TP
N° 3

LST–A (S3)
2009-2010
Algorithmes et Programmation Impérative 2

28 septembre 2009

Matériel fourni : Vous pourrez réaliser chacun des programmes demandés dans la suite à partir du programme `squelette.pas` que vous pouvez télécharger depuis le portail. Ce squelette contient une amorce de programme qui nécessite que sur la ligne de commande d'exécution soit passé un (et un seul) paramètre représentant un nombre entier (la somme à régler dans le problème qui suit). Si le paramètre est absent (ou s'il y en a plusieurs) un appel à la procédure `usage` est lancé, qui affiche un message d'erreur et interrompt l'exécution du programme.

1 Le problème

Exercice 1. Le problème que vous allez résoudre dans ce TP est celui de la conception d'un programme qui compte le nombre de façons de faire l'appoint pour régler une certaine somme avec des pièces de 1, 2, 5, 10, 20, 50, 100 et 200 centimes.

Faire l'appoint pour régler une somme d'une valeur de n centimes, avec les pièces contenues dans l'ensemble

$$PIECES = \{1, 2, 5, 10, 20, 50, 100\},$$

c'est déterminer pour chaque pièce $p \in PIECES$ un entier n_p tel que

$$n = \sum_{p \in PIECES} n_p p.$$

Par exemple, pour régler une somme de $n = 100$ centimes, on peut le faire avec une pièce de 100 centimes et l'appoint est donné par les entiers

n_1	n_2	n_5	n_{10}	n_{20}	n_{50}	n_{100}	n_{200}
0	0	0	0	0	0	1	0

mais on peut le faire avec cent pièces de 1 centime et l'appoint est alors décrit par

n_1	n_2	n_5	n_{10}	n_{20}	n_{50}	n_{100}	n_{200}
100	0	0	0	0	0	0	0

ou bien encore par de très nombreuses autres façons comme par exemple celle décrite par

n_1	n_2	n_5	n_{10}	n_{20}	n_{50}	n_{100}	n_{200}
3	6	5	1	0	1	0	0

.

L'objectif est donc de concevoir un programme capable de calculer le nombre $c(n)$ de façons de faire l'appoint pour une somme n quelconque. Puis de lister toutes ces façons.

Question 1.1. *Vérifiez "à la main" ces premières valeurs de c :*

n	0	1	2	3	4	5	6	7	8	9	10
$c(n)$	1	1	2	2	3	4	5	6	7	8	11

2 Un cas simple avec deux pièces seulement

Exercice 2. Dans cette partie, vous allez considérer une version (très) simple de ce problème, limité au cas où il n'y a que deux valeurs de pièces : 1 et 2 centimes.

$$PIECES = \{1, 2\}$$

Question 2.1. *Justifiez que dans ce cas, on a*

$$c(0) = 1 \quad (1)$$
$$c(1) = 1, \quad (2)$$

et pour tout entier $n \geq 2$

$$c(n) = c(n-2) + 1. \quad (3)$$

Question 2.2. *Réalisez alors dans un programme nommé* `appoint1.pas` *une fonction* `nbAppoint` *qui calcule* $c(n)$ *en fonction de* n.

Question 2.3. *Calculez* $c(n)$ *pour* n *compris entre 0 et 10, et vérifiez.*

Question 2.4. *Exprimez* $c(n)$ *en fonction de* n.

3 On complique un peu avec trois pieces

Exercice 3. On complique un peu le problème avec trois pièces de valeurs : 1, 2 et 3 centimes[1].

$$PIECES = \{1, 2, 3\}$$

Il peut être utile d'introduire une fonction auxiliaire avec un paramètre supplémentaire p qui donne le nombre d'appoints avec des pièces de valeur p au maximum. On désigne par c' cette fonction. Ainsi,

$$c'(n, p) = \text{nbre de façons de faire l'appoint avec des pièces } \leq p,$$

avec p compris entre 1 et 3, et

$$c(n) = c'(n, 3).$$

Question 3.1. *Justifiez que pour toute valeur* p

$$c'(0, p) = 1 \quad (4)$$

et toute somme n

$$c'(n, 1) = 1. \quad (5)$$

Question 3.2. *Justifiez que si* $0 < n < p$

$$c'(n, p) = c'(n, p-1). \quad (6)$$

[1] D'accord ! il n'existe pas de pièces de trois centimes.

Question 3.3. *Justifiez enfin que si* $1 < p \leq n$

$$c'(n,p) \quad = \quad c'(n-p,p) + c'(n,p-1). \tag{7}$$

Question 3.4. *À partir des ces équations, réalisez dans un programme que vous nommerez* `appoint2.pas` *une fonction nommée* `nbAppoint` *qui calcule* $c(n)$.

Question 3.5. *Utilisez votre programme pour calculer* $c(n)$ *pour* n *compris entre 0 et 10, et vérifiez.*

4 Toujours avec trois pièces mais quelconques

Exercice 4. Maintenant vous allez considérer le cas où les trois pièces ne sont pas de valeurs consécutives, mais de valeurs : 1, 2 et 5 centimes.

$$PIECES = \{1, 2, 5\}.$$

Les équations 6 et 7 ne peuvent plus convenir, car elles contiennent toutes les deux l'expression $p - 1$. Or il n'est plus vrai que si p désigne la valeur d'une pièce, $p - 1$ en désigne une aussi.

En revanche, on peut remplacer cette expression $p - 1$ par la valeur d'une fonction en p qui désigne la valeur d'une pièce immédiatement inférieure à p.

