ECUE «Introduction à la programmation » - Session 2

11 juin 2015 - Bruno Bouzy

sans document - durée 1 heure 30

CORRIGE

Exercice 1 (4 points)

Donner la sortie du programme suivant.

```
int main() {

  int a = 7;
  int * p = &a;
  int b = *p;
                printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);
  a *= 5;       printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);
  b += 3;       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 = 7, b = 7, *p = 7.
2: a = 35, b = 7, *p = 35.
3: a = 35, b = 10, *p = 35.
4: a = 35, b = 10, *p = 35, *q = 10.
5: a = 36, b = 45, *p = 36, *q = 45.
6: a = 37, b = 1665, *p = 37, *q = 1665.
7: a = 37, b = 1665, *p = 1665, *q = 1665.
8: a = 37, b = 1665, *p = 1665, *q = 37.
```

0.5 point par ligne correcte.

Exercice 2 (16 points)

Dans cet exercice, on veut écrire `crepe.c` un programme C mélangeant au hasard un tas de crêpes de tailles toutes différentes, ceci en utilisant itérativement des inversions du haut du tas.

Généralités :

Une crêpe est représentée par un entier. Le tas de crêpes est représenté par un tableau d'entier `tas`.

Au début, l'utilisateur entre la taille `taille` du tas de crêpes, le nombre d'actions de mélange `nmelanges` et une graine `graine`.

Ensuite, le programme initialise le tas de crêpes par ordre croissant de taille, la plus grande en bas et la plus petite en haut.

Puis, le programme effectue `nmelanges` actions d'inversion tirées au hasard.

Après chaque action, le programme affiche le tas de crêpes visuellement avec des suites de − et numériquement avec les nombres du tableau `tas`.

Entrées sorties:

4 exécutions du programme sont données ci-dessous : dans la colonne de gauche, une exécution normale et dans la colonne de droite, 3 exécutions avec une erreur d'entrée de l'utilisateur.

Les valeurs tapées par l'utilisateur au clavier sont en gras.

```
crepe: debut:
taille ? (1 < taille < 20) 6
nmelanges ? (0 < nmelanges) 2
graine ? (0 < graine) 1
      --
     ----
    ------
   --------
  ----------
 ------------
==============
{  0  1  2  3  4  5  6  7 }
action = 2
     ----
      --
    ------
   --------
  ----------
 ------------
==============
{  0  2  1  3  4  5  6  7 }
action = 5
  ----------
   --------
```

```
crepe: debut:
taille ? (1 < taille < 20) 20
erreur
crepe: fin.
```

```
crepe: debut:
taille ? (1 < taille < 20) 6
nmelanges ? (0 < nmelanges) -3
erreur
crepe: fin.
```

```
    ------
      --
     ----
 ------------
==============
{  0  5  4  3  1  2  6  7 }
crepe: fin.
```

```
crepe: debut:
taille ? (1 < taille < 20) 6
nmelanges ? (0 < nmelanges) 3
graine ? (0 < graine) 0
erreur
crepe: fin.
```

Dans la première exécution de la colonne de droite, l'utilisateur a entré une taille trop grande.

Dans la seconde, il a entré une taille correcte mais un nombre de mélanges négatif.

Dans la troisième, la graine est incorrecte.

Dans ces 3 exécutions, le programme affiche `erreur` et se termine.

Dans l'exécution de la colonne de gauche, les 3 valeurs entrées par l'utilisateur sont correctes et le programme s'exécute normalement. L'utilisateur a demandé une exécution avec 6 crêpes, 2 actions de mélanges et une graine 1. A la première action, 2 crêpes ont été inversées et 5 crêpes à la deuxième action.

Précisions:

Pour un tas de crêpes, le nombre de crêpes du tas est égal à `taille`. Pour chaque nombre `i` entre `1` et `taille`, il y a une crêpe unique ayant la taille `i`. On la nomme la crêpe `i`. La crêpe `i` est affichée avec une suite de `2i -`.

On notera que le plateau sur lequel repose le tas est affiché avec des =, que la taille du plateau est `taille+1` et qu'il est représenté dans le tableau à l'indice `taille+1`. On notera aussi que l'espace au dessus du tas de crêpes est représenté par un `0`, qu'il est représenté dans le tableau à l'indice `0`. On distinguera donc `taille` qui est le nombre de crêpes du tas et `taille+2` qui est le nombre d'entiers dans le tableau `tas`.

Après avoir initialisé le tableau `tas` dans l'ordre, celui-ci est affiché sous forme de crêpes et sous forme numérique.

Puis, une valeur d'action est tirée au hasard entre `1` et `taille`. Dans l'exemple, la valeur de la première action est `2`, celle de la seconde est `5`. La valeur d'une action correspond au nombre de crêpes inversées en haut du tas.

Après avoir effectué l'action, c'est-à-dire inversé les crêpes du haut du tas, le programme affiche le nouveau tas sous forme de crêpes et sous forme numérique.

Le programme effectue `nmelanges` actions et se termine. Dans l'exécution de la colonne de gauche, le programme a effectué l'action `2` puis l'action `5`.

