Algorithmes et programmation en Pascal

Faculté des Sciences de Luminy — Edouard Thiel

TP

Deug 1 Mass MA
Module de 75 heures
1997 à 2004

Table des matières

1. Calcul d'impôts

On cherche à faire un programme qui calcule l'impôt en fonction des revenus de 2001 convertis en euros (pour des situations simples).

Le programme va poser un certain nombre de questions à l'utilisateur (le salaire, le nombre de parts, etc) et affichera au fur et à mesure les valeurs calculées.

1.1 Votre revenu brut global

- On lit le salaire `s1`.
- Déduction de 10% ou frais réels :
  Soit `d1` 10% de `s1`. On affiche `d1` et on demande si on garde ce montant ou si on le remplace par des frais réels plus élevés.
  Attention `d1` est plafonné à 12229 et vaut au minimum 364.
- Abattement de 20% :
  Soit `v1` 20% de `s1-d1`. Attention `v1` est limité à 22380.
- Le revenu brut global `rbg1` est `s1-d1-v1`.

1.2 Revenu brut global du conjoint

- On demande si il y a un conjoint ou pas (booléen `cj`).
- Si c'est le cas, on refait tout le 1. pour un jeu de variables `s2`, `d2`, `v2` et `rbg2`.

1.3 Déduction de charges

- On lit le montant `pa` de pensions alimentaires et autres charges (limité à 3824 par enfant).
- Le revenu net imposable `rni` est `rbg1+rbg2-pa` ou `rbg1-pa` si il n'y a pas de conjoint.

1.4 Le quotient familial

- On lit le nombre de parts `np` (1 pour célibataire sans enfant, 2 pour marié sans enfant, 2.5 pour marié avec un enfant, etc).
- Le quotient familial `qf` est `rni/np`.

1.5 Calcul de l'impôt

- L'impôt `ip` est établi dans le tableau suivant.

```
si qf ≤ 4121             ip := 0
si qf ∈ ] 4121.. 8104]   ip := (rni*0.075 ) - (  309.08*np)
si qf ∈ ] 8104..14264]   ip := (rni*0.21  ) - ( 1403.12*np)
si qf ∈ ]14264..23096]   ip := (rni*0.31  ) - ( 2829.52*np)
si qf ∈ ]23096..37579]   ip := (rni*0.41  ) - ( 5139.12*np)
si qf ∈ ]37579..46343]   ip := (rni*0.4675) - ( 7299.91*np)
si qf > 46343            ip := (rni*0.5275) - (10080.49*np)
```

• Décote : Si `ip` est inférieur à 760, l'impôt sera `ip-(380-ip/2)`. Si après cela `ip` est inférieur à 61, vous n'avez pas d'impôt à acquitter.

1.6 Résultat

Afficher l'impôt, et le pourcentage que cela représente par rapport à `s1+s2`, et le nombre de mois de salaires que cela représente.

1.7 Exemple

```
∗ ∗ ∗ Calcul de l'impôt 2001 ∗ ∗ ∗

Votre salaire : 13650
La déduction de 10% est 1365.00
Est-il plus intéressant de déclarer des frais réels (O/N) ? O
Frais réels : 1960
L'abattement de 20% est 2338.00
Votre revenu brut global est 9352.00

Avez-vous un conjoint (O/N) ? O
Salaire du conjoint : 17532
La déduction de 10% est 1753.20
Est-il plus intéressant de déclarer des frais réels (O/N) ? N
L'abattement de 20% est 3155.76
Le revenu brut global du conjoint est 12623.04

Pensions alimentaires et autres charges : 3824
Le revenu net imposable arrondi est 18151
Nombre de parts : 2.5
Le quotient familial est 7260.40
La tranche d'imposition est 4121..8104
L'impôt avant décote est 588.62

Aprés décote et abattement, l'impôt est 502.93 EUR
Cela représente 1.61% du salaire, soit 0.19 mois.
```

2. Prêt étudiant

On cherche à faire un programme qui simule un prêt étudiant à remboursement différé, et permet de calculer le nombre ou le montant des mensualités, ainsi que le coût total du crédit.

Le programme va poser un certain nombre de questions à l'utilisateur (la somme empruntée, le taux du crédit, etc) et ensuite affichera mois après mois ce qui reste à rembourser (on obtient un *tableau d'amortissement*).

