Adrien Mahieux - Thomas Canaple
Groupe D

EFREI
ECOLE D'INGENIEURS DES TECHNOLOGIES DE L'INFORMATION ET DU MANAGEMENT

Projet Programmation en C.

L'informatique et les applications à la gestion

Adrien Mahieux - Thomas Canaple
Groupe D

Sommaire

Adrien Mahieux - Thomas Canaple
Groupe D

1 - Introduction

L'informatique trouve des applications dans toutes les entreprises des trois secteurs primaire, secondaire et tertiaire. En effet, elle se révèle un outil indispensable à la gestion des ressources, monétaires en particulier. L'utilisation de l'outil infirmatique permet d'obtenir des résultats précis en un temps minimum.

La réalisation de ce projet comporte trois axes majeurs. Il s'agit d'une part de programmer une interface permettant le calcul des mensualités à payer dans le cas d'un emprunt immobilier. D'autre part, nous devront mettre en place un système de calcul de point mort pour une entreprise. Enfin, il s'agit d'effectuer des estimations boursières. Ces trois aspects forment un aperçu des besoins de l'entreprise en terme de programme informatique.

Dans un premier temps, nous verrons la structure de la fonction principale. Cette étude nous aménera à traiter trois parties distinctes. D'une part, l'étude du couple de fichiers valeurs.c et valeurs.h ; d'autre part, l'étude des fichiers point_mort.c et point_mort.h ; puis enfin des fichiers estimations et résolution.

2 – La fonction principale

2.1 – Le fichier main.h

```
#ifndef MAIN
#define MAIN
void titre();
#endif
```

Quatres lignes composent notre fichier main.h. Les deux premières lignes posent une condition aux préprocésseur, qui ne prendra en compte ce fichier que si celui ci n'est pas déjà inclu. Cette sécurité évite les erreurs de compilations. La troisème ligne comporte le prototype de la fonction titre(). Elle ne prend aucun argument, et ne renvoie aucune valeur de retour. En effet, elle est de type void. Cela s'explique par le fait qu'il s'agit d'une fonction purement d'affichage. Enfin, la dernière ligne achève la définition de ce fichier d'entête.

Quatres lignes semblent peu pour un programme qui traite de trois parties complètes ; mais nous comprendront dans la suite pourquoi ce choix a été réalisé.

2.2 – Le fichier main.c

Le fichier main.c fait appelle à deux fichiers d'entête en particulier :

```
#include <stdlib.h>
#include <stdio.h>
```

Ces fichiers sont des fichiers d'entête standards gérant les entrées/sorties du programme, et permettant l'utilisation des fonctions usuelles telles que printf().

On retrouve d'autres fichiers d'entête apellés :

```
#include "main.h"
#include "valeurs.h"
#include "point_mort.h"
#include "estimations.h"
#include "resolution.h"
```

Ce sont les fichiers header créés pour les besoins du programme.
Interressons nous maitenant à la structure de la fonction principale. On remarque que seule quatres variables sont nécessaires à la réalisation du programme. De plus, la fonction principale ne comporte qu'un switch, permettant de traiter les différents cas saisis par l'utilisateur. Pour chaque cas, seules une à deux fonctions sont apellées. On note que la fonction principale n'est utilisée ici que pour structurer le programme, et réaliser le premier choix de l'utilisateur. On utilisra pour le traitement de ces choix des fonctions scindées en trois fichiers .c qui composent les trois parties du programme. Ils 'agit des fichiers valeurs.c, point_mort.c et estimations.c.

On remarque l'utisation de la fonction system() comme moyen de modifier l'interface du programme :

```
system("title L'informatique et les applications a la gestion");
system("mode con: cols=150 lines=60");
```

Le premier appel de la fonction system() permet de donner un titre à la fenetre du programme. Le second appel permet de modifier la taille en terme de lignes et de colonnes de la console. Ces deux aspects apportent un plus pour le confort de l'utilisateur.

3 – Première partie : autour d'un emprunt immobilier

3.1 – Le fichier valeur.h

Nous avons vu que les différentes parties du programme étaient traitées indépendement de la fonction principale. Dans cette première partie, interréssons nous aux prototypes des fonctions que nous allons utiliser dans la suite du programme.

D'une part, la saisie puis le traitement de nombreuses informations de types différents est facilité par l'utilisation d'une structure qui regroupe tout ces critères dans une seule et même variable. Ce traitement permet l'utilisation d'une fonction de saisie qui renverra la valeur ensuite à une fonction de traitement. Voici la structure en question :

```
typedef struct Res
{
   float montant_emprunt;
   float valeur_bien;
   float frais_notaire;
   float frais_agence;
   float taux_base;
   float taux_assurance;
   float taux_global;
   float montant_garantie;
   float mensualite;
   float endettement;
   int etat;
}Res;
```

D'autre part, pour une meilleur appréhension du déroulement du programme par l'utilisateur, nous utilisons un nouveau titre à chaque nouvelle partie. D'où l'utilisation des quatres fonctions d'affichage suivantes :

```
void titre_Pun();
void titre_echeance();
void titre_amortissement();
void titre_sim();
```

Nous retrouvons ensuite les fonction de triatements nécéssaires au programme. Il s'agit des deux fonctions nécessaires à la comparaison de deux types d'emprunt différents : à amortissment constant et à échéance constante. D'une fonction d'affichage de tableau, qui contient les résultats de ces deux fonctions. Puis une fonction de traitement et une fonction d'affichage nécessaires à la seconde partie de cette première partie du programme.

