
                                 if (temps[4]=='S')
                                     jour_courant.mois=9;
                                   else {
                                         if (temps[4]=='O')
                                             jour_courant.mois=10;
                                           else {
                                                 if (temps[4]=='N')
                                                  jour_courant.mois=11;
```

```
                                                    else
                                                    jour_courant.mois=12;
                                                   }
                                            }
                                     }
                              }
                       }
                }
}

/************************************/
void calculer_jour_courant_annee(temps)
/************************************/

char temps[];

{
   char inter[5];

   inter[0]=temps[20];
   inter[1]=temps[21];
   inter[2]=temps[22];
   inter[3]=temps[23];
   inter[4]='\0';

   jour_courant.annee=(short)(atoi(inter));
}

/*********************************/
void calculer_jour_courant_nom(temps)
/*********************************/

char temps[26];
{
   if (temps[0]=='M')
      strcpy(jour_courant.nom,"Lundi");
      else {
           if (temps[0]=='T')
              {
               if (temps[1]=='u')
                  strcpy(jour_courant.nom,"Mardi");
                  else strcpy(jour_courant.nom,"Jeudi");
              }
              else {
                   if (temps[0]=='W')
                      strcpy(jour_courant.nom,"Mercredi");
                      else {
                           if (temps[0]=='F')
                              strcpy(jour_courant.nom,"Vendredi");
                              else {
                                   if (temps[1]=='a')
                                      strcpy(jour_courant.nom,"Samedi");
                                      else strcpy(jour_courant.nom,
                                                  "Dimanche");
                                   }
                           }
                   }
           }
}
```

```
/*******************************/
void copier_article(s1,s2,taille)
/*******************************/

char *s1,*s2;
int taille;

{
   short i;

   for (i=0;i<taille;++i)
       s1[i]=s2[i];
}

/*********************************/
boolean verifier_jour_traite (date)
/*********************************/

tdate date;

{
   short i;
   boolean traite;

   i=0;
   traite=False;
   while ((i<nbrmenupreprep) && (traite==False))
         {
            if (compare_date(date,tab_menupreprep[i].date,'=')==True)
               traite=True;
               else ++i;
         }
   return(traite);
}

/*********************************/
boolean verifier_jour_imprime (date)
/*********************************/

tdate date;

{
   short i;
   boolean trouve;

   i=0;
   trouve=False;
   while ((i<nbrmenupreprep) && (trouve=False))
         {
            if (compare_date(date,tab_menupreprep[i].date,'=')==True)
               trouve=True;
               else ++i;
         }
   if (trouve==True)
      return(tab_menupreprep[i].imprime);
      else return(False);
}
```

```
/************************************/
void insertion_calendrier(date,position)
/************************************/

tdate date;
short position;

{
   copier_article((char *)&calendrier[position],(char *)&date,
                  sizeof(date));
}

/******************************/
void initialiser_tab_nomjours ()
/******************************/

{
   strcpy(tab_nomjours[0].nom,"Lundi");
   strcpy(tab_nomjours[0].son,"LAX9NDIY");
   strcpy(tab_nomjours[1].nom,"Mardi");
   strcpy(tab_nomjours[1].son,"MAWRDIY");
   strcpy(tab_nomjours[2].nom,"Mercredi");
   strcpy(tab_nomjours[2].son,"MEYRKRAXDIY");
   strcpy(tab_nomjours[3].nom,"Jeudi");
   strcpy(tab_nomjours[3].son,"ZHAH9DIY");
   strcpy(tab_nomjours[4].nom,"Vendredi");
   strcpy(tab_nomjours[4].son,"VAX9NDRAHDIY");
   strcpy(tab_nomjours[5].nom,"Samedi");
   strcpy(tab_nomjours[5].son,"SAEMDIY");
   strcpy(tab_nomjours[6].nom,"Dimanche");
   strcpy(tab_nomjours[6].son,"DIY9MAX9NSH");
}

/*****************************/
void initialiser_tab_nommois ()
/*****************************/

{
   strcpy(tab_nommois[1].nom,"Janvier");
   strcpy(tab_nommois[1].son,"JAX9NVIYEY");
   strcpy(tab_nommois[2].nom,"Février");
   strcpy(tab_nommois[2].son,"FEY9VRIEY");
   strcpy(tab_nommois[3].nom,"Mars");
   strcpy(tab_nommois[3].son,"MAARS");
   strcpy(tab_nommois[4].nom,"Avril");
   strcpy(tab_nommois[4].son,"AAVRIHL");
   strcpy(tab_nommois[5].nom,"Mai");
   strcpy(tab_nommois[5].son,"MEY");
   strcpy(tab_nommois[6].nom,"Juin");
   strcpy(tab_nommois[6].son,"ZHUWAXIH5N");
   strcpy(tab_nommois[7].nom,"Juillet");
   strcpy(tab_nommois[7].son,"ZHUW9IY9EH9");
   strcpy(tab_nommois[8].nom,"Août");
   strcpy(tab_nommois[8].son,"UW");
   strcpy(tab_nommois[9].nom,"Septembre");
   strcpy(tab_nommois[9].son,"SEHTAX9MBRAX");
   strcpy(tab_nommois[10].nom,"Octobre");
   strcpy(tab_nommois[10].son,"OHKTOHBDXRAX");
   strcpy(tab_nommois[11].nom,"Novembre");
```

```
   strcpy(tab_nommois[11].son,"NOH9VAX9MBRAX");
   strcpy(tab_nommois[12].nom,"Décembre");
   strcpy(tab_nommois[12].son,"DEYSAX9MBRAX9");
}

/*****************************************/
void position_jour(tab_nomjours,nom,position)
/*****************************************/

tjour tab_nomjours[];
char *nom;
short *position;

{
   boolean trouve;

   *position=0;
   trouve=False;
   while (trouve==False)
         {
            if (strcmp(tab_nomjours[*position].nom,nom)==0)
                  trouve=True;
                  else ++*position;
         }
}

/*****************************/
void calculer_calendrier_sc1()
/*****************************/

{
   tdate date_temp;
   short jour_a_traiter,position;
   boolean traite;

   calendrier_long=0;
   date_temp.annee=jour_courant.annee;
   date_temp.mois=jour_courant.mois;
   date_temp.jour=jour_courant.jour;
   position_jour(tab_nomjours,jour_courant.nom,&position);
   traite=verifier_jour_traite(date_temp);
   imprime=verifier_jour_imprime(date_temp);
   while ((calendrier_long<param.eloignement_max) && (traite==True))
         {
            insertion_calendrier(date_temp,calendrier_long);
            calendrier[calendrier_long].nom=((position+calendrier_long)%7);
            calendrier[calendrier_long].traite=traite;
            calendrier[calendrier_long].imprime=imprime;
            date_suivante(&date_temp);
            traite=verifier_jour_traite(date_temp);
            imprime=verifier_jour_imprime(date_temp);
            ++calendrier_long;
         }
   jour_a_traiter=1;
   while ((jour_a_traiter<=param.periode_max_a_traiter) &&
          (calendrier_long<param.eloignement_max))
          {
            insertion_calendrier(date_temp,calendrier_long);
            calendrier[calendrier_long].nom=((position+calendrier_long)%7);
```

```
            calendrier[calendrier_long].traite=False;
            calendrier[calendrier_long].imprime=verifier_jour_imprime
                                                (date_temp);
            date_suivante(&date_temp);
            ++jour_a_traiter;
            ++calendrier_long;
          }
}

/*****************************/
void calculer_calendrier_sc2()
/*****************************/

{
   tdate date_temp;
   short position;
   boolean traite;

   calendrier_long=0;
   date_temp.annee=jour_courant.annee;
   date_temp.mois=jour_courant.mois;
   date_temp.jour=jour_courant.jour;
   position_jour(tab_nomjours,jour_courant.nom,&position);
   traite=verifier_jour_traite(date_temp);
   imprime=verifier_jour_imprime(date_temp);
   do
         {
            insertion_calendrier(date_temp,calendrier_long);
            calendrier[calendrier_long].nom=((position+calendrier_long)%7);
            calendrier[calendrier_long].traite=traite;
            calendrier[calendrier_long].imprime=imprime;
            date_suivante(&date_temp);
            traite=verifier_jour_traite(date_temp);
            imprime=verifier_jour_imprime(date_temp);
            ++calendrier_long;
         }
   while (calendrier_long<param.eloignement_max);
}
```

```
/*
 * ---> Fichier: Constitution
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*************************************************/
/*  GESTION DE PTR_MENU_CONST, MENU, CONSTITUTION     */
/*************************************************/
PUBLIC FILE *fdtheme, *fdplat, *fdptrliensplats, *fdliensplats,
           *fdappartenance, *fdptrappartenance, *fdptrmenuconst, *fdconst,
           *fdptrapp, *fdmenu, *fdptrcomposplat, *fdproduit;
PUBLIC char *disk_name;
/*************************************************/
void retrait_art(position,depl_ptr,longueur,reponse)
/*************************************************/

short position,longueur;
short  depl_ptr[];
boolean *reponse;

