

Examen du 12/11/2009

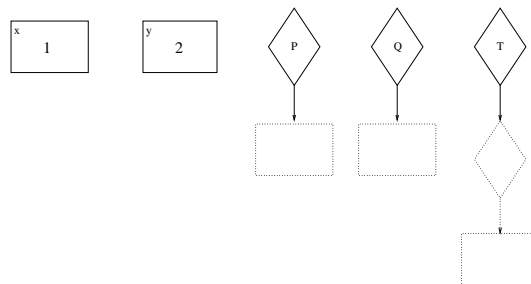
Corrigé

Exercice I

Après l'exécution de

```
int x=1, y=2;  
int *P, *Q, **T;
```

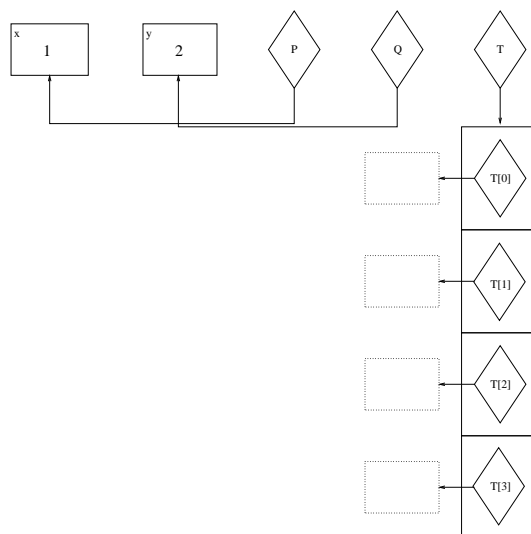
l'état de la mémoire est le suivant



On exécute alors

```
P = &x;  
Q = &y;  
T = (int**) malloc(4*sizeof(int*));
```

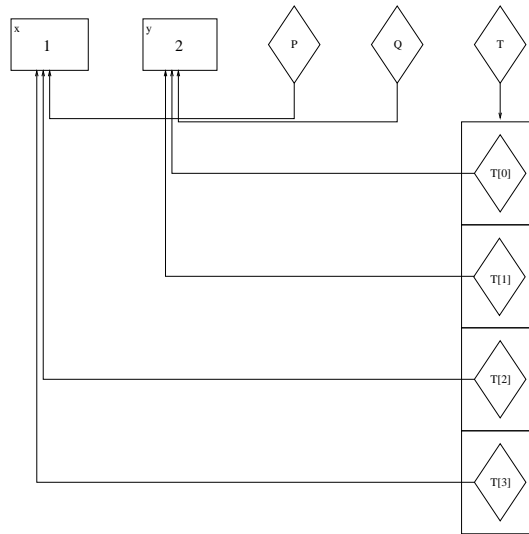
l'état de la mémoire est le suivant



On exécute alors

```
T[0] = T[1] = Q;  
T[2] = T[3] = P;
```

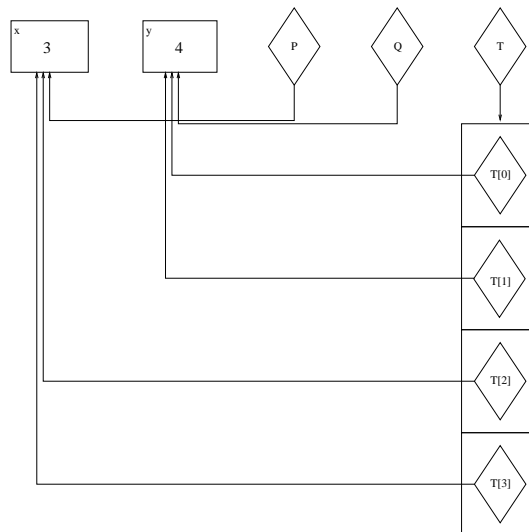
l'état de la mémoire est le suivant



On exécute alors

```
x = 3;  
y = 4;
```

l'état de la mémoire est le suivant



On exécute alors

```
printf("[1] %d %d %d %d\n", x, y, *P, *Q);  
afficher(2, T);
```

