ECUE «Introduction à la programmation » - Session 2

12 juin 2013 - Bruno Bouzy

sans document - durée 1 heure 30

Corrigé

Exercice 1 (2 points)

Ecrire un programme `exo1.c` permettant à l'utilisateur d'entrer un nombre de tentatives et un nombre de succès et affichant le pourcentage de succès. La sortie du programme doit respecter la sortie ci-dessous. On suppose que l'utilisateur entre des valeurs strictement positives. On affichera le pourcentage avec un chiffre après la virgule.

```
nombre de tentatives ? 23
nombre de tentatives = 23.0
nombre de succes ? 17
nombre de succes = 17.0
pourcentage succes = 73.9
```

```
#include <stdio.h>
int main() {
  float t, s;
  printf("nombre de tentatives ? ");
  scanf("%f", &t);
  printf("nombre de tentatives = %.1f\n", t);
  printf("nombre de succes ? ");
  scanf("%f", &s);
  printf("nombre de succes = %.1f\n", s);
  printf("pourcentage succes = %.1f\n", 100*s/t);
  return (0);
}
```

Exercice 2 (4 points)

Donner la sortie du programme suivant.

```
int main() {
  int a = 3;  int * p = &a;  int b = *p;
                printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);
  a *= 2;       printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);
  b += 4;       printf("3: a = %d, b = %d, *p = %d.\n", a, b, *p);
  int * q = &b; printf("4: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
  *q *= (*p)++; printf("5: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
  *q += ++(*p); printf("6: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
  p = q;        printf("7: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
  q = &a;       printf("8: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
  return(0);
}
```

```
1: a = 3, b = 3, *p = 3.
2: a = 6, b = 3, *p = 6.
3: a = 6, b = 7, *p = 6.
4: a = 6, b = 7, *p = 6, *q = 7.
5: a = 7, b = 42, *p = 7, *q = 42.
6: a = 8, b = 50, *p = 8, *q = 50.
7: a = 8, b = 50, *p = 50, *q = 50.
8: a = 8, b = 50, *p = 50, *q = 8.
```

(0.5 point par ligne)

Exercice 3 (6 points)

On considère les suites de nombres réels A_n et B_n définies de la manière suivante:

$A_0=1$ $B_0=9$ (3) $A_{n+1}=(A_n+B_n)/2$ $B_{n+1}=(A_nB_n)^{1/2}$ (4)

1) a) Quelles sont les valeurs de A_1 et B_1 ?

A1 = 5 B1=3
(0.25 point)

1) b) Donner la sortie de:

```
float a=1, b=9; a=(a+b)/2; b=sqrt(a*b); printf("a1=%.1f, b1=%.1f\n", a, b);
```

a1=5.0, b1=6.7
(0.5 point)

1) c) Cette sortie est-elle compatible avec la définition (4) ?

non
(0.25 point)

1) d) Modifier le traitement de la question 1) b) pour qu'il corresponde à la définition (4).

```
float aApres=(a+b)/2;
b=sqrt(a*b);
a=aApres;
```
(1 point)

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

```
void deABaAB(float * a, float * b) {
  float aApres = (*a+*b)/2;
  *b = sqrt(*a**b);
  *a = aApres;
}
```
(1.5 point)

3) Ecrire une fonction `void suite(int n)` affichant les `n` premiers termes des suites A_n et B_n avec 6 chiffres après la virgule. On utilisera `deABaAB`.

```
void suite(int ni) {
  printf("suite:\n");
  int n=0;
  printf("nIterations = %2d: ", n);
  float a=A_0;
  float b=B_0;
  printf("a = %8.6f, b = %8.6f\n", a, b);
  for (n=1; n<=ni; n++) {
    printf("nIterations = %2d: ", n);
    deABaAB_3(&a, &b);
    printf("a = %8.6f, b = %8.6f\n", a, b);
  }
}
```
(2 points)

4) Ecrire un programme `main` appelant `suite` avec `n=3`.

```
int main() {
  suite(3);
  return 0;
}
```
(0.5 point)

Exercice 4 (8 points)

Soit `triRapide.c` le programme suivant.

```
#define TAILLE 8

void affiche(int * t, int premier, int dernier) {
  int i; printf("[ ");
  for (i=0; i<premier; i++) printf ("_ ");
  for (i=premier; i<=dernier; i++) printf ("%d ", t[i]);
  for (i=dernier+1; i<TAILLE; i++) printf ("_ ");
  printf("]\n");
}

int partition(int * t, int premier, int dernier) {
  int i, tmp, j = premier;
  for (i=premier; i<dernier; i++) {
    if (t[i] <= t[dernier]) {
      tmp = t[i]; t[i] = t[j]; t[j] = tmp; j++;
    }
    printf("P: i = %d j = %d ", i, j); affiche(t, premier, dernier);
  }
  tmp = t[dernier]; t[dernier] = t[j]; t[j] = tmp;
  printf("P: pivot = %d ", j); affiche(t, premier, dernier);
  return j;
}

void triRapide(int * t, int premier, int dernier, int niveau) {
  if (premier<dernier) {
    printf("TR debut: pre = %d der = %d niv = %d\n", premier, dernier, niveau);
    int pivot = partition(t, premier, dernier);
    triRapide(t, premier, pivot-1, niveau+1);
    triRapide(t, pivot+1, dernier, niveau+1);
    printf("TR fin: pre = %d der = %d niv = %d ", premier, dernier, niveau);
    affiche(t, premier, dernier);
  }
}

int main() {
  int tab[] = { 6, 25, 18, 0, 15, 10, 12, 16 };
  triRapide(tab, 0, TAILLE-1, 0);
  return 0;
}
```

1) Avec quelles valeurs de `premier`, `dernier` et `niveau`, `triRapide` est-elle appelée par `main` ?