{
 short i;

 i=0;
 *reponse=False;
 while ((i<longueur) && (*reponse==False))
       {if (position==depl_ptr[i])
          *reponse=True;
          else ++i;
       }
 return;
}

/*************************************************/
void nouveau_ptr(ptr_actuel,nouv_ptr,depl_ptr,longueur)
/*************************************************/

short ptr_actuel,*nouv_ptr,longueur;
short depl_ptr[];

{
 short i;
 boolean superieur;

 i=0;
 superieur=False;
 if ((ptr_actuel<depl_ptr[i]) || (ptr_actuel==infini))
    *nouv_ptr=ptr_actuel;
    else {while (superieur==False)
              {if ((ptr_actuel>depl_ptr[i]) && (ptr_actuel<depl_ptr[i+1]))
                       superieur=True;
                       else ++i;
              }
  *nouv_ptr=(ptr_actuel - (i+1));
}
```

```
 return;
}

/*******************************************************/
void majptrmenuconstptrconstsurdeplptr(depl_ptr,longueur)
/*******************************************************/

short depl_ptr[];
short longueur;

{
 short statutbd,nouv_ptr,posit_acc;
 FILE *fdtemp;
 tptr_menu_const art_ptrmenuconst;

 fdtemp=ouvrir_fichier("PTR_MENU_CONST_TEMP",W,disk_name);
 statutbd=F_OK;
 posit_acc=POS_DEBUT;

 while (statutbd==F_OK)
       {
       statutbd=lire_fichier(fdptrmenuconst,disk_name,(char *)&art_ptrmenuconst,
                             sizeof(tptr_menu_const),&posit_acc);
        if (statutbd==F_OK)
          {nouveau_ptr(art_ptrmenuconst.ptr_const,&nouv_ptr,depl_ptr,
                       longueur);
           art_ptrmenuconst.ptr_const=nouv_ptr;
           posit_acc=POS_COURANT;
           statutbd=ecrire_fichier(fdtemp,disk_name,(char *)&art_ptrmenuconst,
                                   sizeof(tptr_menu_const),&posit_acc);
           }
       posit_acc=POS_COURANT;
       }
 fermer_fichier(fdtemp,disk_name);
 fermer_fichier(fdptrmenuconst,disk_name);
 supprimer_fichier("PTR_MENU_CONST",disk_name);
 renommer_fichier("PTR_MENU_CONST_TEMP","PTR_MENU_CONST",disk_name);
 fdptrmenuconst=ouvrir_fichier("PTR_MENU_CONST",RW,disk_name);
 return;
}

/*************************************************************/
short insertion_liste_constitutions(date,repas,theme,position)
/*************************************************************/

tdate date;
short repas,theme,position;

{
 short statutbd,positionart;
 tptr_menu_const art_ptr_menu_const;
 tconstitutionint art_constitution;
 boolean trouve;

 statutbd=ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const);
 if (statutbd==F_OK)
    {
     if (art_ptr_menu_const.ptr_const==infini)
        {
```

```
        art_ptr_menu_const.ptr_const=position;
        ecrire_sur_id(fdptrmenuconst,disk_name,(char *)&art_ptr_menu_const,
                    (char *)&art_ptr_menu_const.date,
                     (sizeof(tdate)+sizeof(short)),
                   sizeof(tptr_menu_const),&positionart);
       }
       else
       {
        positionart=art_ptr_menu_const.ptr_const;
        trouve=False;
        while (trouve==False)
              {
               lire_fichier(fdconst,disk_name,(char *)&art_constitution,
                           sizeof(tconstitutionint),&positionart);
               if (art_constitution.ptr_suivant==infini)
                  trouve=True;
                  else positionart=art_constitution.ptr_suivant;
              }
        art_constitution.ptr_suivant=position;
        ecrire_fichier(fdconst,disk_name,(char *)&art_constitution,
                    sizeof(art_constitution),&positionart);
       }
    }
 return(statutbd);
}

/*******************************************************/
void calculer_depl_ptr_const(date,repas,depl_ptr,longueur)
/*******************************************************/

tdate date;
short repas,*longueur;
short depl_ptr[];

{
 short statutbd,position;
 boolean dernier;
 tptr_menu_const art_ptr_menu_const;
 tconstitutionint art_constitution;

 *longueur=0;
 if (ptr_menu_const_sur_id1(date,repas,&art_ptr_menu_const)==F_OK)
    {
     position=art_ptr_menu_const.ptr_const;
     depl_ptr[*longueur]=art_ptr_menu_const.ptr_const;
     dernier=False;
     statutbd=F_OK;

     while ((statutbd==F_OK) && (dernier==False))
           {
            statutbd=lire_fichier(fdconst,disk_name,(char *)&art_constitution,
                                sizeof(tconstitutionint),&position);
            if (statutbd==F_OK)
                {if (art_constitution.ptr_suivant==infini)
                    dernier=True;
                    ++ *longueur;
                   depl_ptr[*longueur]=art_constitution.ptr_suivant;
                   position=art_constitution.ptr_suivant;
               }
```

```
        }
    ++ *longueur;
   }
 return;
}
```

```
/*
 * ---> Module  : Gestionnaire des Images
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "stdio.h"
#include "exec/types.h"
#include "H_Ilbm.h"
#include "intuition/intuition.h"
#include "exec/memory.h"
#include "exec/devices.h"
#include "graphics/gfx.h"
#include "graphics/gfxbase.h"
#include "graphics/gfxmacros.h"
#include "graphics/text.h"
#include "graphics/view.h"
#include "graphics/clip.h"
#include "graphics/copper.h"
#include "graphics/gels.h"
#include "graphics/regions.h"
#include "hardware/blit.h"
#include "devices/keymap.h"
#include "libraries/diskfont.h"
#include "error.h"
#include "H_Privacy.h"
#include "H_Types.h"
#define INTUITION_REV 0
#define GRAPHICS_REV  0

struct IntuitionBase *IntuitionBase=NULL;
struct GfxBase *GfxBase=NULL;
struct TextFont *TextFont, *oldFont;
struct TextAttr TextAttr;
struct NewWindow NewWindow;
struct Window *Window;
ULONG *DiskfontBase=NULL;
USHORT LENCMAP;
PRIVATE struct Image fond;
PUBLIC struct Screen *Screen;

boolean fgetst(string,lng,fd)
char *string;
int lng;
FILE *fd;
{
 int i;
 for(i=0;i<lng;i++) string[i]=0;
 return fread(string,lng,1,fd);
}

void FermerLib(void)
{
 if(GfxBase!=NULL) CloseLibrary(GfxBase);
 if(IntuitionBase!=NULL) CloseLibrary(IntuitionBase);
 if(DiskfontBase!=NULL) CloseLibrary(DiskfontBase);
}
```

```
void OuvrirLib(void)
{
 IntuitionBase=(struct IntuitionBase *)
  OpenLibrary("intuition.library",INTUITION_REV);
 if(IntuitionBase==NULL)
  exit(False);
 GfxBase=(struct GfxBase *)
  OpenLibrary("graphics.library",GRAPHICS_REV);
 if(GfxBase==NULL)
  {
   FermerLib();
   exit(False);
  }
 if((DiskfontBase=(ULONG *) OpenLibrary("diskfont.library",0))==NULL)
  {
   FermerLib();
   exit(False);
  }
 TextAttr.ta_Name="Souvenir.font";
 TextAttr.ta_YSize=12;
 TextAttr.ta_Style=FS_NORMAL;
 TextAttr.ta_Flags=FPF_DISKFONT|FPF_DESIGNED;
 TextFont= (struct TextFont *) OpenDiskFont(&TextAttr);
 if(TextFont==0)
 {
  FermerLib();
  exit(False);
 }
}

PRIVATE void Decompress(fd, bitmap, hauteur, largeur, profondeur)
FILE *fd;
UBYTE *bitmap, profondeur;
USHORT hauteur, largeur;
{
 UBYTE buffer,code;
 int blargeur=((largeur+15)/16)*2, bplan=hauteur*blargeur;
 int count1, count2, count3, posit, compteur;
 for(count1=0;count1<hauteur;count1++)
  for(count2=0;count2<profondeur;count2++)
   {
    posit=count2*bplan+(count1*blargeur);
    compteur=0;
    while(compteur<blargeur)
     {
      fgetst(&code,1,fd);
      if(code<128)
       {
        fgetst(&bitmap[posit+compteur],code+1,fd);
        compteur=compteur+code+1;
       }
      else
       if(code>128)
        {
         fgetst(&buffer,1,fd);
         for(count3=compteur;count3<(compteur+257-code);count3++)
          bitmap[posit+count3]=buffer;
         compteur=compteur+257-code;
        }
```

```
      else;
     }
   }
}

PRIVATE void LectBitmap(fd, bitmap, hauteur, largeur, profondeur)
FILE *fd;
UBYTE *bitmap, profondeur;
USHORT hauteur, largeur;
{
 int blargeur=((largeur+15)/16)*2, bplan=hauteur*blargeur;
 int count1, count2, posit;
 for(count1=0;count1<hauteur;count1++)
  for(count2=0;count2<profondeur;count2++)
   {
     posit=count2*bplan+(count1*blargeur);
     fgetst(&bitmap[posit],blargeur,fd);
    }
}

