ECUE «Prog 1» - CC3
6 janvier 2016 - Bruno Bouzy
sans document - durée 1 heure 30

CORRIGE
exo1 sur 17 points et exo2 sur 4 points ==> examen sur 21 points.

Exercice 1 (17 points)

Dans cet exercice, on veut écrire `casseTete.c` un programme C permettant à un utilisateur de résoudre un casse-tête. Ci-dessous figure une exécution du programme définissant les règles du casse-tête et les entrées sorties du programme. Les entrées sont en caractères gras.

```
Voici le probleme:
  - - -
| 5 0 1 |
| 2 3 4 |
  - - -
action ? (nombre entier entre -3 et +3, 0 pour aide) 0
                  *****
6 actions sont possibles.
A chaque action correspond un nombre entier A dans [-3, +3].
 A > 0: rotation dans le sens trigonometrique.
 A < 0: rotation dans le sens des aiguilles d'une montre.
 A = 0: affichage de l'aide.
 |A| = 1: rotation des 4 nombres de gauche.
 |A| = 2: rotation des 4 nombres de droite.
 |A| = 3: rotation des 6 nombres.
                  *****
action ? (nombre entier entre -3 et +3, 0 pour aide) 3
  - - -
| 0 1 4 |
| 5 2 3 |
  - - -
action ? (nombre entier entre -3 et +3, 0 pour aide) 1
  - - -
| 1 2 4 |
| 0 5 3 |
  - - -
action ? (nombre entier entre -3 et +3, 0 pour aide) 1
  - - -
| 2 5 4 |
| 1 0 3 |
  - - -
action ? (nombre entier entre -3 et +3, 0 pour aide) -3
  - - -
| 1 2 5 |
| 0 3 4 |
  - - -
action ? (nombre entier entre -3 et +3, 0 pour aide) -3
  - - -
| 0 1 2 |
| 3 4 5 |
  - - -
Bravo! Resolution en 5 actions!
```

Pour commencer, le programme affiche le problème à résoudre sous forme d'un damier de hauteur 2, de largeur 3 et de taille 6. Puis, le programme demande à l'utilisateur une action. Au début, l'utilisateur a tapé l'action 0, ce qui lui a permis d'avoir le mode d'emploi du programme. Comme expliqué par le mode d'emploi, l'utilisateur peut effectuer une des 6 actions représentée par un nombre (-1, +1, -2, +2, -3, +3). Une action permet de faire tourner quatre ou six nombres du damier circulairement dans le sens trigonométrique si le nombre est positif, ou bien dans le sens des aiguilles d'une montre sinon. Par exemple, l'action 3 a permis aux six nombres du damier de tourner dans le sens trigonométrique. Puis l'action 1 a permis aux quatre nombres situés à gauche du damier (0, 1, 2, 5) de tourner dans le sens trigonométrique. Et ainsi de suite jusqu'à ce que le damier soit dans la configuration ordonnée (0 1 2 en haut et 3 4 5 en bas), ce qui termine le programme avec le message `Bravo!` et le nombre d'actions utilisées. Pour programmer `casseTete.c`, on va utiliser un tableau, un `main` et des fonctions C.

1° Ecrire l'en-tête nécessaire à tout programme C. **(0.25 pt)**.

```
#include <stdio.h>
```

2° Définir les constantes `LARGEUR, HAUTEUR, TAILLE` avec les valeurs `3, 2, 6`. **(0.25 pt)**.

```
#define LARGEUR 3
#define HAUTEUR 2
#define TAILLE 6
```

3° Ecrire `void imprimer(int *tab, int l)` affichant un tableau sur une ligne. **(0.5 pt)**.

```
void imprimer(int * tab, int l) {
  int i;
  for (i=0; i<l; i++) printf("%d ", tab[i]);
  printf("\n");
}
```