3.2 – Le fichier valeurs.c

Afin de traiter les données mathématiquement, ce fichier fait appel au header math.h.
D'autre part, nou utilisons des valeurs constantes qui sont celles données dans le sujet du projet. Ainsi, à la compilation, chacune de ces constantes est automatiquement remplacée par sa valeur correspondante.

```
#define EMPRUNT 160000
#define DUREE_MOIS 180
#define DUREE_ANNEE 15
#define TAUX 0.042
```

Le calcul des valeurs demandées en ce qui concerne de prêt à échéance constante est assez simple puisque'il suffit de remplir le tableau par une boucle for, en utilisant la formule de ce prêt :

```
for( i=1 ; i<=DUREE_MOIS ; i++ ){
	tab[i-1] = ( EMPRUNT * taux_periodique ) * ( 1 - pow(1 + taux_periodique, -
DUREE_MOIS) );
}
```

Le calcul des valeurs du prêt à amortissement constant est quant à lui plus délicat. Il faut tenir compte de l'amortissment, ainsi que du jour où le remboursement doit avoir lieu. En effet, selon la durée en mois, la valeur du mois de l'année actuelle va changer.

```
for( i=1 ; i<=DUREE_MOIS ; i++ ){
	 if( i==1 ){
		j=1;
	}
	if( i%12 == 1 ){
		j = i/12;
	}
	interets = TAUX * ( EMPRUNT * ( DUREE_ANNEE - j + 1) ) /
DUREE_ANNEE;
	tab[i-1] =  ( am + interets ) / 12;
}
```

L'affichage du tableau contenant les résultats résulte d'une simple boucle for{}. La technique consiste à sauter des lignes entre chaque année :

```
if( i%12 == 0 ){
	printf("\n");
}
```

La fonction simu_pret_immo() consiste à demander à l'utilisateur les données nécéssaires au traitement par la fonction précédente. Les valeurs des données sont quelques fois dépendantes d'un switch{} ou d'une condition if{}. La fonction d'affichage affiche_res_simu_immo() met en page les résultats de l'opération précédente et les affiche à l'écran.

4 – Deuxième partie : calcul d'un point mort

4.1 - Le fichier point_mort.h

Dans cette partie, il s'agit de calculer le point mort de la valeur d'une marchandise. En d'autre terme, il s'agit de la valeur pour laquelle l'entrprise ne fera ni de bénéfices, ni de perte sur les opération nécéssaires à la mise en vente de cette marchandise.

Nous prendront en compte cinq critères de vente, soit cinq fonctions de traitement suivant le critère choisi par l'utilisateur.

Nous voyons ici que toutes ces fonctions sont de types void car entièrement indépendantes des unes des autres.

```
void titre_Pdeux();
void titre_simulation();
void titre_1sim();
void titre_2sim();
void titre_3sim();
void titre_4sim();
void titre_5sim();

void simulation();
void sim1();
void sim2();
void sim3();
void sim4();
void sim5();
```

4.2 – Le fichier point_mort.c

Nous fixons tout d'abord les valeurs que le rpojet nous impose de fixer. Ces variables constantes seront urilisés dans toutes les fonctions du fichier point_mort.c et seront donc définis par un define pour une optimisation maximale.

```
#define FIXE_LOYER 750
#define FIXE_FAI 30
#define FIXE_SALAIRE 1255
```

Ces couts fixes, que nous espèrons assez proches des couts réels, sont représentatifs des dépensent fixes des entreprises.

Les cinq fonctions réalisées se présentent de la même manière : il s'agit de saisir les données des valeurs qui doivent être fixées par l'utilisateur. Le coût de la cinquièle valeur est calculée de façon à ce que l'entreprise ne fasse ni bénéfices, ni pertes.

Dans l'exemple de la matière première comme valeur à déterminer, cela donne :

```
couts_matieres_premieres = - ( quantite_vendue * prix_vente ) + couts_cornets +
couts_transports;
```

Il s'agit ensuite de faire varier la valeur de couts_matiers_permieres de 10% das le positif et dans le négatif pour obtenir une autre valeur du point mort. On obtient alors des pertes ou des bénéfices.