PRIVATE int ChargeIlbm(nom,imagedata)
char *nom;
ILBM_32 *imagedata;
{
 long FORMID=(('F'*256+'O')*256+'R')*256+'M';
 long ILBMID=(('I'*256+'L')*256+'B')*256+'M';
 long BMHDID=(('B'*256+'M')*256+'H')*256+'D';
 long CMAPID=(('C'*256+'M')*256+'A')*256+'P';
 long GRABID=(('G'*256+'R')*256+'A')*256+'B';
 long CRNGID=(('C'*256+'R')*256+'N')*256+'G';
 long CAMGID=(('C'*256+'A')*256+'M')*256+'G';
 long BODYID=(('B'*256+'O')*256+'D')*256+'Y';
 FILE *fd;
 long identificateur,longueur;
 int crngcount, len, bitmaplen, blargeur;
 fd=fopen(nom,"r");
 imagedata->camg=0;
 fgetst((char *) &identificateur,4,fd);
 if(identificateur!=FORMID)
  return 1 ;
 else
  {
   fgetst((char *) &longueur,4,fd);
   fgetst((char *) &identificateur,4,fd);
   if(identificateur!=ILBMID)
    return 1;
   else
    {
     crngcount=0;
     while(fgetst((char *) &identificateur,4,fd))
      {
       fgetst((char *) &longueur,4,fd);
       len=longueur;
       if(identificateur==BMHDID)
        fgetst((char *) &imagedata->bmhd,len,fd);
       else if(identificateur==CMAPID)
        {
         imagedata->cmap= (CMAPelem *) malloc(len);
         fgetst((char *) imagedata->cmap,len,fd);
```

```
        LENCMAP=len/3;
       }
      else if(identificateur==GRABID)
       fgetst((char *) &imagedata->grab,len,fd);
      else if(identificateur==CRNGID)
       fgetst((char *) &imagedata->crng[crngcount++],len,fd);
      else if(identificateur==CAMGID)
       fgetst((char *) imagedata->camg,len,fd);
      else if(identificateur==BODYID)
       {
        blargeur=((imagedata->bmhd.w+15)/16)*2;
        bitmaplen=imagedata->bmhd.h*blargeur*(imagedata->bmhd.nPlanes);
        if((imagedata->body=(USHORT *)
         AllocMem(bitmaplen,MEMF_CHIP | MEMF_CLEAR))==NULL)
          {
           printf("ERREUR manque de CHIP RAM\n");
           free(imagedata->cmap);
           FermerLib();
           exit(1);
          }
        if(imagedata->bmhd.compression==0)
         LectBitmap(fd,imagedata->body,imagedata->bmhd.h,
                    imagedata->bmhd.w,imagedata->bmhd.nPlanes);
        else
         Decompress(fd,imagedata->body,imagedata->bmhd.h,
                     imagedata->bmhd.w,imagedata->bmhd.nPlanes);
       }
     }
    }
  }
 fclose(fd);
}

PRIVATE void ChargerImage(nom,picto,palette,viewmode,scwidth,scheight)
char *nom;
struct Image *picto;
USHORT *palette;
long *viewmode;
USHORT *scwidth, *scheight;
{
 ILBM_32 imagedata;
 int i,len1;
 UBYTE *couleur;
 USHORT *couleur1;
 ChargeIlbm(nom,&imagedata);
 len1=1<<(imagedata.bmhd.nPlanes);
 *scheight=imagedata.bmhd.pageheight;
 *scwidth=imagedata.bmhd.pagewidth;
 picto->ImageData=imagedata.body;
 picto->LeftEdge=imagedata.bmhd.x;
 picto->TopEdge=imagedata.bmhd.y;
 picto->Width=imagedata.bmhd.w;
 picto->Height=imagedata.bmhd.h;
 picto->Depth=imagedata.bmhd.nPlanes;
 picto->PlanePick=31;
 picto->PlaneOnOff=31;
 picto->NextImage=NULL;
 couleur=(UBYTE *) imagedata.cmap;
 couleur1=palette;
```

```
 for(i=0;i<len1;i++) couleur1[i]=0;
 for(i=0;i<len1;i++)
  couleur1[i]=couleur[i*3]*16+couleur[i*3+1]+couleur[i*3+2]/16;
 couleur=NULL;
 free(imagedata.cmap);
 *viewmode=0;
 if(imagedata.camg!=0)
  {
   if(imagedata.camg&0x8000)
    *viewmode=*viewmode&HIRES;
   if(imagedata.camg&0x4)
    *viewmode=*viewmode&LACE;
  }
 else
  {
   if(*scheight>=400)
    *viewmode=*viewmode&HIRES;
   if(*scwidth>=640)
    *viewmode=*viewmode&LACE;
  }
}

PRIVATE void LibererImage(picto)
struct Image *picto;
{
 int blargeur=((picto->Width+15)/16)*2;
 int len=picto->Depth*blargeur*picto->Height;
 FreeMem((char *) picto->ImageData,len);
}

void ChargeFond(nom)
char *nom;
{
 USHORT palette[64],scwidth,scheight;
 long viewmode;
 ChargerImage(nom,&fond,palette,&viewmode,&scwidth,&scheight);
 LoadRGB4(&Screen->ViewPort,palette,LENCMAP);
 NewWindow.LeftEdge = 0;
 NewWindow.TopEdge  = 0;
 NewWindow.Width = scwidth;
 NewWindow.Height = scheight;
 NewWindow.DetailPen = 0;
 NewWindow.BlockPen = 1;
 NewWindow.Title = NULL;
 NewWindow.Flags =ACTIVATE|BORDERLESS;
 NewWindow.IDCMPFlags = GADGETUP;
 NewWindow.Type = CUSTOMSCREEN;
 NewWindow.FirstGadget = NULL;
 NewWindow.CheckMark = NULL;
 NewWindow.Screen = Screen;
 NewWindow.BitMap = NULL;
 NewWindow.MinWidth = 0;
 NewWindow.MinHeight = 0;
 NewWindow.MaxWidth = scwidth;
 NewWindow.MaxHeight = scheight;
 if((Window = (struct window *) OpenWindow(&NewWindow))==NULL)
  {
   CloseScreen(Screen);
   LibererImage(&fond);
```

```
   FermerLib();
   printf("erreur sur la fenetre\n");
   exit(FALSE);
  }
}

void AfficheFond(void)
{
 Move(Window->RPort,0,0);
 DrawImage(Window->RPort,&fond,0,0);
}

void FermeFond(Void)
{
 CloseWindow(Window);
 LibererImage(&fond);
}

void ChargeImage(nom,picto)
char *nom;
struct Image *picto;
{
   USHORT palette[64], scw, sch;
   long viewmodes;
   ChargerImage(nom,picto,palette,&viewmodes,&scw,&sch);
}

void LibereImage(picto)
struct Image *picto;
{
   LibererImage(picto);
}

void AfficheImage(picto, posx, posy)
struct Image *picto;
USHORT posx, posy;
{
 Move(Window->RPort,0,0);
 DrawImage(Window->RPort,picto,posx,posy);
}

void SuperposeImage(picto, bitmap, nbreligne, nbrecolonne, posx, posy)
struct Image *picto;
UBYTE *bitmap;
USHORT nbreligne, nbrecolonne, posx, posy;
{
 int i,j,k, pos, pos1;
 UBYTE *ligne;
 for(i=0;i<picto->Depth;i++)
    for(j=0;j<nbreligne;j++)
       for(k=0;k<(nbrecolonne+7)/8;k++)
       {
        pos1=(i*picto->Height*((picto->Width+15)/16))
             +((j+posy)*((picto->Width+15)/16));
        ligne=&picto->ImageData[pos1];
        pos=(i*nbreligne*nbrecolonne)+(j*nbrecolonne)+k;
        ligne[k+posx]=ligne[k+posx]|bitmap[pos];
       }
}
```

```
/*
 * ---> Module  : Gestionnaire des Gadgets
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Image.h"
#include "libraries/dos.h"
#include "stdlib.h"
#include "H_String.h"
#define MAX_LIGNE  3
#define MAX_COL    30
#define NO_COUL_FOND 33

PRIVATE struct Gadget MonGadget =
{
  NULL,0,0,0,0,GADGHCOMP|GADGIMAGE,
  TOGGLESELECT|RELVERIFY,BOOLGADGET,NULL,
  NULL,NULL,False,NULL,0,NULL
 };

PRIVATE char undobuff[128];

PRIVATE short bord[10];

PRIVATE struct Border StrBord =
{
 0,0,1,0,JAM1,5,bord,NULL
};

PRIVATE struct StringInfo strinfo =
{
 NULL,undobuff,0,0,0,0,0,0,0,0,0,0,NULL
};

PRIVATE struct Gadget ChaineGadget =
{
  NULL,0,0,0,0,GADGHCOMP,
  RELVERIFY,STRGADGET,NULL,
  NULL,NULL,False,(APTR)&strinfo,0,NULL
 };