4° Ecrire `void afficher(int *tab, int hauteur, int largeur)` affichant un tableau en 2 dimensions en respectant les entrées sorties ci-dessus. **(2 points)**.

```
void afficher(int * tab, int hauteur, int largeur) {
  int i, j;
  printf("  "); for (j=0; j<largeur; j++) printf("- "); printf("\n");
  for (i=0; i<hauteur; i++) {
    printf("| ");
    for (j=0; j<largeur; j++) {
      printf("%d ", tab[i*largeur+j]);
    }
    printf("| \n");
  }
  printf("  "); for (j=0; j<largeur; j++) printf("- "); printf("\n");
}
```

5° Ecrire `void afficherModeEmploi()` affichant le mode d'emploi du casse-tête. **(0.5 pt)**.

```
void afficherModeEmploi() {
  printf("                    *****          \n");
  printf("6 actions sont possibles.\n");
  printf("A chaque action correspond un nombre entier A dans [-3, +3].\n");
  printf(" A > 0: rotation dans le sens trigonometrique.\n");
  printf(" A < 0: rotation dans le sens des aiguilles d'une montre.\n");
  printf(" A = 0: affichage de l'aide.\n");
  printf(" |A| = 1: rotation des 4 nombres de gauche.\n");
  printf(" |A| = 2: rotation des 4 nombres de droite.\n");
  printf(" |A| = 3: rotation des 6 nombres.\n");
  printf("                    *****          \n");
}
```

6° Ecrire `int fini(int *tab, int l)` retournant `1` si le casse-tête est fini, `0` sinon. **(0.5 point)**.

```
int fini(int * tab, int l) {
  int i;
  for (i=0; i<l; i++) if (tab[i]!=i) return 0;
  return 1;
}
```

7° Ecrire `int deClavierVersAction()` demandant une action valide (c'est-à-dire +1, -1, +2, -2, +3 ou -3) au clavier et retournant cette action. On utilisera une boucle. **(2 points)**.

```
int deClavierVersAction() {
  int action;
  do {
    printf("action ? (nombre entier entre -3 et +3, 0 pour aide) ");
    scanf("%d", &action);
    if (action==0) afficherModeEmploi();
  } while ((action==0) || (action<-3) || (action>+3));
  return action;
}
```

8° Ecrire `int deHasardVersAction()` retournant une action valide tirée au hasard. **(1 pt)**.

```
int deHasardVersAction() {
  int action = rand() % 6;
  if (action<3) return 1+action;
  else          return action-6;
}
```

9° Ecrire la procédure `void effectuerAction(int *tab, int action)` modifiant le tableau `tab` en fonction de l'action. **(3 points)**.

```
void effectuerAction(int * tab, int action) {
  int tmp;
  if (action==+1) {
    tmp = tab[0];
    tab[0] = tab[1];
    tab[1] = tab[4];
    tab[4] = tab[3];
    tab[3] = tmp;
  }
  else if (action==-1) {
    tmp = tab[0];
    tab[0] = tab[3];
    tab[3] = tab[4];
    tab[4] = tab[1];
    tab[1] = tmp;
  }
  else if (action==+2) {
    tmp = tab[1];
    tab[1] = tab[2];
    tab[2] = tab[5];
    tab[5] = tab[4];
    tab[4] = tmp;
  }
  else if (action==-2) {
    tmp = tab[1];
    tab[1] = tab[4];
    tab[4] = tab[5];
    tab[5] = tab[2];
    tab[2] = tmp;
  }
  else if (action==+3) {
    tmp = tab[0];
    tab[0] = tab[1];
    tab[1] = tab[2];
    tab[2] = tab[5];
    tab[5] = tab[4];
    tab[4] = tab[3];
    tab[3] = tmp;
  }
  else if (action==-3) {
    tmp = tab[0];
    tab[0] = tab[3];
    tab[3] = tab[4];
    tab[4] = tab[5];
    tab[5] = tab[2];
    tab[2] = tab[1];
    tab[1] = tmp;
  }
}
```

10° Ecrire la procédure `void deHasardACasseTete(int *tab, int l)` remplissant le tableau `tab` avec des nombres au hasard. **(3 points)**.

```
void deHasardACasseTete(int * tab, int l)
{
  int age, p, where, cnt;
  for (age=0; age<l; age++) tab[age] = -1;
  for (age=0; age<l; age++) { // printf("age = %d\n", age);
    where = rand() % (l-age); // printf("where = %d\n", where);
    cnt = 0;
    for (p=0; p<l; p++) {
      if (tab[p]!=-1) continue;
      if (cnt==where) { // printf("cnt = %d\n", cnt);
      tab[p] = age;
      break;
      }
      cnt++;
    }
  }
}
```

