ECUE «Introduction à la programmation »

Contrôle continu n°3 (CC3) 7 janvier 2019
sans document - durée 1 heure 30

CORRIGE CORRIGE CORRIGE CORRIGE

Dans tous les exercices, les entrées clavier sont indiquées en caractères gras.

Exercice 1 (4 points)

Ecrire un programme affichant la lettre 'K' de taille `T`. `T` est entré au clavier par l'utilisateur. Si `T<2`, le programme affiche une erreur et retourne `-1`. Sinon, il affiche le 'K' avec des '`*`' et des '=', puis retourne `T`. L'exécution du programme respectera exactement les entrées-sorties ci-dessous. (On remarquera que le K est dessiné avec `2T-1` lignes de `2T` caractères).

```
T ? 4
*======*
*====*==
*==*====
**======
*==*====
*====*==
*======*
```

```
T ? 3
*====*
*==*==
**====
*==*==
*====*
```

```
T ? 2
*==*
**==
*==*
```

```
T ? 1
erreur: T<2.
```

```
#include <stdio.h>
int main() {
  int T, i, j;
  printf("T ? "); scanf("%d", &T);
  if (T<2) {
    printf("erreur: T < 2.\n");
    return -1;
  }
  // printf("T = %d\n", T);
  // 1 point pour ci-dessus
  // 3 points pour ci-dessous
  for (i=1; i<2*T; i++) {
    printf("*");
    if (i==T) printf("*");
    else      printf("=");
    int k = (i<T) ? T-i : i-T;
    for (j=1; j<T; j++) {
      if (j==k) printf("=*");
      else      printf("==");
    }
    printf("\n");
  }
  return T;
}
```

Exercice 2 (4 points)

Donner la sortie du programme ci-dessous. Pour chacune des trois lignes avec commentaire, tenir compte de la couleur (Violet, Saumon, Gris ou Jaune clair) de votre copie pour utiliser la valeur précisée dans le commentaire de la ligne à la place de la valeur donnée en gras dans la ligne.

```
#include <stdio.h>
int main() {
  int a = +5;                     // Violet:-3, Saumon:-3, Gris:+3, Jaune:+5
  int * p = &a;
  int b = *p;
  printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);
  a *= -3;                        // Violet:-4, Saumon:+5, Gris:+5, Jaune:-3
  printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);
  b += -4;                        // Violet:+7, Saumon:+3, Gris:-3, Jaune:-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);
}
```

Violet=bleu, saumon=rouge, gris=vert, jaune=jaune.

VIOLET :

```
1: a = -3, b = -3, *p = -3.
2: a = 12, b = -3, *p = 12.
3: a = 12, b = 4, *p = 12.
4: a = 12, b = 4, *p = 12, *q = 4.
5: a = 13, b = 16, *p = 13, *q = 16.
6: a = 14, b = 224, *p = 14, *q = 224.
7: a = 14, b = 224, *p = 224, *q = 224.
8: a = 14, b = 224, *p = 224, *q = 14.
```

SAUMON :

```
1: a = -3, b = -3, *p = -3.
2: a = -15, b = -3, *p = -15.
3: a = -15, b = 0, *p = -15.
4: a = -15, b = 0, *p = -15, *q = 0.
5: a = -14, b = -15, *p = -14, *q = -15.
6: a = -13, b = 195, *p = -13, *q = 195.
7: a = -13, b = 195, *p = 195, *q = 195.
8: a = -13, b = 195, *p = 195, *q = -13.
```

GRIS :

```
1: a = 3, b = 3, *p = 3.
2: a = 15, b = 3, *p = 15.
3: a = 15, b = 0, *p = 15.
4: a = 15, b = 0, *p = 15, *q = 0.
5: a = 16, b = 15, *p = 16, *q = 15.
6: a = 17, b = 255, *p = 17, *q = 255.
7: a = 17, b = 255, *p = 255, *q = 255.
8: a = 17, b = 255, *p = 255, *q = 17.
```