PRIVATE struct Gad1
{
   char *id;
   struct Gadget *prec;
} *Info1, *Info2;

PRIVATE struct IntuiText txt=
{
 1,1,JAM1,0,0,NULL,NULL,NULL
};

void SelectionneGadget(g)
struct Gadget *g;
{
 g->Flags=g->Flags|SELECTED;
}
```

```
void SelectionneEStrGadget(g)
struct Gadget *g;
{
 ActivateGadget(g,Window,NULL);
}

void SelectionneFStrGadget(g, req)
struct Gadget *g;
struct Requester *req;
{
 ActivateGadget(g,Window,req);
}

void DeselectionneGadget(g)
struct Gadget *g;
{
 g->Flags=g->Flags & ~SELECTED;
}

void ActiveGadget(g)
struct Gadget *g;
{
 g->Flags=g->Flags & ~GADGDISABLED;
}

void DesactiveGadget(g)
struct Gadget *g;
{
 g->Flags=g->Flags|GADGDISABLED;
}

boolean TestSelectionGadget(g)
struct Gadget *g;
{
 return !((g->Flags & ~SELECTED)==(g->Flags));
}

PRIVATE void InitInGadget(bulletxt, palette, coul_car, coul_fond)
struct IntuiText bulletxt[];
struct TextAttr *palette;
USHORT coul_car, coul_fond;
{
 int i;
 for(i=0;i<MAX_LIGNE;i++)
 {
  bulletxt[i].FrontPen=coul_car;
  if(coul_fond == NO_COUL_FOND)
  {
   bulletxt[i].BackPen=0;
   bulletxt[i].DrawMode=JAM1;
  }
  else
  {
   bulletxt[i].BackPen=coul_fond;
   bulletxt[i].DrawMode=JAM2;
  }
  bulletxt[i].LeftEdge=0;
  bulletxt[i].TopEdge=0;
```

```
  bulletxt[i].ITextFont=palette;
  bulletxt[i].IText=NULL;
  if(i<(MAX_LIGNE-1)) bulletxt[i].NextText=&bulletxt[i+1];
  else bulletxt[i].NextText=NULL;
 }
}

PRIVATE void GarnirInGadget(bulletxt, msg, posx, posy, inc)
struct IntuiText bulletxt[];
char *msg;
USHORT posx, posy, inc;
{
   char dec_msg[MAX_LIGNE][MAX_COL];
   int i,j,k;
   for(i=0;i<MAX_LIGNE;i++) dec_msg[i][0]='\0';
   i=j=0;
   while((msg[i]!='\0')&&(j<MAX_LIGNE))
   {
      k=0;
      while((msg[i]!='\0')&&(msg[i]!='#')&&(k<25))
      {
         dec_msg[j][k]=msg[i];
         k++;
         i++;
      }
      dec_msg[j][k]='\0';
      j++;
      if(msg[i]!='\0')i++;
   }
   for(i=0;i<MAX_LIGNE;i++)
   {
      if(bulletxt[i].IText==NULL)
         bulletxt[i].IText=malloc(strlen(dec_msg[i])+1);
      else
      {
         free(bulletxt[i].IText);
         bulletxt[i].IText=malloc(strlen(dec_msg[i])+1);
      }
      strcpy(bulletxt[i].IText,dec_msg[i]);
      bulletxt[i].LeftEdge=posx;
      bulletxt[i].TopEdge=posy+(i*inc);
   }
}

void AfficheInGadget(msg, g, posx, posy, coul_car, coul_fond, police, dec)
char *msg;
struct Gadget *g;
USHORT posx, posy, coul_car, coul_fond, dec;
struct TextAttr *police;
{
   if(g->GadgetText==NULL)
     g->GadgetText=(struct IntuiText *)
                   malloc(MAX_LIGNE * sizeof(struct IntuiText));
   InitInGadget(g->GadgetText, police, coul_car, coul_fond);
   GarnirInGadget(g->GadgetText, msg, posx, posy, dec);
}

void LibereInGadget(bulletxt)
struct IntuiText bulletxt[];
```

```
{
 int i;
 for(i=0;i<MAX_LIGNE;i++)
    if((bulletxt!=NULL)&&(bulletxt[i].IText!=NULL))
      free(bulletxt[i].IText);
 if(bulletxt!=NULL) free(bulletxt);
}

void AfficheBulle(msg, g)
char *msg;
struct Gadget *g;
{
   AfficheInGadget(msg, g, 4, 8, 12, NO_COUL_FOND, NULL, 10);
}

void OuvreGadget(gadget,fichier)
char *fichier;
struct Gadget *gadget;
{
 struct Image *picto;
 *gadget=MonGadget;
 picto=(struct Image *)malloc(sizeof(struct Image));
 ChargeImage(fichier,picto);
 gadget->Width=picto->Width;
 gadget->Height=picto->Height;
 gadget->GadgetRender=(APTR) picto;
}

void FermeGadget(gadget)
struct Gadget *gadget;
{
 LibereImage((struct Image *)gadget->GadgetRender);
}

void AjouteEGadget(g, gb, nom, posx, posy)
struct Gadget *g, gb;
char *nom;
USHORT posx, posy;
{
 *g=gb;
 g->LeftEdge=posx;
 g->TopEdge=posy;
 (struct Gad1 *)(g->UserData)=(struct Gad1 *)malloc(sizeof(struct Gad1));
 ((struct Gad1 *)(g->UserData))->id=malloc(strlen(nom)+1);
 ((struct Gad1 *)(g->UserData))->prec=malloc(sizeof(struct Gadget));
 strcpy(((struct Gad1 *)(g->UserData))->id,nom);
 ((struct Gad1 *)(g->UserData))->prec=NULL;
 g->NextGadget=Window->FirstGadget;
 if(Window->FirstGadget!=NULL)
 {
  Info1=(struct Gad1 *)((Window->FirstGadget)->UserData);
  Info1->prec = g;
 }
 Window->FirstGadget=g;
 ActiveGadget(g);
}

void AjouteFGadget(g, gb, req, nom, posx, posy, type)
struct Gadget *g, gb;
```

```
struct Requester *req;
char *nom;
USHORT posx, posy;
boolean type;
{
 *g=gb;
 g->LeftEdge=posx;
 g->TopEdge=posy;
 (struct Gad1 *)(g->UserData)=(struct Gad1 *)malloc(sizeof(struct Gad1));
 ((struct Gad1 *)(g->UserData))->id=malloc(strlen(nom)+1);
 ((struct Gad1 *)(g->UserData))->prec=malloc(sizeof(struct Gadget));
 strcpy(((struct Gad1 *)(g->UserData))->id,nom);
 ((struct Gad1 *)(g->UserData))->prec=NULL;
 g->NextGadget=req->ReqGadget;
 g->GadgetType=g->GadgetType|REQGADGET;
 if(type)
  g->Activation=g->Activation|ENDGADGET;
 if(req->ReqGadget!=NULL)
 {
  Info1=(struct Gad1 *) ((req->ReqGadget)->UserData);
  Info1->prec = g;
 }
 req->ReqGadget=g;
 ActiveGadget(g);
}

void AjouteEStrGadget(g, nom, posx, posy, nbrecar, chaine)
struct Gadget *g;
char *nom, *chaine;
USHORT posx, posy, nbrecar;
{
 ++nbrecar;
 *g=ChaineGadget;
 ((struct StringInfo *) g->SpecialInfo)->Buffer=chaine;
 ((struct StringInfo *) g->SpecialInfo)->MaxChars=nbrecar;
 ((struct StringInfo *) g->SpecialInfo)->BufferPos=strlen(chaine);
 g->LeftEdge=posx+2;
 g->TopEdge=posy+2;
 g->Width=nbrecar*8;
 g->Height=10;
 (struct Gad1 *)(g->UserData)=(struct Gad1 *)malloc(sizeof(struct Gad1));
 ((struct Gad1 *)(g->UserData))->id=malloc(strlen(nom)+1);
 ((struct Gad1 *)(g->UserData))->prec=malloc(sizeof(struct Gadget));
 strcpy(((struct Gad1 *)(g->UserData))->id,nom);
 ((struct Gad1 *)(g->UserData))->prec=NULL;
 g->NextGadget=Window->FirstGadget;
 if(Window->FirstGadget!=NULL)
 {
  Info1=(struct Gad1 *)((Window->FirstGadget)->UserData);
  Info1->prec = g;
 }
 Window->FirstGadget=g;
 ActiveGadget(g);
 StrBord.LeftEdge=posx;
 StrBord.TopEdge=posy;
 bord[0]=0;
 bord[1]=0;
 bord[2]=g->Width+4;
 bord[3]=0;
```