11° Ecrire la fonction `int resoudre(int *tab)` permettant à l'utilisateur de résoudre le casse-tête du tableau `tab` et retournant la longueur de la solution. Tant que le damier n'est pas ordonné, elle appelle itérativement `deClavierVersAction`, `effectuerAction`, `afficher`, et `fini`. **(3 points)**.

```
int resoudre(int * tab) {
  int action, l=0;
  do {
    action = deClavierVersAction();
    // action = deHasardVersAction();
    effectuerAction(tab, action);
    afficher(tab, HAUTEUR, LARGEUR);
    l++;
  } while (fini(tab, TAILLE)==0);

  printf("Bravo! Resolution en %d actions!\n", l);
  return 0;
}
```

12° Ecrire le `main` déclarant le tableau `tableau`, appelant `deHasardACasseTete`, `afficher`, `resoudre`. **(1point)**.

```
int main() {
  int tableau[TAILLE], longueur;
  printf("Bonjour.\n");
  deHasardACasseTete(tableau, TAILLE);
  printf("Voici le casse-tete:\n");
  afficher(tableau, HAUTEUR, LARGEUR);
  longueur = resoudre(tableau);
  printf("Au revoir.\n");
  return 0;
}
```

Exercice 2 (4 points)

Donner la sortie du programme. Tenir compte de la **couleur (bleue, rouge, verte ou jaune) de votre copie** pour initialiser `a` et `b`. avec les valeurs précisées en commentaires.

```
#include <stdio.h>
int main()   {    // Copie Bleue (B): faire a=2, copie Rouge (R): faire a=3.
  int a = 2;       // Copie Verte (V): faire a=4, copie Jaune (J): faire a=5.
  int * p = &a;
  int b = *p;                printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);
  a = 3;   /* R:4 V:5 J:2 */ printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);
  b += 4;  /* R:5 V:2 J: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);  }
```

Bleu :

```
1: a = 2, b = 2, *p = 2.
2: a = 3, b = 2, *p = 3.
3: a = 3, b = 6, *p = 3.
4: a = 3, b = 6, *p = 3, *q = 6.
5: a = 4, b = 9, *p = 4, *q = 9.
6: a = 5, b = 45, *p = 5, *q = 45.
7: a = 5, b = 45, *p = 45, *q = 45.
8: a = 5, b = 45, *p = 45, *q = 5.
```

Rouge :

```
1: a = 3, b = 3, *p = 3.
2: a = 4, b = 3, *p = 4.
3: a = 4, b = 8, *p = 4.
4: a = 4, b = 8, *p = 4, *q = 8.
5: a = 5, b = 12, *p = 5, *q = 12.
6: a = 6, b = 72, *p = 6, *q = 72.
7: a = 6, b = 72, *p = 72, *q = 72.
8: a = 6, b = 72, *p = 72, *q = 6.
```

Vert :

```
1: a = 4, b = 4, *p = 4.
2: a = 5, b = 4, *p = 5.
3: a = 5, b = 6, *p = 5.
4: a = 5, b = 6, *p = 5, *q = 6.
5: a = 6, b = 11, *p = 6, *q = 11.
6: a = 7, b = 77, *p = 7, *q = 77.
7: a = 7, b = 77, *p = 77, *q = 77.
8: a = 7, b = 77, *p = 77, *q = 7.
```

Jaune :

```
1: a = 5, b = 5, *p = 5.
2: a = 2, b = 5, *p = 2.
3: a = 2, b = 8, *p = 2.
4: a = 2, b = 8, *p = 2, *q = 8.
5: a = 3, b = 10, *p = 3, *q = 10.
6: a = 4, b = 40, *p = 4, *q = 40.
7: a = 4, b = 40, *p = 40, *q = 40.
8: a = 4, b = 40, *p = 40, *q = 4.
```

0.5 point par ligne correcte.