ECUE «Introduction à la programmation »

Contrôle continu n°3 – 10 janvier 2013
sans document - durée 1 heure 30

Exercice 1 (2 points)

Avec des boucles `for`, écrire un programme affichant toutes les solutions de l'équation `(1)` :

`A x B = C` `(1)`

où `A`, `B` et `C` sont trois chiffres compris entre `1` et `9` tous distincts.

Exercice 2 (6 points)

1) Ecrire une fonction `int deIntervalleANombre(int a, int b)` demandant à l'utilisateur un nombre entier appartenant à l'intervalle `[a,b]` et retournant ce nombre. La fonction demande répétitivement le nombre à l'utilisateur tant que le nombre n'appartient pas à `[a,b]`.

2) Ecrire une fonction `void tabMaxMin` prenant en entrée un tableau d'entiers `tab` et une longueur `l` de tableau, et donnant en sortie le maximum `max` et le minimum `min` des valeurs du tableau.

3) Ecrire un programme `main` remplissant un tableau de `3` entiers appartenant à l'intervalle `[0,9]` en utilisant la fonction `deIntervalleANombre`, puis affichant les valeurs du tableau, puis appelant `tabMaxMin` et affichant le maximum et le minimum des valeurs du tableau.

On respectera les entrée-sorties de l'exécution ci-dessous:

```
t[0] : x ? (0<=x<=9) 10
x ? (0<=x<=9) -1
x ? (0<=x<=9) 0
t[1] : x ? (0<=x<=9) 9
t[2] : x ? (0<=x<=9) 5
t[0] = 0 , t[1] = 9 , t[2] = 5 ,
max = 9, min = 0
```

Exercice 3 (6 points)

On considère la suite U_n de nombres réels définie les équations `(2)`:

$U_0 = 0$

$U_{n+1} = U_n + 4/(2n+1)$ si n est pair. `(2)`

$U_{n+1} = U_n - 4/(2n+1)$ si n est impair.

1) Ecrire une fonction `float piLeibniz1(int n)` affichant les `n` premiers termes de la suite U_n et retournant le dernier terme.

2) Ecrire une fonction `float piLeibniz2(float pr, int * n)` affichant les termes de la suite U_n tant que $|U_n - U_{n-1}| > pr$, retournant le dernier terme, et mettant le nombre d'itérations effectuées dans `*n`.

3) Ecrire une fonction `main` appelant la fonction `piLeibniz2` avec une précision `0.01`. et affichant le résultat et le nombre d'itérations effectuées.

Exercice 4 (6 points)

On considère les suites A_n , B_n , C_n , D_n , P_n de nombres réels définies de la manière suivante:

$A_0=1$ $B_0=1/2^{1/2}$ $C_0=1/4$ $D_0=1$ (3)

$A_{n+1}=(A_n+B_n)/2$ $B_{n+1}=(A_nB_n)^{1/2}$ $C_{n+1}=C_n-D_n(A_n-B_n)^2/4$ $D_{n+1}=2D_n$ (4)

$P_n = (A_n+B_n)^2/(4C_n)$ (5)

1) a) Donner la sortie du traitement suivant:

```
float a=1, b, c=0.25, d=1; b=sqrt(0.5);
printf("a0=%.2f, b0=%.2f, c0=%.2f, d0=%.2f\n", a, b, c, d);
```

1) b) Donner une approximation de A_1, B_1, C_1, D_1 avec 2 chiffres après la virgule.

2) Donner la sortie du traitement suivant:

```
float a=1, b, c=0.25, d=1; b=sqrt(0.5);
a = (a+b)/2;
b = sqrt(a*b);
c = c - d*(a-b)*(a-b)/4;
d = 2*d;
printf("a1=%.2f, b1=%.2f, c1=%.2f, d1=%.2f\n", a, b, c, d);
```

3) Modifier le traitement ci-dessus pour qu'il corresponde à la définition `(4)`.

4) Ecrire une fonction `void deABCDaABCD(float * a, float * b, float * c, float * d)` prenant `a`, `b`, `c`, `d` en entrée et les valorisant en sortie selon la définition `(4)`.

5) Ecrire une fonction `float deABCaP(float a, float b, float c)` prenant `a`, `b`, `c` en entrée et retournant une valeur correspondant à la définition `(5)`.

6) Ecrire une fonction `float piBrentSalamin(int n)` affichant les `n` premiers termes de la suite P_n et retournant le dernier terme calculé. On utilisera `deABCDaABCD` et `deABCaP`.

7) Ecrire une fonction `main` appelant `piBrentSalamin` avec `3` itérations et affichant le résultat.

Epilogue

Les suites U_n et P_n des exercices 3 et 4 convergent vers Π. La fonction `piLeibniz2` donne Π avec une précision `0.01` au bout de deux cents itérations et avec une précision `0.000001` au bout de deux millions d'itérations. La fonction `piBrentSalamin` donne Π avec une précision `0.01` en une seule itération et avec une précision `0.000001` en deux itérations. Pour obtenir plus de précision, il faut abandonner le type `float`.