```
 bord[4]=g->Width+4;
 bord[5]=g->Height+4;
 bord[6]=0;
 bord[7]=g->Height+4;
 bord[8]=0;
 bord[9]=0;
 DrawBorder(Window->RPort,&StrBord,0,0);
}

void AjouteFStrGadget(g, req, nom, posx, posy, nbrecar, chaine, type)
struct Gadget *g;
struct Requester *req;
char *nom, *chaine;
USHORT posx, posy, nbrecar;
boolean type;
{
 *g=ChaineGadget;
 ((struct StringInfo *) g->SpecialInfo)->Buffer=chaine;
 ((struct StringInfo *) g->SpecialInfo)->MaxChars=nbrecar;
 ((struct StringInfo *) g->SpecialInfo)->BufferPos=strlen(chaine);
 g->LeftEdge=posx;
 g->TopEdge=posy;
 (struct Gad1 *)(g->UserData)=(struct Gad1 *)malloc(sizeof(struct Gad1));
 ((struct Gad1 *)(g->UserData))->id=malloc(strlen(nom)+1);
 ((struct Gad1 *)(g->UserData))->prec=malloc(sizeof(struct Gadget));
 strcpy(((struct Gad1 *)(g->UserData))->id,nom);
 ((struct Gad1 *)(g->UserData))->prec=NULL;
 g->NextGadget=req->ReqGadget;
 g->GadgetType=g->GadgetType|REQGADGET;
 if(type)
  g->Activation=g->Activation|ENDGADGET;
 if(req->ReqGadget!=NULL)
 {
  Info1=(struct Gad1 *) ((req->ReqGadget)->UserData);
  Info1->prec = g;
 }
 req->ReqGadget=g;
 ActiveGadget(g);
}

void SupprimeEGadget(g)
struct Gadget *g;
{
 if(g->NextGadget!=NULL)
 {
  Info1=(struct Gad1 *) (g->NextGadget)->UserData;
  Info2=(struct Gad1 *) g->UserData;
  Info1->prec=Info2->prec;
 }
 if(((struct Gad1 *)(g->UserData))->prec!=NULL)
 {
  Info1=(struct Gad1 *) g->UserData;
  ((struct Gadget *)(Info1->prec))->NextGadget=g->NextGadget;
 }
 else
  Window->FirstGadget=g->NextGadget;
 g->NextGadget=NULL;
 ((struct Gad1 *)(g->UserData))->prec=NULL;
 DesactiveGadget(g);
```

```
 free(((struct Gadl *)(g->UserData))->id);
 free(((struct Gadl *)(g->UserData))->prec);
 free((struct Gadl *)g->UserData);
 if(g->GadgetText!=NULL)
   LibereInGadget(g->GadgetText);
 g->UserData=NULL;
 g->GadgetText=NULL;
}

void SupprimeFGadget(g,req)
struct Gadget *g;
struct Requester *req;
{
 if(g->NextGadget!=NULL)
 {
  Info1=(struct Gadl *) (g->NextGadget)->UserData;
  Info2=(struct Gadl *) g->UserData;
  Info1->prec=Info2->prec;
 }
 if(((struct Gadl *)(g->UserData))->prec!=NULL)
 {
  Info1=(struct Gadl *) g->UserData;
  ((struct Gadget *)(Info1->prec))->NextGadget=g->NextGadget;
 }
 else
  req->ReqGadget=g->NextGadget;
 g->NextGadget=NULL;
 ((struct Gadl *)(g->UserData))->prec=NULL;
 DesactiveGadget(g);
 free(((struct Gadl *)(g->UserData))->id);
 free(((struct Gadl *)(g->UserData))->prec);
 free((struct Gadl *) g->UserData);
 if(g->GadgetText!=NULL)
   LibereInGadget(g->GadgetText);
 g->UserData=NULL;
 g->GadgetText=NULL;
}

void RaffraichitEGadget(void)
{
 RefreshGadgets(Window->FirstGadget,Window,NULL);
}

void RaffraichitFGadget(req)
struct Requester *req;
{
 RefreshGadgets(req->ReqGadget,Window,req);
}

char *IdGadget(g)
struct Gadget *g;
{
 return(((struct Gadl*)(g->UserData))->id);
}

void AfficheTitre(Titre)
char *Titre;
{
 static struct IntuiText Title;
```

```
 Title=txt;
 Title.IText=malloc(strlen(Titre)+1);
 strcpy(Title.IText,Titre);
 PrintIText(Window->RPort, &Title, (320-IntuiTextLength(&Title))/2, 12);
 free(Title.IText);
}

void AfficheTexte(ATexte,posx,posy,coul_car,coul_fond,texteattr)
char *ATexte;
USHORT posx, posy;
USHORT coul_car, coul_fond;
struct TextAttr *texteattr;
{
 static struct IntuiText *aTexte;
 aTexte=malloc(MAX_LIGNE * sizeof(struct IntuiText));
 InitInGadget(aTexte, texteattr, coul_car, coul_fond);
 GarnirInGadget(aTexte, ATexte, 0, 0, 15);
 PrintIText(Window->RPort, aTexte, posx, posy);
 LibereInGadget(aTexte);
}

struct Image *GetImgInGadget(g)
struct Gadget *g;
{
 return (struct Image *) g->GadgetRender;
}

void SetImgInGadget(g, img)
struct Gadget *g;
struct Image *img;
{
 g->GadgetRender=(APTR)img;
}

void ChangePosGadget(g, posx, posy)
struct Gadget *g;
USHORT posx, posy;
{
   g->LeftEdge=posx;
   g->TopEdge=posy;
}
```

```
/*
 * ---> Fichier : Gestionnaire des accès disques
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "dos.h"
#include "stdio.h"
#include "H_Privacy.h"
#include "H_Gadget.h"
#include "H_Types.h"

/*********************************************/
/*                                           */
/* Definition des status d'acces aux fichiers */
/*                                           */
/*********************************************/

#define F_OK          0
#define F_PAS_OUVERT  1
#define F_PAS_FERME   2
#define F_INEXISTANT  3
#define F_MAUV_CLE    4
#define F_MAUV_POS    5
#define F_FIN_FICH    6
#define F_ERR_ECRIT   7
#define F_ERR_INCONNUE 8

/*************************************************/
/*                                               */
/* Definition des modes d'ouverture d'un fichier */
/*                                               */
/*************************************************/

#define R           "rb"
#define W           "wb"
#define RW          "rb+"

/*********************************************************/
/*                                                       */
/* Definition des positions pour les acces aux fichiers */
/*                                                       */
/*********************************************************/

#define POS_FIN     (-1)
#define POS_COURANT (-2)
#define POS_DEBUT   (-3)
PUBLIC st_msg tab_msg[nbre_msg];

/*****************************************/
/*                                       */
/* Fonction de vérification de disquette */
/*                                       */
/*****************************************/

short pbd_teste_disque(disk_name)
char *disk_name;
{
```

```
    if(getasn(disk_name))
       return True;
    else
       return False;
}

PRIVATE int pbd_teste_disque_protege(disk_name)
char *disk_name;
{
    struct DISKINFO info;
    getdfs(disk_name,&info);
    return !(info.id_DiskState=(info.id_DiskState & ~0xFFFFFFFDL)>>1);
}

PRIVATE void teste_disque(disk_name)
char *disk_name;
{
   st_msg string;
   strcpy(string.nom,tab_msg[30].nom);
   strcat(string.nom,disk_name);
   while(!pbd_teste_disque(disk_name))
      ConfirmationOK(string);
}

PRIVATE void teste_disque_protege(disk_name)
char *disk_name;
{
   st_msg string;
   strcpy(string.nom,tab_msg[31]);
   strcat(string.nom,disk_name);
   strcat(string.nom,tab_msg[32]);
   while(pbd_teste_disque_protege("df0:"))
   {
      ConfirmationOK(string);
      teste_disque(disk_name);
   }
}

/*********************************************************/

PRIVATE void stringcpy(dest,source,lng)
char *dest, *source;
short lng;
{
   short i;
   for(i=0;i<lng;i++)
      dest[i]=source[i];
}

PRIVATE void cls(void)
{
   printf("\c0x1b[0;0H%c[J",0x1b);
}

PRIVATE short stringcmp(st1,st2,lng)
char *st1, *st2;
short lng;
{
   short i=0;
```

```
   while((st1[i]==st2[i])&&(i<lng)) i++;
   if(i==lng)
      return 0;
   else
      if(st1[i]<st2[i])
         return -1;
      else
         return 1;
}

PRIVATE short ltos(lval)
long lval;
{
   short tab=lval;
   return tab;
}

short poscourante(fd)
FILE *fd;
{
    long retval=ftell(fd);
    return ltos(retval);
}

/*********************************************************/

short filelength(fd)
FILE *fd;
{
    short retval,pos=poscourante(fd);
    fseek(fd,0,2);
    retval=poscourante(fd);
    fseek(fd,pos,0);
    return retval;
}

PRIVATE short pbd_lire_fichier(fd,record,pos,sizeofrec)
FILE *fd;
char *record;
short *pos,sizeofrec;
{
    if(*pos==POS_DEBUT)
      fseek(fd,0,0);
    else if(*pos!=POS_COURANT)
      fseek(fd,*pos,0);
    *pos=poscourante(fd);
    return (short)fread(record,sizeofrec,1,fd);
}

