L3 MIAGE Nanterre
2016-2017

Algorithmique et programmation
Feuille de TD numéro 1.

Invariant de boucle

Exercice 1. On voudrait savoir ce que fait l'algorithme suivant sur le tableau `T` dont les indices de début et de fin sont respectivement `premier` et `dernier`.

```
j := premier
pour i de premier à dernier - 1
        si T[i] <= T[dernier] alors
            echanger T[i] et T[j]
            j := j + 1
        fin si
fin pour
echanger T[dernier] et T[j]
```

Pour cela on va utiliser les invariants de boucle.
Question 1 Pour avoir une idée du déroulement de l'algorithme faites le tourner sur le tableau suivant avec `premier` valant 0 et `dernier` valant 7.

2	9	3	8	6	10	4	5

Question 2 Soit la propriété $P(i, j)$: "*tous les termes d'indice compris entre* 0 *et* $j - 1$ *sont plus petits que* `T[dernier]` *alors que ceux d'indice compris entre* j *et* $i - 1$ *sont plus grands.*"
-Montrez que $P(0, 0)$ est vraie.
-Montrez par récurrence que $P(i, j)$ est vraie à l'entrée de la boucle i.
Question 3 Indiquez ce que fait le programme.

Exercice 2. Pierre prétend que le code suivant permet de vérifier si la lettre e apparait dans le tableau de caractères T de taille N. On supposera les variables déclarées et le tableau initialisé.

```
int propriete=0;
for(i=0;i<N;i++){
   if(T[i]=='e') propriete=1;
   else          propriete=0;
}
```

Question 1 Déroulez le programme pour le tableau $T = [a, b, g, j, l]$.
Pour le tableau $U = [e, g, e, g, h]$.
Question 2 A l'aide d'un invariant de boucle montrez lui que son affirmation est fausse.

Manipulation de tableau

Exercice 3 (Produit de matrices)**.** Dans cet exercice, vous coderez les matrices comme des tableaux de pointeurs sur des tableaux. Soient A et B deux matrices carrées de dimension n. (N.B. On appelle matrice carrée de dimension n un tableau de flottants à deux dimensions de taille $n \times n$). On note $A[i][j]$ le coefficient (d'indice i,j) de la matrice A situé sur la ligne i et la colonne j.

Le produit $C = A \times B$ de deux matrices carrées A et B de dimension n est une matrice carrée de dimension n dont le terme d'indice i,j est calculé comme la somme des produits terme à terme des coefficients de la ligne i par ceux de la colonne j. C'est-à-dire (sous forme mathématique) par

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] * B[k][j].$$

Par exemple, si A et B sont des matrices carrées de taille 3 alors

$$C[i][j] = A[i][0] * B[0][j] + A[i][1] * B[1][j] + A[i][2] * B[2][j]$$

Question 1 Ecrivez la fonction `terme_produit` qui renvoie le coefficient d'indice i,j du produit de la matrice A par la matrice B. Le prototype est

```
float terme_produit(int i, int j, int n, float **A, float **B);
```

Question 2 Ecrivez le code de la fonction qui crée la matrice produit, calcule tous les termes de la matrice produit de A et de B et renvoie l'adresse de l'espace créé. Le prototype est

```
float ** produit_mat(int n,float **A, float **B);
```

Question 3 Ecrivez la fonction `initiale` qui va créer une matrice et remplir les coefficients au hasard entre -100 et 100. Le prototype est

```
float ** initiale(int n);
```

Pour tirer un flottant on utilisera la fonction `drand48()` qui tire un nombre entre 0 et 1. La première instruction du programme doit être `srand48(time(NULL))` et vous devez inclure les bibliothèques `stdlib.h` et `time.h`.

Question 4 Complétez les fonctions précédentes dans le programme suivant :

```
int main(){
  int n=15;
  float **A, **B, **C;
  ...=initiale(n);
  ...=initiale(n);
  ...=produit_mat(int n,...,...);
}
```

Exercice 4 (Déterminant d'une matrice de taille 3)**.** Dans cet exercice, vous coderez les matrices comme des tableaux à 2 dimensions, dans le style du C ansi. Soit une matrice A définie par

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

La valeur du déterminant de A, selon la règle de Sarrus, est

$$a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{13} \cdot a_{21} \cdot a_{32} - a_{31} \cdot a_{22} \cdot a_{13} - a_{32} \cdot a_{23} \cdot a_{11} - a_{33} \cdot a_{21} \cdot a_{12} \,.$$

Question 1 Ecrivez la fonction $det(...)$, dont vous remplirez vous même l'en tête, qui calcule et renvoie le déterminant.
Question 2 Intégrez la fonction précédente dans le programme suivant :

```
int main()
{ float A[3][3]={{1,2,3},{4,5,6},{7,8,9}};
  float resultat=det(....);
  return 0;
}
```

Exercice 5 (Insertion d'un élément)**.** L'exercice est un exercice corrigé dans la moulinette.
Soit Tab un tableau d'entiers de taille $taille$ contenant n termes avec $n < taille$.
Question 1 Ecrivez la fonction `insere` qui ne fait rien si l'élément b n'apparaît pas dans le tableau et sinon insére un élément a juste avant la première occurence de b. Son prototype est

```
void insere(int *T, int n, int taille, int a, int b);
```

Question 2 Appliquez votre fonction dans un programme pour le tableau

$$Tab = [0, 3, 14, 6, 15, \cdot, \cdot]$$

en insérant l'élément 10 avant l'élément 6 et l'élément 2 avant l'élément 15.