2.1 Déroulement du programme

On demande la somme empruntée `se`, le taux annuel de crédit `ta` en %, puis le nombre de mois `m1` au début du crédit où l'emprunteur ne rembourse rien.

On propose ensuite 2 options : calculer la durée du remboursement pour une mensualité donnée (choix A), ou calculer la mensualité pour rembourser le prêt en un temps donné (choix B).

2.2 Durée pour une mensualité donnée

On demande le montant d'une mensualité `sm` (payable à partir du mois `m1+1`). Regardons comment évolue la dette avant et pendant les remboursements.

• Admettons que l'emprunteur désire ne rien rembourser les 3 premières années (il rentre `m1: 36`). La somme due `sd` par l'emprunteur au départ du prêt est égale à la somme empruntée, soit `se`.

Au bout d'un an, la dette est `se * (1 + ta/100)`. Si on écrit `fa := 1 + ta/100`, on peut encore dire que la dette `sd` au bout d'un an est `se*fa`. Au bout de 2 ans, `sd` est `se*fa*fa`; au bout de 3 ans, `sd` est `se*fa*fa*fa`.

Maintenant si on veut connaître la dette mois après mois, il suffit de prendre le facteur mensuel `fm` égal à la racine $12^{ème}$ du facteur annuel `fa`. On le calcule en écrivant `fm := exp(ln(fa)/12)`.

Au bout de 1 mois, `sd` est `se*fm`; au bout de 2 mois, `sd` est `se*fm*fm`; ... (au bout de 12 mois, `sd` est `se*fm*fm*`$\cdots$`*fm` ce qui est la même chose que `se*fa` puisque $\texttt{fm}^{12} = \texttt{fa}$).

Comme on le voit, il est facile de calculer mois après mois l'augmentation de la dette dans la première période du prêt où l'on ne rembourse encore rien.

• Nous entrons dans la période de remboursement : au mois `m1` l'emprunteur a une dette `sd` que nous venons de calculer.

Au mois `m1+1`, il y a un mois d'intérêts en plus, et on rembourse la première mensualité `sm`; donc la dette `sd` devient `sd*fm - sm`.

Chaque mois on fait comme cela, on ajoute les interêts du mois et on retranche la mensualité : on voit ainsi décroître `sd`.

Le prêt est entièrement remboursé quand `sd` est nul. Attention la dernière mensualité n'est pas forcément `sm` mais peut être inférieure.

Pour connaître la durée du prêt, il suffit de compter le nombre de mois `m2` pendant lesquels on verse des mensualités. Le coût total du crédit est ce qu'on a payé en totalité ; on l'affiche, ainsi que le pourcentage que cela représente par rapport à la somme empruntée.

2.3 Mensualités pour une durée donnée

Dans cette partie on demande la durée de remboursement `Bm2`, puis le programme cherche la bonne valeur de la mensualité `Bsm` pour que le prêt se fasse bien dans le temps `Bm2`.

Pour cela, le programme fait une série d'essais de différentes mensualités `sm`, fait à chaque fois tous les calculs du §2.2 et regarde si le nombre de mensualités obtenu `m2` est le nombre attendu `Bm2`.

Le choix de la stratégie est ouvert : recherche exhaustive, recherche dichotomique, etc. On affiche les tentatives au fur et à mesure ; puis lorsque la solution est obtenue, on affiche tous les résultats utiles tels que le coût du crédit et le % comme au §2.2.

2.4 Exemple

∗ ∗ ∗ Prêt étudiant ∗ ∗ ∗

Somme empruntée : **6000**
Taux annuel en % : **8.75**
Nb de mois différé : **20**
A : Chercher durée – B : Chercher mensualités – Votre choix : **A**
Mensualité : **160**

Mois	Payé	Dette	Coût total
0	0.00	6000.00	0.00
1	0.00	6042.09	0.00
2	0.00	6084.47	0.00
...			
20	0.00	6900.28	0.00
21	160.00	6788.68	160.00
22	160.00	6676.30	320.00
...			
71	160.00	86.09	8160.00
72	86.09	0.00	8246.09

6000.00 EUR empruntés à 8.75 %, différé de 20 mois, mensualités 160 EUR
Nombre de mensualités : 52. Coût total du crédit : 8246.09 EUR soit 137 %.