PRIVATE short pbd_ecrire_fichier(fd,record,pos,sizeofrec)
FILE *fd;
char *record;
short *pos,sizeofrec;
{
    if(*pos==POS_FIN)
      fseek(fd,0,2);
    else
      if(*pos==POS_DEBUT)
        fseek(fd,0,0);
```

```
      else
        if(*pos!=POS_COURANT)
          fseek(fd,*pos,0);
    *pos=poscourante(fd);
    return (short)fwrite(record,sizeofrec,1,fd);
}

PRIVATE short pbd_fournir_sur_id(fd,record,ident,pos,sizeofid,sizeofrec)
FILE *fd;
char *record,*ident;
short sizeofid,sizeofrec,*pos;
{
    char *cle;
    short deb=0,fin=(filelength(fd)/sizeofrec)-1,cour;
    cour=((deb+fin)/2)*sizeofrec;
    cle=(char *) malloc(sizeofid);
    stringcpy(cle,ident,sizeofid);
    stringcpy(ident,"",1);
    pbd_lire_fichier(fd,record,&cour,sizeofrec);
    while((stringcmp(ident,cle,sizeofid)!=0)&&(deb<=fin))
    {
       if(stringcmp(ident,cle,sizeofid)<0)
          deb=(cour/sizeofrec)+1;
       else
          fin=(cour/sizeofrec)-1;
       cour=((deb+fin)/2)*sizeofrec;
       stringcpy(ident,"",sizeofid);
       pbd_lire_fichier(fd,record,&cour,sizeofrec);
    }
    *pos=cour;
    if(stringcmp(ident,cle,sizeofid)==0)
    {
      free(cle);
      return F_OK;
    }
    else
    {
      free(cle);
      return F_MAUV_CLE;
    }
}

PRIVATE FILE *pbd_ouvrir_fichier(filename,mode)
char *filename, *mode;
{
    return fopen(filename,mode);
}

PRIVATE short pbd_fermer_fichier(fd)
FILE *fd;
{
    return (short) fclose(fd);
}

PRIVATE short pbd_effacer_fichier(filename)
char *filename;
{
    return (short) remove(filename);
}
```

```
PRIVATE short pbd_renommer_fichier(old,new)
char *old,*new;
{
    return (short) rename(old,new);
}

/*******************************************************/

FILE *ouvrir_fichier(filename,mode,disk_name)
char *filename, *disk_name, *mode;
{
    char nom[128];
    FILE *fd;
    strcpy(nom,disk_name);
    strcat(nom,":");
    strcat(nom,filename);
    fd=pbd_ouvrir_fichier(nom,mode);
    if(fd==NULL) fd=(FILE *)F_PAS_OUVERT;
    return fd;
}

short lire_fichier(fd, disk_name, record, sizeofrec, pos)
FILE *fd;
char *disk_name, *record;
short sizeofrec, *pos;
{
   short retval=F_OK;
   if(*pos>0) *pos=(*pos)*sizeofrec;
   if(pbd_lire_fichier(fd, record, pos, sizeofrec)==0)
      retval=F_FIN_FICH;
   *pos=(*pos)/sizeofrec;
   return retval;
}

short lire_sur_id(fd, disk_name, record, ident, sizeofid, sizeofrec, pos)
FILE *fd;
char *disk_name, *record, *ident;
short sizeofid, sizeofrec, *pos;
{
   short retval=F_OK;
   if(*pos>0) *pos=(*pos)*sizeofrec;
   retval=pbd_fournir_sur_id(fd, record, ident, pos, sizeofid, sizeofrec);
   *pos=(*pos)/sizeofrec;
   return retval;
}

short ecrire_fichier(fd, disk_name, record, sizeofrec, pos)
FILE *fd;
char *disk_name, *record;
short sizeofrec, *pos;
{
   short retval=F_OK;
   if(*pos>0) *pos=(*pos)*sizeofrec;
   if(pbd_ecrire_fichier(fd, record, pos, sizeofrec)==0)
      retval=F_ERR_ECRIT;
   *pos=(*pos)/sizeofrec;
   return retval;
}
```

```
short ecrire_sur_id(fd, disk_name, record, ident, sizeofid, sizeofrec, pos)
FILE *fd;
char *disk_name, *record, *ident;
short sizeofid, sizeofrec, *pos;
{
   short retval=F_OK;
   char *record1;
   if(*pos>0) *pos=(*pos)*sizeofrec;
   record1=(char *) malloc(sizeofrec);
   stringcpy(record1,record,sizeofrec);
   if((retval=pbd_fournir_sur_id(fd, record, ident, pos, sizeofid, sizeofrec))==F_OK)
      if(pbd_ecrire_fichier(fd, record1, pos, sizeofrec)==0)
         retval=F_ERR_ECRIT;
   *pos=(*pos)/sizeofrec;
   free(record1);
   return retval;
}

short effacer_fichier(fd, disk_name, filename, mode, sizeofrec, pos)
FILE *fd;
char *disk_name, *filename, *mode;
short sizeofrec, *pos;
{
   short retval=F_OK, posb=poscourante(fd), posc;
   char *record;
   short lng=filelength(fd);
   FILE *fw;
   record=(char *)malloc(lng);
   if(*pos>=0) posb=(*pos)*sizeofrec;
   else if(*pos==-1) posb=lng-sizeofrec;
   else if(*pos==-3) posb=0;
   posc=POS_DEBUT;
   pbd_lire_fichier(fd, record, &posc, lng);
   fw=pbd_ouvrir_fichier("TAMPON",W);
   posc=POS_FIN;
   if(pbd_ecrire_fichier(fw, record, &posc, posb)==0)
      retval=F_ERR_ECRIT;
   else
   {
    posc=POS_FIN;
    if(pbd_ecrire_fichier(fw, &record[posb+sizeofrec], &posc,
                          lng-(posb+sizeofrec))==0)
         retval=F_ERR_ECRIT;
   }
   *pos=(*pos)/sizeofrec;
   pbd_fermer_fichier(fw);
   pbd_fermer_fichier(fd);
   pbd_effacer_fichier(filename);
   pbd_renommer_fichier("TAMPON",filename);
   fd=pbd_ouvrir_fichier(filename,mode);
   fseek(fd,posb,0);
   return retval;
}

short modifie_fichier(fd, disk_name, record, sizeofrec, pos)
FILE *fd;
char *disk_name, *record;
short sizeofrec, *pos;
```

```
{
   short retval=F_OK;
   if(*pos>0) *pos=(*pos)*sizeofrec;
   if(pbd_ecrire_fichier(fd, record, pos, sizeofrec)==0)
      retval=F_ERR_ECRIT;
   *pos=(*pos)/sizeofrec;
   return retval;
}

short modifie_sur_id(fd, disk_name, record, ident, sizeofid, sizeofrec, pos)
FILE *fd;
char *disk_name, *record, *ident;
short sizeofid, sizeofrec, *pos;
{
   short retval=F_OK;
   char *record1;
   record1=(char *) malloc(sizeofrec);
   stringcpy(record1,record);
   if(*pos>0) *pos=(*pos)*sizeofrec;
   if((retval=pbd_fournir_sur_id(fd, record, ident, pos, sizeofid, sizeofrec))==F_OK)
      if(pbd_ecrire_fichier(fd, record1, pos, sizeofrec)==0)
         retval=F_ERR_ECRIT;
   *pos=(*pos)/sizeofrec;
   return retval;
}

short fermer_fichier(fd, disk_name)
FILE *fd;
char *disk_name;
{
   short retval=F_OK;
   if(pbd_fermer_fichier(fd)!=0)
      retval=F_PAS_FERME;
   return retval;
}

short supprimer_fichier(filename, disk_name)
char *filename, *disk_name;
{
   short retval=F_OK;
   if(pbd_effacer_fichier(filename)!=0)
      retval=F_ERR_INCONNUE;
   return retval;
}

short renommer_fichier(oldfname, newfname, disk_name)
char *newfname, *oldfname, *disk_name;
{
   short retval=F_OK;
   if(pbd_renommer_fichier(oldfname, newfname)!=0)
      retval=F_ERR_INCONNUE;
   return retval;
}
```