Questions:

1) Définir `TAILLE_MAX` la taille effective maximale du tableau `tas` avec la valeur `21`.
(1 point).

```
#define TAILLE_MAX 21
```

2) Programmer le début du `main`. Il contient la déclaration du tableau `tas` (de taille `TAILLE_MAX`), des variables `taille`, `nmelanges` et `graine`. Il demande à l'utilisateur les valeurs de `taille`, `nmelanges` et `graine` en vérifiant que les valeurs entrées sont correctes. Il se termine en retournant `-1` sinon. On respectera les 4 exemples d'exécution.
(2 points).

```
int main() {
  int tas[TAILLE_MAX], taille, nmelanges, graine;

  printf("crepe: debut:\n");

  printf("taille ? (1 < taille < %d) ", TAILLE_MAX-1);
  scanf("%d", &taille);
  if ((taille<=1) || (taille>=TAILLE_MAX-1)) {
      printf("erreur.\n"); return -1;
  }

  printf("nmelanges ? (0 < nmelanges) ");
  scanf("%d", &nmelanges);
  if (nmelanges<=0) {
      printf("erreur.\n"); return -1;
  }

  printf("graine ? (0 < graine) ");
  scanf("%d", &graine);
  if (graine<=0) {
      printf("erreur.\n"); return -1;
  }
```

3) Programmer la procédure `void afficherNombres(int * tab, int tt)` affichant le tableau `tab`. `tt` est le nombre de crêpes. On respectera les entrées sorties de l'exécution.
(1 point).

```
void afficherNombres(int * tab, int tt)
{
  int p;
  printf("{ ");
  for (p=0; p<=tt+1; p++) printf("%2d ", *(tab+p));
  printf("}\n");
}
```

4) Programmer la procédure `void afficherCrepes(int * tab, int tt)` affichant le tas de crêpes avec des – et des = correspondant aux valeurs du tableau `tab`. `tt` est le nombre de crêpes. On respectera les entrées sorties de l'exécution.
(4 points).

```
void afficherCrepes(int * tab, int tt)
{
  int p, x;
  for (p=1; p<=tt; p++) {
    for (x=0; x<=tt-tab[p]; x++) printf(" ");
    for (x=0; x<2*tab[p];   x++) printf("-");
    printf("\n");
  }
  for (x=0; x<2*tt+2; x++) printf("=");
  printf("\n");
}
```

5) Programmer la procédure `void initialiser(int * tab, int tt)` initialisant le tableau `tab` avec un tas ordonné de `tt` crêpes, puis affichant le tas de crêpes et les nombres du tableau. On notera que le tas de crêpes ordonné correspond au tableau dont les valeurs sont égales aux indices du tableau (`tas[i]=i` pour `i` allant de `0` à `taille+1`). Cette procédure appellera les procédures des questions 3 et 4.
(2 points).

```
void initialiser(int * tab, int tt)
{
  int i;
  for (i=0; i<=tt+1; i++) tab[i] = i;
  afficherCrepes(tab, tt);
  afficherNombres(tab, tt);
}
```

6) Programmer la procédure `void effectuerAction(int * tab, int a)` inversant les `a` crêpes du dessus du tas correspondant au tableau `tab`. Son effet correspondra aux entrées sorties de l'exécution de la colonne de gauche donnée en exemple. On pourra utiliser un tableau intermédiaire `tmp`.
(3 points).

```
void effectuerAction(int * tab, int a)
{
  int tmp[TAILLE_MAX], p;
  for (p=1; p<=a; p++) { tmp[p] = tab[a-p+1]; }
  for (p=1; p<=a; p++) { tab[p] = tmp[p]; }
}
```

7) Pour un tableau `tab` correspondant à un tas de `tt` crêpes, programmer la procédure `void melanger(int * tab, int tt, int nmel)` effectuant `nmel` fois le traitement suivant :

(1) avec la fonction `rand()` tirage aléatoire du nombre `action` de crêpes à inverser en haut du tas,

(2) affichage de `action`,

(3) appel de `effectuerAction(tab, action)`,

(4) affichage du tas de crêpes,

(5) affichage des nombres du tableau,

conformément aux entrées sorties de l'exécution et en appelant les procédures appropriées.
(2 points).

```
void melanger(int * tab, int tt, int nmel)
{
  int n, action;
  for (n=1; n<=nmel; n++) {
    action = 1 + rand() % tt;
    printf("action = %d\n", action);
    effectuerAction(tab, action);
    afficherCrepes(tab, tt);
    afficherNombres(tab, tt);
  }
}
```

8) En appelant les procédures adéquates et en respectant les entrées sorties, finir la programmation du `main` commencée à la question 2. Entre autre, au bon endroit du `main`, on initialisera la suite aléatoire avec la variable `graine` et la fonction `srand`. Le `main` se termine en retournant `0`.
(1 point).

```
  ...
  srand(graine);

  initialiser(tas, taille);
  melanger(tas, taille, nmelanges);

  printf("crepe: fin.\n");
  return 0;
}
```