ECUE «Prog 1» - CC3
10 janvier 2018 - Bruno Bouzy
sans document - durée 1 heure 30

Dans les exécutions de programmes, les entrées clavier sont en **gras** et les sorties en police normale.

CORRIGE CORRIGE CORRIGE CORRIGE

Exercice 1 (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 = +3;                    // Violet:-3, Saumon:-2, Gris:+1, Jaune:+3
  int * p = &a;
  int b = *p;
  printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);

  a *= -3;                       // Violet:-2, Saumon:+5, Gris:+5, Jaune:-3
  printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);

  b += -2;                       // Violet:+2, Saumon:+3, Gris:-1, Jaune:-2
  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 :

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

SAUMON :

```
1: a = -2, b = -2, *p = -2.
2: a = -10, b = -2, *p = -10.
3: a = -10, b = 1, *p = -10.
4: a = -10, b = 1, *p = -10, *q = 1.
5: a = -9, b = -9, *p = -9, *q = -9.
6: a = -8, b = 72, *p = -8, *q = 72.
7: a = -8, b = 72, *p = 72, *q = 72.
8: a = -8, b = 72, *p = 72, *q = -8.
```

GRIS :

```
1: a = 1, b = 1, *p = 1.
2: a = 5, b = 1, *p = 5.
3: a = 5, b = 0, *p = 5.
4: a = 5, b = 0, *p = 5, *q = 0.
5: a = 6, b = 5, *p = 6, *q = 5.
6: a = 7, b = 35, *p = 7, *q = 35.
7: a = 7, b = 35, *p = 35, *q = 35.
8: a = 7, b = 35, *p = 35, *q = 7.
```

JAUNE :

```
1: a = 3, b = 3, *p = 3.
2: a = -9, b = 3, *p = -9.
3: a = -9, b = 1, *p = -9.
4: a = -9, b = 1, *p = -9, *q = 1.
5: a = -8, b = -8, *p = -8, *q = -8.
6: a = -7, b = 56, *p = -7, *q = 56.
7: a = -7, b = 56, *p = 56, *q = 56.
8: a = -7, b = 56, *p = 56, *q = -7.
```

0.5 point par ligne correcte.

Notation dure : si une ligne est fausse, le correcteur peut compter 0 pour les lignes suivantes.

Exercice 2 (3 points)

Ecrire un programme affichant la lettre 'Z' 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 'Z' avec des `*` séparées par des `=`, puis retourne `T`. L'exécution du programme respectera les entrées-sorties ci-dessous. (On remarquera que `T` est égal au nombre de `*` sur chacune des 2 lignes horizontales du 'Z').

```
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);

  // 0.5 point
  for (j=1; j<T; j++) printf("*=");
  printf("*\n");

  // 2 points
  for (i=2; i<2*T-1; i++) {
    for (j=1; j<=2*T-i-1; j++) printf("=");
    printf("*");
    for (j=1; j<i; j++) printf("=");
    printf("\n");
  }

  // 0.5 point
  for (j=1; j<T; j++) printf("*=");
  printf("*\n");

  return T;
}
```

Exercice 3 (6 points)

1) Ecrire une fonction `int diane(int a, int b)` demandant à l'utilisateur un nombre entier dans l'intervalle `[a,b]` et retournant ce nombre. La fonction demande répétitivement le nombre à l'utilisateur tant que le nombre tapé au clavier n'appartient pas à `[a,b]`. **(2 pts)**

```
// 2 points
int diane(int a, int b)  {
  int x;
  do {
    printf (" x ? (entre %d et %d) ", a, b);
    scanf("%d", &x);
  } while (x>b||x<a);
  return x;
}
```

2) Ecrire une procédure `void correze` prenant en entrée deux nombres entiers, `m` et `n`, et deux nombres en sortie: le quotient `q` et le reste `r` de la division entière de `m` par `n`. **(2 pts)**

```
// 2 points
void correze(int m, int n, int * q, int * r)  {
  *q = m/n;
  *r = m%n;
}
```

3) Ecrire un programme `main` demandant un entier `c` dans l'intervalle `[0,99]`et un entier `d` dans l'intervalle `[2,9]`, puis affichant le quotient et le reste de la division entière de `c` par `d`. On utilisera la fonction `diane` et la procédure `correze` des questions 1 et 2. Le programme respectera les entrées sorties des exécutions suivantes. **(2 pts)**

```
Demande de c:
      x ? (0<=x<=99) 123
      x ? (0<=x<=99) -1
      x ? (0<=x<=99) 45
c = 45
Demande de d:
      x ? (2<=x<=9) 1
      x ? (2<=x<=9) 6
d = 6
45/6 = 7 reste 3
```

```
Demande de c:
      x ? (0<=x<=99) 60
c = 60
Demande de d:
      x ? (2<=x<=9) 7
d = 7
60/7 = 8 reste 4
```

```
// 2 points
int main() {
  int c, d, q, r;
  printf("Demande de c:\n"); c = diane(0, 99);
  printf("c = %d\n", c);
  printf("Demande de d:\n"); d = diane(2, 9);
  printf("d = %d\n", d);
  correze(c, d, &q, &r);
  printf("%d/%d = %d reste %d\n", c, d, q, r);
  return 0;
}
```

Exercice 4 (7 points)

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

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

1) a) Quelles sont les valeurs de A_1 et B_1 ? **(0.25 pt)**

$A_1 = 5$ et $B_1 = 3$

1) b) Donner la sortie de: **(0.5 pt)**

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

```
a1 = 5.00, b1 = 6.71
donner le demi point si a1 est 5.00 et si la valeur de b1 est entre 6.01 et 6.99.
```

1) c) Cette sortie est-elle compatible avec la définition (2) ? **(0.25 pt)**

non.

1) d) Modifier le traitement de la question 1) b) pour qu'il corresponde à la définition (2). **(1 pt)**

```
float a=1,b=9;    float a2=(a+b)/2;  b=sqrt(a*b);   a=a2;
```

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

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

1 point pour la déclaration (nom + liste type paramètre *) correcte.
1 point pour le corps de la fonction.

3) Ecrire une fonction `void suite(int n)` affichant les termes des suites A_n et B_n avec 6 chiffres après la virgule pour les indices allant de `0` à `n` compris. On utilisera `deABaAB`. **(2 pts)**

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

1 point pour l'appel correct à deABaAB.
0.5 point pour la déclaration correcte de la fonction.
0.5 point pour les printf corrects et la boucle correcte.

4) Ecrire un programme `main` appelant `suite` avec `n=4`. **(0.5 pt)**

```
#include <stdio.h>
#include <math.h>
#define N_ITERATIONS  4
#define A_0  1
#define B_0  9
// les fonctions ici.
int main()
{
  suite(N_ITERATIONS);
  return 0;
}
```

5) Quelle est l'option à utiliser pour compiler ce programme avec succès ? **(0.5 pt)**

```
-lm
```