3. Le jeu des mille euros

On cherche à réaliser un jeu des mille euros. L'ordinateur choisit un prix secret entre 1 et 1000 et le joueur doit le deviner en un nombre minimum de coups.

À chaque essai du joueur, l'ordinateur dit si c'est plus cher, si c'est moins cher ou si c'est gagné. On prévoit aussi le jeu inverse : l'utilisateur choisit un prix secret, et l'ordinateur essaie de le deviner. C'est l'utilisateur qui lui dit si c'est plus, ou moins, ou gagné. On fera enfin jouer l'ordinateur contre lui-même : il choisit n'importe quel prix secret et essaie de le deviner !

Le problème est décomposé en plusieurs procédures et fonctions, que l'on demande de **paramétrer** complètement.

3.1 Programme principal

Déclarer la constante nommée `MILLE`. Faire un programme principal, qui affiche un message de bienvenue puis propose un menu :

A	Vous jouez contre l'ordinateur	**C**	Le programme se teste
B	L'ordinateur joue contre vous	**D**	Quitter

En fonction du choix de l'utilisateur : soit le programme se termine (choix D), soit il affiche un message d'erreur (mauvais choix), soit le programme va exécuter l'une des procédure `vous_jouez`, `ordi_joue` ou `teste_prog`, puis réafficher le menu et attendre un nouveau choix.

3.2 Vous jouez contre l'ordinateur

• Écrire une fonction `tirage_secret` qui renvoie un entier entre 1 et `MILLE`. On utilisera la fonction `random(x)` qui renvoie un entier au hasard entre `0` et `x-1`. (Il faut appeler une fois `randomize;` au début du programme principal pour initialiser le générateur de nombre aléatoires ; sinon `random(x)` renvoie tout le temps `0`).

• Écrire une fonction `compare_prix` qui reçoit en paramètres les entiers `secret` et `essai`, et renvoie -1, 0 ou 1 selon que le prix à deviner est inférieur, égal ou supérieur à l'essai.

• Écrire la procédure `vous_jouez`, qui fait un tirage secret, puis demande au joueur des tentatives numérotées. À chaque essai, la procédure affiche si c'est plus, si c'est moins, si ce n'est pas dans l'intervalle de 1 à `MILLE`, ou si c'est gagné. La procédure se termine lorsque le joueur a trouvé le prix.

3.3 L'ordinateur joue contre vous

• Écrire une fonction `faire_essai` qui reçoit en paramètres les bornes `min` et `max` d'un intervalle, et renvoie un entier dans cet intervalle. On peut ne pas choisir cet entier n'importe comment ...

• Écrire une procédure `reborne` qui reçoit en paramètres les bornes `min` et `max`, la valeur `essai` et un entier `compar` qui vaut -1 ou $+1$ selon que le prix à deviner est inférieur ou supérieur à l'essai. Cette procédure va se servir de ces données `essai` et `compar` pour changer l'une des bornes `min` ou `max`.

• Écrire la procédure `ordi_joue`, qui va essayer de deviner le prix que l'utilisateur a secrètement choisi.

La procédure affiche une série d'essais numérotés. Pour chaque essai affiché, l'utilisateur lui répond l'un des caractères `'+'`, `'-'` ou `'='`. La procédure lit ce caractère puis affiche un message d'erreur si le caractère lu n'est pas l'un de ceux attendus, ou bien se termine si le caractère est `'='`, ou bien fait un nouvel essai d'après la réponse.

Selon que c'est `'+'` ou `'-'`, on mettra à jour l'intervalle entre `inf` et `sup` avec `reborne`, puis on fera un nouvel essai avec `faire_essai`.

3.4 Le programme se teste

Si les procédures et fonctions du §3.3 sont bien écrites, elles doivent trouver le bon prix rapidement et ceci pour n'importe quel prix. Le seul moyen d'en être sûr, est que le programme se teste lui-même !

Écrire la procédure `teste_prog`, qui pour chaque prix secret de 1 à `MILLE`, essaie de deviner le prix avec `faire_essai` et `reborne`, et se répond lui-même avec `compare_prix`. La procédure affiche pour chaque prix secret le nombre de coups pour le deviner, et à la fin affiche la moyenne des nombres de coups.