5 – Troisième partie : estimations boursières

5.1 – Les fichiers estimations.h / estimations.c

Dans cette partie, nous créons, via de nombreux calculs mathématiques, une estimation des valeur boursières à partir des valeurs entrées.. A cause des formules de statistiques relativement complexes, il y a plusieurs fonctions de calcul. Nous avons donc cinq fonctions qui réalisent des sommes différentes :

```
float somme1(float tab[], int tai);
float somme2(float tab[], int tai);
float somme3(float tab[], int tai);
float sommeyprimemoinsybaraucarre(float coeff[],float tab[], int tai);
float sommeymoinsybaraucarre(float coeff[],float tab[], int tai);
```

Et cinq fonctions qui réalisent des moyennes de valeurs agencées différements :

```
float moyenney(float tab[], int tai);
float moyennex(int tai);
float moyxybar(float tab[], int tai);
float moyxbarybar(float tab[], int tai);
float moyxcarre(int tai);
```

Nous avons choisi des noms à ralonge pour ne pas se perdre dans l'usage des fonctions de moyenne.

Les fonctions moyenne et somme utilisent des boucles for, parfois des tableau dynamiques.
La fonction modele_linaire() demande à l'utilisateur de selectionner le mode désiré en indiquant le nombre correspondant (1 ou 2) . Cette fonction effectue la selection via un switch et lance ainsi le calcul du cas linéaire ou celui du second degrès.

- Cas du premier degré :
  - Premier cas : Tout d'abord, il s'agit de calculer l'équation de la droite correspodante à l'aide des fonctions.
    Puis, il faudra calculer l'estimation résultante grâce a une simple équation du premier degré.

Seocnd cas custom :

pareil mais il faut saisir les données à traiter. Pis calculer R pour voir si le modèle linéaire correspond bien.

- Cas du second degré :
  Le cas du second degré est géré par la fonction modele_second_degres. Il n'y a qu'une seule sous-section a cette fonction qui affiche la valeur calculée.
  Dans cette fonction, l'utilisateur est invité à entrer les valeurs du cours à traiter. Avec ces

valeurs, la fonction resoud un système du second degré pour trouver l'équation de la courbe. Enfin, il calcule R pour voir si le modèle est acceptable. Pour être considéré comme valide, R doit être supérieur a 0,87 :

```
if( r >= 0.87 ){
    printf("  Le modele du second degres est acceptable dans ce cas\n");
}
else{
    printf("Le modele de second degré n'est pas acceptable dans ce cas\n\n ");
    system("pause");
    return 0;
}
```

Lorsque les données sont entrées, l'utilisateur doit indiquer a quel jour il souhaite avoir l'estimation. Pour donner le resultat, le programme va utiliser l'équation de la courbe :

```
y = coeff[0] * t * t + coeff[1] * t + coeff[2];
```

5.2 – Le couple resolution.h / resolution.c

Comme vu dans la partie précédente, nous avons de nombreux calculs principalement statistiques pour determiner certains résultats. Au vu des calculs complexes que nécessitent la résolution d'un système de trois équations à trois inconnues et afin de garder un ensemble de fichiers clairs dont le nom correspond bien au sujet qu'ils traitent, nous allons créer un autre fichier .c / .h

Dans notre cas, nous aurons besoin de faire differentes sommes. Nous inclurons donc comme toujours les prototypes des fonctions dans le fichier .h

Comme précédemment, nous avons utilisé des noms de fonctions longs. Ce choix peut sembler lourd lors de la création du programme, mais la comprehension générale du programme en est grandement facilitée.

Dans le fichier source, nous allons donner un nom à toutes les variables suivant le schéma :

ax + by +cz = d
ex + fy + gz = h
ix + jy + kz = l

avec x, y , z inconnues

```
a = sommexcarre(tai);
b = sommex(tai);
c = tai;
d = sommey(tab, tai);
e = sommexcube(tai);
f = sommexcarre(tai);
g = sommex(tai);
h = sommexy(tab, tai);
i = sommexquatre(tai);
j = sommexcube(tai);
k = sommexcarre(tai);
l = sommexcarrey(tab, tai);
```

Puis on calcule le Delta :

```
delta = a*f*k + b*g*i + c*e*j - c*f*i - a*g*j – b*e*k;
```

Et un simple test termine le calcul :

```
if( delta != 0 ){
    coeff[0] = ( d*f*k + b*g*l + c*h*j - c*f*l - d*g*j - b*h*k ) / delta;
    coeff[1] = ( a*h*k + d*g*i + c*e*l - c*h*i - a*g*l - d*e*k ) / delta;
    coeff[2] = ( a*f*l + b*h*i + d*e*j - d*f*i - a*h*j - b*e*l ) / delta;
}
else{
     printf("  Le systeme est impossible\n");
```

```
}
```

6 – conclusion

En conclusion, ce projet nous a montré que l'informatique était un outil ayant un champ d'action très vaste et qu'il peut s'appliquer aussi bien à la finance qu'au multiples besoins d'une entreprise en utilisant les mathématiques.

Ce genre de programme est également utilisé en bourse, dans les banques... On en trouve même disponibles sur internet pour calculer les taux de remboursement de diverses banques.

Les divers calculs mathématiques et statistiques sont ainsi le coeur de l'application. Ici comme dans toute realisation de programme, il est impératif de connaître les tenants et les aboutissants pour trouver la facon la plus rapide et pratique d'obtenir le resultat escompté.