Exercice 6 (Cryptage simple)**.** L'algorithme de cryptage de César est un algorithme qui prend une clef k et qui remplace la $n$ième lettre de l'alphabet dans le message en clair par la $n+k$ième modulo 27 (le 27ème caractère est l'espace) dans le message crypté. Modulo signifie que si on dépasse la limite supérieure des lettres, on reprend au début de l'alphabet.

Par exemple si la clef est $k = 4$ alors la lettre a est remplacée par la lettre e et la lettre z par c.

Ecrivez la fonction `decrypt(char *c,j)` qui prend en entrée un tableau de caractères de taille j qui est la phrase cryptée et qui affiche les 26 possibilités de phrases en clair.

Suites récurrentes

Exercice 7 (Fraction continue)**.** La décomposition en fraction continue d'un nombre α est une expression

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \cfrac{1}{\ddots}}}}}$$

où tous les termes a_0, a_1, ..., a_i, ... sont entiers et les dénominateurs de toutes les fractions sont positifs. Ces termes peuvent se calculer sous la forme d'une équation récurrente. On pose $a_i = \lfloor u(i) \rfloor$, et la suite $u(n)$ est définie par $u(0) = \alpha$ et $u(i+1) = \frac{1}{u(i) - a_i}$.

Par exemple le nombre 2.4 se décompose de la manière suivante :

$$2.4 = \frac{12}{5} = 2 + \frac{2}{5} = 2 + \frac{2}{5} = 2 + \frac{1}{2 + \frac{1}{2}}\,.$$

Si α est rationnel alors la suite u est finie, le dernier terme étant $a_n = u(n)$. On peut alors poser par convention, $a_i = u(i) = \infty$ pour $i > n$.

Question 1 Ecrivez la fonction `decomp` dont le prototype est

```
void decomp(double alpha, int n);
```

qui affiche les n premiers coefficients (les termes a_n) de la décomposition en fraction continue du nombre α. Si la suite est trop courte (si la décomposition comporte moins de n termes), les ∞ qui suivent ne seront pas affichés.

Question 2 Ecrivez (en modifiant légèrement votre fonction précédente) la fonction `decomp_tab` qui remplit un tableau avec les n premiers coefficients de la décomposition en fraction continue du nombre α, en complétant par des 0 si la suite est trop courte. Ce tableau sera supposé créé à la bonne taille. Le prototype est

```
void decomp_tab(double alpha, int n, int *tableau);
```

On appelle convergent d'ordre k d'un nombre α le nombre rationnel obtenu en utilisant les k premiers coefficients de la décomposition en fraction continue de α. Ce nombre est noté $r_k(\alpha)$.

$$r_k(\alpha) = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots\ {\scriptstyle a_{k-2} + \frac{1}{a_{k-1}}}}}}\,.$$

Par exemple $r_2(2,4) = 2 + \frac{1}{2} = \frac{5}{2} = 2.5$.

On peut aussi calculer $r_k(\alpha)$ grâce à une suite récurrente : On initialise $r := a_{k-1}$ puis pour i allant de $k-2$ à 0 on fait $r := a_i + 1/r$.

Question 3 On suppose qu'on a la décomposition en fraction continue du nombre α. Cette décomposition est d'ordre supérieur à k c'est-à-dire qu'elle comporte plus de k termes. Cette décomposition est rangée dans un tableau. Ecrivez la fonction `convergent` qui renvoie la valeur approchée du convergent d'ordre k à partir de cette décomposition. Le prototype est

```
double convergent(int k, int *tableau);
```

Question 4 Soit $\pi = 3.141592653589$. Calculez la décomposition en fraction continue d'ordre 6 et affichez les 5 premiers convergents.
Question 5 Il est possible d'améliorer le temps de calcul des n premiers convergents de la question 4, en remarquant que $r_i(\alpha) = v_i/w_i$ avec

$$v_{-2} = 0,\ v_{-1} = 1,\ v_i = a_i v_{i-1} + v_{i-2}$$

et

$$w_{-2} = 1,\ w_{-1} = 0,\ w_i = a_i w_{i-1} + w_{i-2}\,.$$

Refaites le calcul des convergents en utilisant cette observation.
Question 6 On peut éviter les pertes de précision des questions 1 et 2 lors du calcul de l'inverse d'un nombre réel, en appliquant plutôt l'algorithme d'Euclide à α et 1. Pour cela, on peut introduire une nouvelle suite récurrente u' qui se calcule par $u'_{-1} = \alpha$, $u'_0 = 1$ et

$$u'_{i+1} = u'_{i-1} - a_i \cdot u'_i$$

on calcule alors a_i par $a_i = \lfloor u'_{i-1}/u'_i \rfloor$ Complétez la fonction `decomp2` qui calcule la décomposition en fraction continue en utilisant cette nouvelle manière. L'en tête de cette fonction est :

```
void decomp2(long double alpha);
```

Question 7 Exécutez `decomp2(M_PI); decomp2(2*acosl(0));`