JAUNE :

```
1: a = 5, b = 5, *p = 5.
2: a = -15, b = 5, *p = -15.
3: a = -15, b = 1, *p = -15.
4: a = -15, b = 1, *p = -15, *q = 1.
5: a = -14, b = -14, *p = -14, *q = -14.
6: a = -13, b = 182, *p = -13, *q = 182.
7: a = -13, b = 182, *p = 182, *q = 182.
8: a = -13, b = 182, *p = 182, *q = -13.
```

Exercice 3 (12 points)

Une **carte** à jouer est représentée par un nombre entier `C` entre `0` et `51`. Sa **couleur** est le quotient de la division entière de `C` par `13` (entier entre `0` et `3`). Sa **hauteur** est le reste de la division entière de `C` par `13` (entier entre `0` et `12`).

1° Ecrire une procédure affichant une carte. L'affichage se fera sur 2 caractères. 1er caractère : la couleur de la carte ('`C`' pour `0`, '`D`' pour `1`, '`H`' pour `2`, '`S`' pour `3`). 2ème caractère : la hauteur de la carte (`h+2` pour h dans `{0, 1, 2, 3, …, 7}`, '`T`' pour `8`, '`J`' pour `9`, '`Q`' pour `10`, '`K`' pour `11`, '`A`' pour `12`). La signature de la procédure sera `void afficheCarte(int carte) ;` (Par exemple, si `carte` vaut `4`, la procédure affiche `C6` (Six de Trèfle[1]). Si `carte` vaut `38` elle affiche `HA` (As de Coeur). `DK` correspond au Roi de Carreau et vaut `24`.) **(1 point)**.

```
void afficheCarte(int carte)
{
  int c = carte/13, h = carte%13;
  char cc;
  if (c==0) cc = 'C';
  else if (c==1) cc = 'D';
  else if (c==2) cc = 'H';
  else if (c==3) cc = 'S';
  char ch;
  if (h<8) ch = 2 + h + '0';
  else if (h==8) ch = 'T';
  else if (h==9) ch = 'J';
  else if (h==10) ch = 'Q';
  else if (h==11) ch = 'K';
  else if (h==12) ch = 'A';
  printf("%c%c ", cc, ch);
}
```

2° Un **talon** de `52` cartes est représenté par un tableau de taille `52`. Ecrire une procédure initialisant un talon de `l` cartes avec des cartes ordonnées de `0` à `l-1`. Sa signature sera `void initialiseTalonOrdonne(int * tab, int l) ;` **(1 point)**.

```
void initialiseTalonOrdonne(int * tab, int l)
{
  for (int i=0; i<l; i++) *(tab+i) = i;
}
```

1C correspond à « Clubs » (Trèfle). D à « Diamonds » (Carreau). H à « Hearts » (Coeur), S à « Spades » (Pique), T à « Ten » (Dix), J à « Jack » (Valet), Q à « Queen » (Dame), K à « King » (Roi), A à « As ».

Une **main** de joueur contenant entre `0` et `13` cartes est représentée par un tableau de taille `13`. Chaque case du tableau correspond à une position de carte dans la main du joueur.

3° Ecrire une procédure affichant un tableau de cartes de taille `l`. L'affichage se fera sur une seule ligne et sera entre accolades. Sa signature sera `void afficheTableau(int * tab, int l) ;` Elle appellera `afficheCarte` autant de fois que nécessaire. **(1 point)**.

```
void afficheTableau(int * tab, int l)
{
  printf("{ ");
      for   (int   i=0;   i<l;   i++)   if   (*(tab+i)   !=   -1)
afficheCarte(*(tab+i));
  printf("}\n");
}
```