```
/*
 * ---> Fichier : Gestionnaire d'impression
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include <exec/types.h>
#include <devices/printer.h>
#include <exec/io.h>
#include "H_Privacy.h"
#include "H_Image.h"

union printerIO
{
 struct IOStdReq    ios;
 struct IODRPReq    iodrp;
 struct IOPrtCmdReq iopc;
};

static UBYTE *prtErrorText[] =
{
 "OK",
 "ANNULATION PAR L'UTILISATEUR",
 "PAS IMPRIMANTE GRAPHIQUE",
 "NE PEUT PAS RECEVOIR",
 "MAUVAISES DIMENSIONS",
 "NE PEUT PAS RECEVOIR",
 "MANQUE DE MEMOIRE POUR LES VARIABLES",
 "PAS DE MEMOIRE POUR LE BUFFER"
};

union printerIO *printerReq;

union printerIO *CreatePrtReq()
{
 struct MsgPort *prtport;
 union printerIO *request;
 if(!(prtport=CreatePort(NULL,0)))
   return 0;
 if(!(request=(union printerIO *)CreateExtIO(prtport,
              sizeof(union printerIO))))
 {
  DeletePort(prtport);
  return 0;
 }
 return request;
}

void DeletePrtReq(request)
union printerIO *request;
{
 struct MsgPort *prtport;
 prtport=request->ios.io_Message.mn_ReplyPort;
 DeleteExtIO((struct IORequest *) request);
 DeletePort(prtport);
}

int OpenPrinter(request)
```

```
union printerIO *request;
{
 return(OpenDevice("printer.device",0,(struct IORequest *)request,0));
}

void DumpRPort(request, rastport, colormap, modes, sx, sy, sw, sh, dc, dr, s)
union printerIO *request;
struct RastPort *rastport;
struct ColorMap *colormap;
ULONG modes;
UWORD sx, sy, sw, sh;
LONG dc, dr;
UWORD s;
{
 request->iodrp.io_Command = PRD_DUMPRPORT;
 request->iodrp.io_RastPort = rastport;
 request->iodrp.io_ColorMap = colormap;
 request->iodrp.io_Modes = modes;
 request->iodrp.io_SrcX = sx;
 request->iodrp.io_SrcY = sy;
 request->iodrp.io_SrcWidth = sw;
 request->iodrp.io_SrcHeight = sh;
 request->iodrp.io_DestCols = dc;
 request->iodrp.io_DestRows = dr;
 request->iodrp.io_Special = s;
 SendIO((struct IORequest *) request);
}

int print(rport, posx, posy, width, height)
struct RastPort *rport;
short posx, posy, width, height;
{
 struct IntuiMessage *msg;
 struct MsgPort *port;
 struct RastPort *rp;
 struct ViewPort *vp;
 ULONG usersig, printersig, signal;
 boolean fin = False;

 if(!(printerReq = CreatePrtReq()))
 {
  printf("Erreur d'imprimante creation\n");
  return 1;
 }
 else
 if(OpenPrinter(printerReq)!=0)
 {
  DeletePrtReq(printerReq);
  printf("Erreur d'imprimante ouverture\n");
  return 1;
 }
 else
 {
  port = printerReq->ios.io_Message.mn_ReplyPort;
  usersig = 1 << Window->UserPort->mp_SigBit;
  printersig = 1 << port->mp_SigBit;
  rp = rport;
  vp = &Screen->ViewPort;
  DumpRPort(printerReq, rp, vp->ColorMap, vp->Modes,
```

```
            (UWORD)posx, (UWORD) posy, (UWORD) width, (UWORD) height,
            0L, 0L, SPECIAL_ASPECT|SPECIAL_NOFORMFEED|SPECIAL_FULLROWS);
    signal = Wait(usersig | printersig);
    if(signal & usersig)
      while(msg = (struct IntuiMessage *)GetMsg(Window->UserPort))
      {
       ReplyMsg((struct Message *)msg);
       AbortIO(printerReq);
       WaitIO(printerReq);
      }
    if(signal & printersig)
      while((struct MsgPort *)GetMsg(port));
    CloseDevice((struct IORequest *)printerReq);
    DeletePrtReq(printerReq);
  }
}
```

```
/*
 * ---> Module  : Gestionnaire des sons
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Privacy.h"
#include <exec/types.h>
#include <devices/narrator.h>
#define LONG_PHRASE 512

PRIVATE BYTE audio_chan[] = {3, 5, 10, 12};
PRIVATE struct MsgPort write_port;
PRIVATE struct narrator_rb  Voice_io;
PRIVATE UBYTE NarratorOpenError = -1;
PUBLIC  boolean VOIX;

PRIVATE void CleanUp(void)
{
   if (write_port.mp_SigBit != -1)
     FreeSignal(write_port.mp_SigBit);
   if (NarratorOpenError==0)
     CloseDevice((struct IORequest *) &Voice_io);
}

void Prononce (string)
char *string;
{
   ULONG Signals;
   boolean rep=False;
   if(VOIX)
   {
      if ((NarratorOpenError =
      OpenDevice("narrator.device",0, (struct IORequest *) &Voice_io, 0)) !=0)
      {
         printf("Erreur d'ouverture de Narrator");
         CleanUp();
         exit(1);
      }
      if ((write_port.mp_SigBit = AllocSignal(-1)) == -1)
      {
         printf("Erreur d'Allocation du signal");
         CleanUp();
         exit(1);
      }
      write_port.mp_Node.ln_Name = "speech_write";
      write_port.mp_Node.ln_Type = NT_MSGPORT;
      write_port.mp_Flags = PA_SIGNAL;
      write_port.mp_SigTask = (struct Task *)FindTask(NULL);
      NewList(&write_port.mp_MsgList);
      Voice_io.message.io_Command = (UWORD) CMD_WRITE;
      Voice_io.message.io_Offset = 0;
      Voice_io.message.io_Data = (APTR) string;
      Voice_io.message.io_Length = (ULONG) strlen(string);
      Voice_io.message.io_Message.mn_Node.ln_Type = NT_MSGPORT;
      Voice_io.message.io_Message.mn_Length = sizeof(Voice_io);
      Voice_io.message.io_Message.mn_ReplyPort = &write_port;
      Voice_io.message.io_Unit = 0;
```

```
        Voice_io.rate = 150;
        Voice_io.pitch = 100;
        Voice_io.mode = 0;
        Voice_io.sex=0;
        Voice_io.ch_masks = (UBYTE *)audio_chan;
        Voice_io.nm_masks = sizeof(audio_chan);
        Voice_io.volume = 64;
        Voice_io.sampfreq=22000;
        Voice_io.mouths=0;
        SendIO((struct IORequest *) &Voice_io);
        while(rep==False)
        {
           Signals = Wait(
           (1L << Voice_io.message.io_Message.mn_ReplyPort->mp_SigBit));
           if(Signals & (1L <<
              Voice_io.message.io_Message.mn_ReplyPort->mp_SigBit))
           {
              if(Voice_io.message.io_Error == -2)
                Voice_io.message.io_Error = 0;
              if(Voice_io.message.io_Error != 0)
                printf("Erreur de Narator : (%ld)",Voice_io.message.io_Error);
              Voice_io.message.io_Error = 0;
              CleanUp();
              rep=True;
           }
        }
     }
  }
```

```
/*
 * ---> Module  : Gestionnaire des chaines de caractères
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

char *strcpy(s1,s2)
char *s1, *s2;

{
   short i;
   for(i=0;s2[i]!='\0';i++)
       s1[i]=s2[i];
   s1[i]='\0';
   return s1;
}

int strlen(s1)
char *s1;

{
   int i;
   for(i=0;s1[i]!='\0';i++);
   return i;
}

char *strcat(s1,s2)
char *s1, *s2;

{
   int i , j=strlen(s1);
   for(i=0;s2[i]!='\0';i++)
       s1[j++]=s2[i];
   s1[j]='\0';
   return s1;
}

int strcmp(s1,s2)
char *s1, *s2;

{
   int i;
   for(i=0;((s1[i]==s2[i])&&(s1[i]!='\0')&&(s2[i]!='\0'));i++);
   if((s1[i]=='\0')&&(s2[i]!='\0'))
     return -1;
   else if((s2[i]=='\0')&&(s1[i]!='\0'))
     return 1;
   else
     return s1[i]-s2[i];
}
```

```
/*
 * ---> Module  : Utilitaires de conversion
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */
#include "H_Privacy.h"
char *itoa(entier)
int entier;
{
   int i=0,j=0;
   char *retval,retvalb[6];
   retval=(char *) malloc(6);
   if(entier<0)
   {
      j=1;
      entier = -entier;
   }
   do
   {
      retvalb[i++]=entier%10+'0';
   } while ((entier /= 10) > 0);
   if(j==1)
      retvalb[i++]='-';
   for(j=0;j<i;j++)
      retval[j]=retvalb[i-j-1];
   retval[i]='\0';
   return retval;
}

int dtoi(nbre)
double nbre;
{
 int retval,dig,sign;
 stcd_i(fcvt(nbre,0,&dig,&sign),&retval);
 return retval;
}

boolean eval(st1,op,st2)
char *st1, *op, *st2;
{
 short ev=strcmp(st1,st2);
 if(ev<0)
   if((!strcmp(op,"<"))||(!strcmp(op,"<="))||(!strcmp(op,"=<")))
     return True;
   else
     return False;
 else if(ev>0)
   if((!strcmp(op,">"))||(!strcmp(op,">="))||(!strcmp(op,"=>")))
     return True;
   else
     return False;
 else
   if((!strcmp(op,"="))||(!strcmp(op,"<="))||(!strcmp(op,">="))||
      (!strcmp(op,"=<"))||(!strcmp(op,"=>")))
     return True;
   else
     return False;
}
```