3.5 Exemple

```
*** Le jeu des mille euros ***
   A : Vous jouez contre l'ordinateur
   B : L'ordinateur joue contre vous
   C : Le programme se teste
   D : Quitter
Votre choix : A
Essai 1 : 450
                c'est PLUS
Essai 2 : 720
                c'est MOINS
Essai 3 : 670
                c'est PLUS
Essai 4 : 695
                GAGNE en 4 coups!!
   A : Vous jouez contre l'ordinateur
   B : L'ordinateur joue contre vous
   C : Le programme se teste
   D : Quitter
Votre choix : B
Essai 1 : 500 (+ - =) ? -
Essai 2 : 250 (+ - =) ? +
Essai 3 : 375 (+ - =) ? +
Essai 4 : 437 (+ - =) ? -
Essai 5 : 406 (+ - =) ? =
TROUVE en 5 coups!!
   A : Vous jouez contre l'ordinateur
   B : L'ordinateur joue contre vous
   C : Le programme se teste
   D : Quitter
Votre choix : D
```

4. Mastermind

On cherche à réaliser un jeu de Mastermind. L'ordinateur choisit une combinaison secrète (5 chiffres de 1 à 8) et le joueur doit la deviner en un nombre minimum de coups.

Le problème est décomposé en plusieurs procédures et fonctions, que l'on demande de paramétrer complètement.

Dans le programme principal on utilise 2 tableaux de 5 entiers `secret` et `essai`.

4.1 Tirage de la combinaison secrète

Faire une procédure `tirage_secret` qui fait un tirage aléatoire et le stocke dans le tableau `s`. On utilisera la fonction `random(x)` qui renvoie un entier au hasard entre `0` et `x-1`. (Il faut appeler une fois `randomize;` au début du programme principal, pour initialiser le générateur de nombre aléatoires ; sinon `random(x)` renvoie tout le temps `0`).

Pour la mise au point de la procédure et du passage de paramètres : faire un programme principal dans lequel on appelle la procédure puis on affiche la combinaison secrète. Exécuter plusieurs fois. ⟶ Cela s'appelle tracer la procédure.

4.2 Lecture d'une combinaison

Faire une procédure `lit_combi` qui lit une combinaison au clavier, et la place dans le tableau `e`.

Si la combinaison est incorrecte (ses chiffres ne sont pas entre 1 et 8), la procédure affiche un message d'erreur et recommence à lire une nouvelle combinaison.

Si la combinaison est `00000`, la lecture est abandonnée, et la procédure signale au programme (par un booléen) que le joueur désire connaître la solution et arrêter la partie.

4.3 Comparaison des combinaisons

Pour chaque tentative, l'ordinateur donne le gain réalisé :

`gain` = 10 ∗ <*nombre de chiffres trouvés bien placés*>
+ <*nombre de chiffres trouvés mal placés*>

Exemple

- À trouver `15472`
- Joue `27466`
- Le gain de `27466` est 12 (1 bien placé, 2 mal placés).

⟶ Le joueur a gagné lorsque le gain est 50.

Faire une fonction `calcule_gain` qui compare les tableaux `s` et `e` et renvoie le gain. Tracer abondamment pour la mise au point. Attention, un chiffre compté bien placé ne doit pas être compté mal placé ...

4.4 Programme principal

Faire un programme qui appelle les procédures ou fonctions `tirage_secret`, `lit_combi` et `calcule_gain`, avec les tableaux `secret` et `essai`.

Après le tirage secret, le programme lit la tentative du joueur, la compare à la combinaison secrète et recommence à lire, ceci aussi longtemps que le joueur n'a pas gagné ou n'a pas abandonné.

4.5 Exemple

∗ ∗ ∗ Mastermind ∗ ∗ ∗

Tapez 5 chiffres accolés de 1 à 8, ou 00000 pour quitter.

Essai 1 ? **12345**
score = 11
Essai 2 ? **13355**
score = 01
Essai 3 ? **12141**
score = 11
Essai 4 ? **31323**
score = 10
Essai 5 ? **61748**
score = 22
Essai 6 ? **77888**
score = 10
Essai 7 ? **71644**
score = 50

∗ ∗ ∗ GAGNÉ en 7 coups ∗ ∗ ∗

La combinaison secrète était 71644.