Question 4.1. *Dans un programme que vous nommerez* `appoint3.pas`*, réalisez une fonction nommée* `valeurPrecedente` *qui donne la valeur immédiatement inférieure à la valeur passée en paramètre, et adaptez en conséquence votre fonction de calcul du nombre d'appoints.*

Question 4.2. *Avec le programme ainsi réalisé, calculez le nombre d'appoints pour* n *compris entre 1 et 10, et vérifiez.*

5 Le cas général

Exercice 5. Dans cette partie, vous résolvez le problème posé initialement avec

$$PIECES = \{1, 2, 5, 10, 20, 50, 100, 200\}.$$

Vous vous efforcerez de faire en sorte que le programme soit facilement adaptable à d'autres ensembles de pièces, avec la condition que ces ensemles contiennent toujours la valeur 1 de sorte qu'il y ait toujours au moins une façon de régler une somme n quelqu'elle soit ($n_1 = n$ et $n_p = 0$ pour $p \in PIECES \setminus \{1\}$).

5.1 Première méthode

Question 5.1. *Dans un programme que vous nommerez* `appoint4.pas`*, adaptez la fonction* `valeurPrecedente` *à l'ensemble de pièces considéré maintenant.*

Question 5.2. *Avec le programme ainsi réalisé, calculez* $c(n)$ *pour* n *compris entre 1 et 10, et vérifiez.*

5.2 Seconde méthode

La seconde méthode va consister à ne plus utiliser la fonction `valeurPrecedente` mais à représenter l'ensemble des pièces par un tableau comme le montre le listing ci-dessous

```
const
   NB_PIECES = 8; // nbre de pieces disponibles
type
   INDICE = 1..NB_PIECES;
const
   PIECES : array[INDICE] of CARDINAL =
      (1,2,5,10,20,50,100,200); // valeurs des pieces dans l'ordre croissant
```

On va changer la fonction auxiliaire c' utilisée jusqu'à présent de sorte qu'elle fasse référence à la représentation sous forme d'un tableau de l'ensemble $PIECES$. Ainsi la définition de c' sera désormais

$$c'(n,i) = \text{nbre de façons de faire l'appoint avec des pièces dont l'indice ne dépasse pas } i,$$

avec $1 \leq i \leq$ NB_PIECES, et

$$c(n) = c'(n, \texttt{NB\_PIECES})$$

Question 5.3. *Réexprimez les équations 4, 5, 6 et 7 dans ce nouveau contexte.*

Question 5.4. *Réalisez dans un programme nommé* `appoint5.pas` *le calcul de* $c(n)$.

Question 5.5. *Avec le programme ainsi réalisé, calculez* $c(n)$ *pour* n *compris entre 1 et 10, et vérifiez.*

6 Et maintenant, on veut lister toutes les façons

Exercice 6. On ne se contente plus du nombre de façons, mais on souhaite connaître toutes les façons.

Question 6.1. *Dans un programme nommé* `appoint6.pas`*, réalisez une procédure nommée* `listerAppoint` *qui affiche à l'écran toutes les façons de faire l'appoint pour régler une somme donnée en paramètre. Votre programme devra par exemple produire l'affichage*

```
$ ./appoint 10
 10
 5 5
 5 2 2 1
 5 2 1 1 1
 5 1 1 1 1 1
 2 2 2 2 2
 2 2 2 2 1 1
 2 2 2 1 1 1 1
 2 2 1 1 1 1 1 1
 2 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1
```

qui montre les 11 façons de faire l'appoint.

7 Enfin une curiosité mathématique

Exercice 7. Ce qui suit n'est plus un problème de nature informatique.

Considérons la fonction numérique d'une variable réelle

$$f(x) = \prod_{p \in PIECES} \frac{1}{1-x^p} = \frac{1}{(1-x)(1-x^2)(1-x^5)\dots(1-x^{200})}.$$

Cette fonction est une fraction rationnelle définie, continue et dérivable sur l'intervalle $]-1,1[$.

On peut visualiser sa courbe représentative sur cet intervalle en utilisant l'utilitaire `gnuplot`[2].

Question 7.1. *Dans un terminal tapez la commande :*

```
$ gnuplot
```

Après quelques lignes de présentation, vous entrez dans un dialogue avec un interpréteur de commandes `gnuplot`*. L'attente d'une commande est marquée par le prompt :*

```
gnuplot>
```

Tapez la commande

```
plot [-1:1] [0:20]  1/((1-x)*(1-x**2)*(1-x**5)*(1-x**10)*(1-x**20)*(1-x**50)*(1-x**100)*(1-
```

Vous allez comparer les courbes représentatives sur l'intervalle $]-1,1[$ de cette fonction f et de polynômes de la forme

$$p_n(x) = \sum_{k=0}^{n} c(k)x^k,$$

pour différentes valeurs de n.

Question 7.2. *Dans* `gnuplot`*, visualisez les représentations de f et p_4 avec la commande*

```
plot [-1:1] [0:20]  1/((1-x)*(1-x**2)*(1-x**5)*(1-x**10)*(1-x**20)*(1-x**50)*(1-x**100)*(1-
1+x+2*x**2+2*x**3
```

Question 7.3. *Recommencez le travail précédent avec des valeurs de n croissantes. Que constatez-vous ?*

Pour avoir une explication du lien entre la fonction f et les polynômes p_n, il faut étudier les développements en série de la fonction f.

[2] cf `http://www.gnuplot.info`.