```
/*
 * ---> Fichier : Initialisation des sons des nombres
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Privacy.h"
#include "H_Types.h"
#include "H_Types2.h"
PUBLIC char tab_unite[17][max_unite];
PUBLIC char tab_dizaine[11][max_dizaine];
void GarnirTab_unite(void)
{
   strcpy(tab_unite[0],"ZEYDXROW");
   strcpy(tab_unite[1],"AX4N");
   strcpy(tab_unite[2],"DUH2");
   strcpy(tab_unite[3],"THRUHAA");
   strcpy(tab_unite[4],"KAETR");
   strcpy(tab_unite[5],"SIHNK");
   strcpy(tab_unite[6],"SIYS");
   strcpy(tab_unite[7],"SEHT");
   strcpy(tab_unite[8],"WIXT");
   strcpy(tab_unite[9],"NAEF");
   strcpy(tab_unite[10],"DIYS");
   strcpy(tab_unite[11],"OW7S");
   strcpy(tab_unite[12],"DUWSS");
   strcpy(tab_unite[13],"TREH2");
   strcpy(tab_unite[14],"KAETOWR2");
   strcpy(tab_unite[15],"KIH9NZ");
   strcpy(tab_unite[16],"SEH9Z");
}

void GarnirTab_dizaine(void)
{
   strcpy(tab_dizaine[0],"PRAXMIYEY");
   strcpy(tab_dizaine[1],"DIYS");
   strcpy(tab_dizaine[2],"VAE9NGT");
   strcpy(tab_dizaine[3],"THRAH9NTH");
   strcpy(tab_dizaine[4],"KAERAH5NT");
   strcpy(tab_dizaine[5],"SIHNKAH5NT");
   strcpy(tab_dizaine[6],"SWAESAH5NT");
   strcpy(tab_dizaine[7],"SEHPTAH5NT");
   strcpy(tab_dizaine[8],"KAETRAXVAE9NGT");
   strcpy(tab_dizaine[9],"NOHNAH5NT");
   strcpy(tab_dizaine[10],"SAA");
}
```

```
/*
 * ---> Fichier  : Article
 * ---> Type     : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_Types2.h"

/*********************************/
void initialiser_tabart(tab_article)
/*********************************/

tarticle *tab_article;

{
   strcpy(tab_article[0].nom,"");
   strcpy(tab_article[0].son,"");
   strcpy(tab_article[1].nom,"le");
   strcpy(tab_article[1].son,"L AX3");
   strcpy(tab_article[2].nom,"la");
   strcpy(tab_article[2].son,"L AA");
   strcpy(tab_article[3].nom,"les");
   strcpy(tab_article[3].son,"L EH");
   strcpy(tab_article[4].nom,"l'");
   strcpy(tab_article[4].son,"L ");
   strcpy(tab_article[5].nom,"un");
   strcpy(tab_article[5].son,"UHN");
   strcpy(tab_article[6].nom,"une");
   strcpy(tab_article[6].son,"UN2");
   strcpy(tab_article[7].nom,"des");
   strcpy(tab_article[7].son,"D EH2");
   strcpy(tab_article[8].nom,"de la");
   strcpy(tab_article[8].son,"DAXLAA");
   strcpy(tab_article[9].nom,"du");
   strcpy(tab_article[9].son,"DUH");
   strcpy(tab_article[10].nom,"de");
   strcpy(tab_article[10].son,"DAX");
   strcpy(tab_article[11].nom,"d'");
   strcpy(tab_article[11].son,"D ");
}

/*********************************/
void initialiser_tab_cond(tab_cond)
/*********************************/

tconditionnement *tab_cond;

{
   strcpy(tab_cond[0].nom,"");
   strcpy(tab_cond[0].son,"");
   strcpy(tab_cond[1].nom,"paquet");
   strcpy(tab_cond[1].son,"PAAKEH");
   strcpy(tab_cond[2].nom,"pot");
   strcpy(tab_cond[2].son,"PAO");
   strcpy(tab_cond[3].nom,"boîte");
   strcpy(tab_cond[3].son,"BWAAT");
   strcpy(tab_cond[4].nom,"sachet");
   strcpy(tab_cond[4].son,"SAASH EH");
```

```
    strcpy(tab_cond[5].nom,"sac");
    strcpy(tab_cond[5].son,"SAA K");
    strcpy(tab_cond[6].nom,"gousse");
    strcpy(tab_cond[6].son,"G UW SS");
    strcpy(tab_cond[7].nom,"bouteille");
    strcpy(tab_cond[7].son,"BUW TEHY");
    strcpy(tab_cond[8].nom,"botte");
    strcpy(tab_cond[8].son,"BAOT");
    strcpy(tab_cond[9].nom,"barquette");
    strcpy(tab_cond[9].son,"BAARKEH2T");
}
```

```
/*
 * ---> Fichier : Conversion
 * ---> Type    : Source complète
 * ---> Auteurs : Topet Léopold
 *                Vonèche Xavier
 */

#include "H_PrimBD.h"
#include "H_Types2.h"
/*****************/
/*  CONVERSIONS  */
/*****************/


/**********************************/
/*  CONVERSIONS DE TYPES SIMPLES  */
/**********************************/

/**************************/
boolean conventbool(boolint)
/**************************/

booleanint boolint;

{
 if (boolint==0)
    return(False);
    else return(True);
}

/**************************/
booleanint convboolent(bool)
/**************************/

boolean bool;

{
 if (bool==True)
    return(1);
    else return(0);
}

/****************************************/
/*  CONVERSIONS INTERNE VERS PROGRAMME  */
/****************************************/

/*********************************************/
void convintprgtheme(art_theme_temp,art_theme)
/*********************************************/

tthemeint *art_theme_temp;
ttheme *art_theme;

{
 strcpy(art_theme->nom,art_theme_temp->nom);
 strcpy(art_theme->son_nom,art_theme_temp->son_nom);
 art_theme_temp->article=art_theme_temp->article;
 art_theme->priorite=art_theme_temp->priorite;
 art_theme->couleur=art_theme_temp->couleur;
 art_theme->no_theme=art_theme_temp->no_theme;
```

```
 return;
}

/***********************************/
void convintprgmenu(art_temp,art_menu)
/***********************************/

tmenuint *art_temp;
tmenu *art_menu;

{
 art_menu->nnbr_pers=art_temp->nnbr_pers;
 art_menu->nbr_pers=art_temp->nbr_pers;
 art_menu->modifie=conventbool(art_temp->modifie);
 art_menu->imprime=conventbool(art_temp->imprime);
 return;
}

/***********************************/
void convintprgplat(art_temp,art_plat)
/***********************************/

tplatint *art_temp;
tplat *art_plat;

{
 art_plat->no_plat=art_temp->no_plat;
 strcpy(art_plat->nom,art_temp->nom);
 strcpy(art_plat->son_nom,art_temp->son_nom);
 art_plat->article=art_temp->article;
 art_plat->cout=art_temp->cout;
 return;
}

/*************************************************/
void convintprgconstitution(art_temp,art_constitution)
/*************************************************/

tconstitutionint *art_temp;
tconstitution *art_constitution;

{
 art_constitution->modifie_plat=art_temp->modifie_plat;
 return;
}

/*************************************/
void convintprgrepas(art_temp,art_repas)
/*************************************/

trepasint *art_temp;
trepas *art_repas;

{
 art_repas->no_repas=art_temp->no_repas;
 strcpy(art_repas->nom,art_temp->nom);
 strcpy(art_repas->son_nom,art_temp->son_nom);
 art_repas->article=art_temp->article;
 art_repas->priorite=art_temp->priorite;
 return;
```

```
}

/************************************************/
void convintprgpreparation(art_temp,art_preparation)
/************************************************/

tptr_menu_const *art_temp;
tpreparation *art_preparation;

{
 art_preparation->date=art_temp->date;
 return;
}

/*****************************************/
void convintprgproduit(art_temp,art_produit)
/*****************************************/

tproduitint *art_temp;
tproduitcompos *art_produit;

{
 copier_article((char *)art_produit,(char *)art_temp,sizeof(tproduitint));
 return;
}

/*****************************************/
/*  CONVERSIONS PROGRAMME VERS INTERNE  */
/*****************************************/

/************************************************/
void convprgintpreparation(art_preparation,art_temp)
/************************************************/

tpreparation *art_preparation;
tpreparationint *art_temp;

{
 return;
}

/***********************************/
void convprgintmenu(art_menu,art_temp)
/***********************************/

tmenu *art_menu;
tmenuint *art_temp;

{
 art_temp->nnbr_pers=art_menu->nnbr_pers;
 art_temp->nbr_pers=art_menu->nbr_pers;
 art_temp->modifie=convboolent(art_menu->modifie);
 art_temp->imprime=convboolent(art_menu->imprime);
 return;
}

/*********************************************************************/
void convprgintconstitution(art_constitution,theme,plat,ptr_suivant,
                            art_temp)
/*********************************************************************/
```

```
tconstitution *art_constitution;
short theme,plat,ptr_suivant;
tconstitutionint *art_temp;

{
 art_temp->no_theme=theme;
 art_temp->no_plat=plat;
 art_temp->modifie_plat=art_constitution->modifie_plat;
 art_temp->ptr_suivant=ptr_suivant;
 return;
}
```