(M1102) Examen d'algo / prog

Le 27 novembre 2020 durée : 3 heures

Alain Casali alain.casali@univ-amu.fr
Aix Marseille Université
I.U.T. d'Aix en Provence - Département Informatique

Preambule

Documents autorisés : une feuille simple A4 manuscrite recto / verso.
Remarques :

- Lire attentivement tout le sujet, avant de commencer. Le barème, donné à titre indicatif, est sur 20 points, et 0 de bonus ;
- Tout au long de ce test, donnez de la sémantique à vos variables ! Toute variable n'ayant pas la sémantique adaptée sera considérée comme fausse, et par conséquent le morceau de code dans laquelle elle est présente aussi ;
- Toutes les fonctions écrites ou utilisées en cours ou en TD sont utilisables dans l'état ;
- Écriture au crayon de papier interdite.

Exercice 1.

Dans cet exercice, on cherche à écrire un programme qui permet :

— De générer aléatoirement un tableau `TabInt` d'entiers naturels ;
— De calculer et afficher, pour chaque élément du tableau, le nombre d'éléments qui lui sont strictement supérieures.

Plus précisément, vous devez écrire un algorithme qui :

— demande la saisie avec validation d'un nombre entier positif N ;
— demande la saisie avec validation d'un nombre entier positif M. On s'assure que M > N ;
— génère le tableau d'entiers positifs `TabInt` (de taille N) aléatoirement dans l'intervalle [1 ; M]
— parcourt le tableau `TabInt`, et pour chaque indice, produit l'affichage suivant :
A la position i du tableau, on trouve le nombre j
Il a exactement k valeur(s) supérieure(s) dans le tableau
où i, j, k devront être remplacés par les valeurs correspondantes.

Question 1.1 (4 points) :

Ecrire l'algorithme.

Question 1.2 (4 points) :

Modifiez l'algorithme précédant de façon à ce qu'on ne fasse qu'un unique affichage si le nombre à l'indice j du tableau a déjà été rencontré précédemment. Pour cela, vous devez vous appuyer sur un autre tableau d'entiers qui contiendra la liste des nombres déjà rencontrés.

Exercice 2.

Le but de cet exercice est d'écrire un algorithme permettant de prévoir combien va rapporter une somme d'argent S placée sur un compte à taux fixe T au bout de D années. Par exemple, si on place 1000 sur un livret A ayant pour taux d'intérêt 4.25% pendant 10 ans, alors on disposera de 1516.26€à terme.

Question 2.1 (5 points) :

A l'instar de la fonction `SaisirEntierSupX ()`, nous disposons d'une fonction `SaisirReelPositif ()` permettant de saisir un réel positif avec validation. Le profil de cette fonction est le suivant :

```
fonction SaisirReelPositif (Invite : in string, MsgErr : in string) renvoie reel ;
```

Ecrire un algorithme qui :

1. demande la saisie avec validation de la somme d'argent S de type `entier_naturel` ;
2. demande la saisie avec validation du taux d'intérêt T de type `reel` ;

3. demande la saisie avec validation de la durée du placement D de type `entier_naturel` ;
4. effectue le calcul du nouveau capital et l'affiche à l'écran comme suit :
   Au bout de # années, votre capital sera de $ euros
   où # et $ devront être remplacés par les valeurs correspondantes.

Exercice 3.

Dans cette partie, nous supposons qu'on saisit uniquement des chaines de caractères au clavier. En conséquence, il faut modifier le corps des fonctions servant à la saisie avec validation. Remarquez que c'est l'hypothèse utilisée dans la plupart des langages de programmation modernes (Java, PHP, C#,C++,). Pour cela, nous disposons des deux prédicats `StringToInt()` et `StringToFloat()` dont le profil est donné ci-dessous :

```
fonction StringToInt (Chaine : in string ; Nb : out entier) renvoie booleen;

fonction StringToFloat (Chaine : in string ; R : out reel) renvoie booleen;
```

Ces deux prédicats renvoient un booléen valant vrai si la chaine de caractères Chaine est convertible respectivement en entier ou en réel, et faux sinon.

Question 3.1 (2 points) :

Transformez le profil de ces prédicats en procédure.

Question 3.2 (5 points) :

Ecrivez le corps de la fonction `StringToInt ()` selon les spécifications imposées ci-dessus. Pour cela, nous rappelons que :

- seul le premier caractère de la chaine de caractères Chaine peut valoir +' ou - (mais ce n'est pas une obligation) ;
- on dispose de la fonction rang applicable aux caractères, qui renvoie le rang (un `entier_naturel`), dans l'ensemble des caractères, du caractère ASCII qui lui est passé en paramètre : `rang ('8') - rang ('0') = 8` D'une manière générale, pour un caractère C donné et représentant un chiffre, nous avons la formule : `rang(c)  rang(0') = valeur_numerique_de_c`
- Pour transformer une chaîne de caractère valide, nous utilisons l'algorithme de Horner. Cet algorithme est fourni de manière verbeuse ci-dessous :
  1. Testez si le premier caractère de chaine et un '+' (cas 1), un '-' (cas 2) ou un chiffre (cas 3). Si on n'est dans aucun de ces trois cas, alors renvoyer Faux. Pour les deux premiers cas, si le deuxième caractère existe et est un chiffre, initialisez `Nb` à `rang(Chaine[1]) - rang('0')` si on est dans le cas 1, ou a `rang('0') - rang(Chaine[1])` si on est dans le cas 2. Dans le cas 3, l'initialisation de `Nb` se fait avec `rang(Chaine0) - rang('0')`.
  2. Parcourez la chaine de gauche à droite. Pour chaque indice i rencontré, si le caractère courant n'est pas un chiffre, alors renvoyez Faux. Sinon appliquer la formule suivante pour mettre à jour `Nb` : $Nb \leftarrow Nb * 10 + rang(Chaine[i]) - rang('0')$.
  3. Si la fin de la chaîne de caractères a été atteinte, alors renvoyez Vrai.