4° Etant donné un tableau `t1` de taille `l` en entrée, écrire une procédure remplissant un tableau `t2` de taille `l` mélangé de cartes. On utilisera des appels à `rand() % N` pour obtenir des nombres entiers aléatoires entre `0` et `N-1`. Sa signature sera `void melange(int * t1, int l, int * t2) ;` **(3 points)**.

```
void melange(int * t1, int l, int * t2)
{
  for (int i=0; i<l; i++) {
    int h = rand() % (l-i);
    int k = -1;
    for (int j=0; k<h; j++) {
      if (*(t1+j)==-1) continue;
      if (++k==h) {
     *(t2+i) = *(t1+j);
     *(t1+j) = -1;
      }
    }
  }
}
```

5° Etant donné un talon `t` de `l` cartes et 4 mains de joueurs, écrire une procédure qui distribue les `l` cartes du talon aux 4 mains. Sa signature sera `void distribue(int * t, int l, int * m1, int * m2, int * m3, int * m4) ;` **(1 point)**.

```
void distribue(int * t, int l, int * m1, int * m2, int * m3, int *
m4)
{
  for (int i=0; i<l; i++) {
    int j = i/4;
    switch (i%4) {
    case 0: *(m1+j) = *(t + i); break;
    case 1: *(m2+j) = *(t + i); break;
    case 2: *(m3+j) = *(t + i); break;
    case 3: *(m4+j) = *(t + i); break;
    }
  }
}
```

6° Etant donné un tableau `m` de `l` cartes, écrire une procédure triant `m` dans le sens croissant. La signature de la procédure sera `void tri(int * m, int l) ;` **(3 points)**.

```
void tri(int * m, int l)
{
  for (int n=0; n<l; n++) {
    for (int i=0; i<l-1; i++) {
      if (*(m+i)>*(m+i+1)) {
     int x = *(m+i);
     *(m+i) = *(m+i+1);
     *(m+i+1) = x;
      }
    }
  }
}
```

7° Ecrire un `main()` déclarant 6 tableaux : un talon ordonné, un talon mélangé, 4 mains, et appelant les procédures ci-dessus pour : initialiser le talon ordonné, mélanger les cartes dans le talon mélangé, afficher le talon mélangé, distribuer les cartes aux 4 joueurs, trier les 4 mains, les afficher. La sortie du programme aura la forme ci-dessous. **(2 points)**.

```
#include <stdio.h>
#include <stdlib.h>
// ici les fonctions des questions précédentes.
int main()
{
  printf("Bridge debut:\n");
  int talonOrdonne[52], talon[52];
  int main1[13], main2[13], main3[13], main4[13];

  srand(1);
  initialiseTalonOrdonne(talonOrdonne, 52);
  initialiseMainVide(main1, 13);
  initialiseMainVide(main2, 13);
  initialiseMainVide(main3, 13);
  initialiseMainVide(main4, 13);

  melange(talonOrdonne, 52, talon);
  printf("Talon mélangé:\n");
  afficheTableau(talon, 52);

  distribue(talon, 52, main1, main2, main3, main4);
  tri(main1, 13);
  tri(main2, 13);
  tri(main3, 13);
  tri(main4, 13);
  printf("Mains triées:\n");
  afficheTableau(main1, 13);
  afficheTableau(main2, 13);
  afficheTableau(main3, 13);
  afficheTableau(main4, 13);

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

```
Bridge debut:
Talon mélangé:
{ S2 SK H3 C7 D7 H5 D4 D2 H7 H6 C4 CT D3 C8 C3 C6 HA C5 D5 S3 S9 C9 SJ HK
CQ DJ CK SQ S6 D8 SA C2 D6 DA DK ST CA D9 HQ S7 S4 S8 DT H2 CJ H9 H8 S5
HJ DQ HT H4 }
Mains triées:
{ CJ CQ CA D3 D6 D7 H7 HJ HA S2 S4 S6 S9 }
{ C5 C8 C9 D8 D9 DJ DQ DA H5 H6 H9 S8 SK }
{ C3 C4 CK D4 D5 DT DK H3 H8 HT HQ SJ SA }
{ C2 C6 C7 CT D2 H2 H4 HK S3 S5 S7 ST SQ }
Bridge fin.
```