`0, 7, 0`
(0.25 point)

2) a) Donner la sortie du programme jusqu'à la fin de la première exécution de `partition`.

```
TR debut: pre = 0 der = 7 niv = 0
P: i = 0 j = 1 [ 6 25 18 0 15 10 12 16 ]
P: i = 1 j = 1 [ 6 25 18 0 15 10 12 16 ]
P: i = 2 j = 1 [ 6 25 18 0 15 10 12 16 ]
P: i = 3 j = 2 [ 6 0 18 25 15 10 12 16 ]
P: i = 4 j = 3 [ 6 0 15 25 18 10 12 16 ]
P: i = 5 j = 4 [ 6 0 15 10 18 25 12 16 ]
P: i = 6 j = 5 [ 6 0 15 10 12 25 18 16 ]
P: pivot = 5 [ 6 0 15 10 12 16 18 25 ]
(2 points)
```

2) b) En quelle position du tableau la dernière valeur du tableau a-t-elle été déplacée ?

```
En position 5
(0.5 point)
```

2) c) Cette position s'appelle le pivot. Quelle est la valeur du tableau au pivot ?

```
Sa valeur est 16.
(0.25 point)
```

2) d) Quelle propriété vérifient les valeurs dont la position est inférieure à la valeur du pivot ?

```
Elles sont inférieures à celle du pivot.
(0.25 point)
```

2) e) Quelle propriété vérifient les valeurs dont la position est supérieure à la valeur du pivot ?

```
Elles sont supérieures à celle du pivot.
(0.25 point)
```

3) a) Avec quelles valeurs de `premier` et `dernier`, `triRapide` de niveau 1 est-elle appelée la première fois par `triRapide` de niveau 0 ? la seconde fois ?

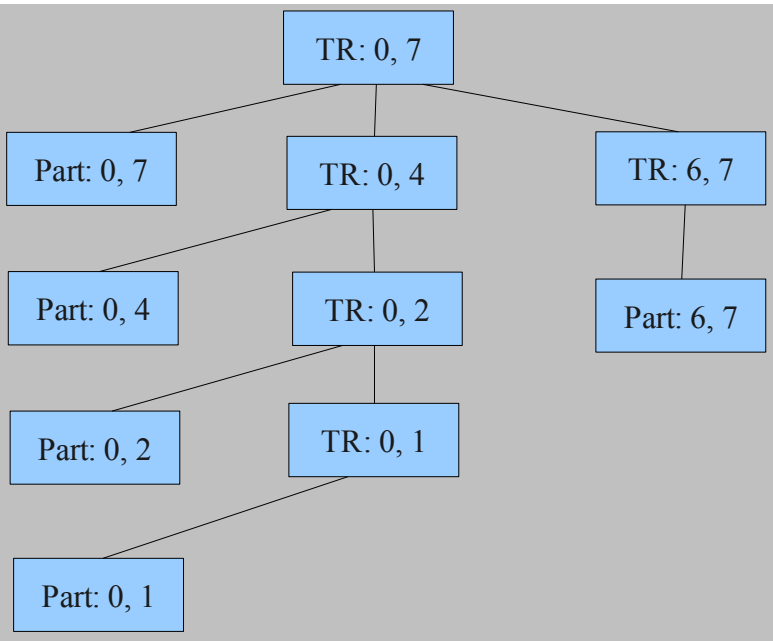
```
0 et 4 la première fois
6 et 7 la seconde fois
(0.5 point)
```

3) b) Donner la suite de la sortie du programme jusqu'à la fin de l'exécution du `main`.

```
TR debut: pre = 0 der = 4 niv = 1
P: i = 0 j = 1 [ 6 0 15 10 12 _ _ _ ]
P: i = 1 j = 2 [ 6 0 15 10 12 _ _ _ ]
P: i = 2 j = 2 [ 6 0 15 10 12 _ _ _ ]
P: i = 3 j = 3 [ 6 0 10 15 12 _ _ _ ]
P: pivot = 3 [ 6 0 10 12 15 _ _ _ ]
TR debut: pre = 0 der = 2 niv = 2
P: i = 0 j = 1 [ 6 0 10 _ _ _ _ _ ]
P: i = 1 j = 2 [ 6 0 10 _ _ _ _ _ ]
P: pivot = 2 [ 6 0 10 _ _ _ _ _ ]
TR debut: pre = 0 der = 1 niv = 3
P: i = 0 j = 0 [ 6 0 _ _ _ _ _ _ ]
P: pivot = 0 [ 0 6 _ _ _ _ _ _ ]
TR fin: pre = 0 der = 1 niv = 3 [ 0 6 _ _ _ _ _ _ ]
TR fin: pre = 0 der = 2 niv = 2 [ 0 6 10 _ _ _ _ _ ]
TR fin: pre = 0 der = 4 niv = 1 [ 0 6 10 12 15 _ _ _ ]
TR debut: pre = 6 der = 7 niv = 1
P: i = 6 j = 7 [ _ _ _ _ _ _ 18 25 ]
P: pivot = 7 [ _ _ _ _ _ _ 18 25 ]
TR fin: pre = 6 der = 7 niv = 1 [ _ _ _ _ _ _ 18 25 ]
TR fin: pre = 0 der = 7 niv = 0 [ 0 6 10 12 15 16 18 25 ]
```

(2.5 points)

3) c) Dessiner l'arbre d'appel des fonctions `triRapide` et `partition`. Chaque noeud de l'arbre correspondra à un appel de `triRapide` pour lequel `premier<dernier` est vrai ou à un appel de `partition`. Dans chaque noeud, on précisera les valeurs de `premier`, `dernier`.



(1.5 point)