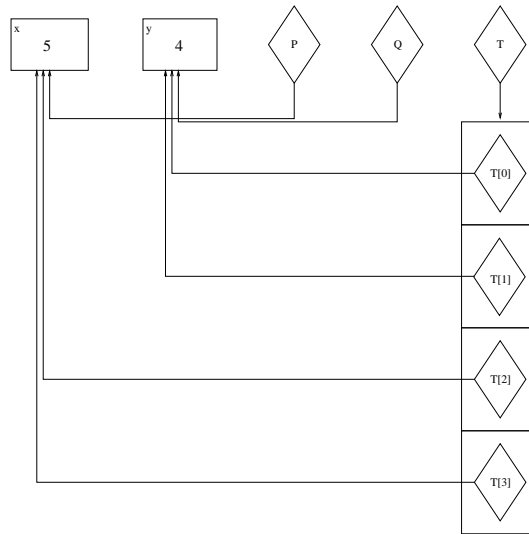
ce qui affiche à l'écran

```
[1] 3 4 3 4  
[2] 4 4 3 3
```

On exécute alors

```
*P = 5;
```

l'état de la mémoire est le suivant



On exécute alors

```
afficher(3,T);
```

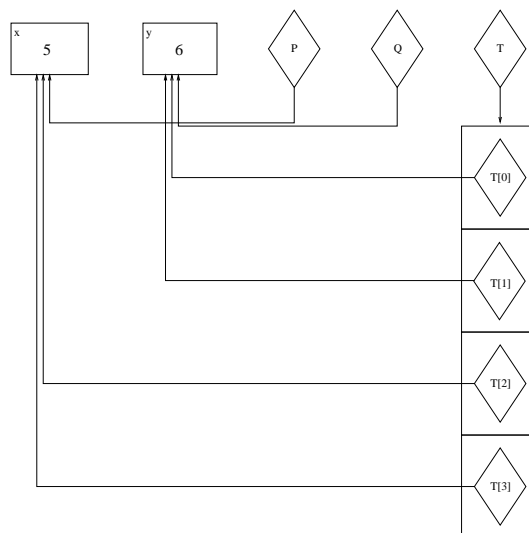
ce qui affiche à l'écran

```
[3] 4 4 5 5
```

On exécute alors

```
y = 6;
```

l'état de la mémoire est le suivant



On exécute alors

```
afficher(4,T);
```

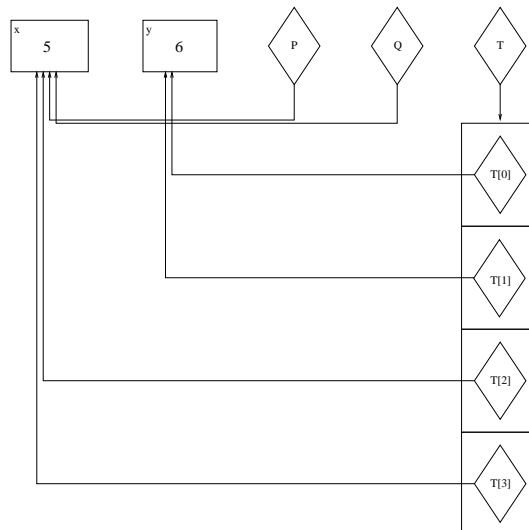
ce qui affiche à l'écran

```
[4] 6 6 5 5
```

On exécute alors

```
Q = &x;
```

l'état de la mémoire est le suivant



On exécute alors

```
afficher(5,T);  
printf("[6] %d %d %d %d\n",x,y,*P,*Q);
```

ce qui affiche à l'écran

```
[5] 6 6 5 5  
[6] 5 6 5 5
```

Exercice II

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

```
#define NbEssais 1000000
```

```
int aleatoire (int max) {  
    return(rand()%max+1);  
}
```

```
int experience_reussie() {  
    /* Retourne 1 si l'expérience est reussie et 0 sinon */
```

```
    int i, tirage;
```

```
    int urne = aleatoire(6);
```

```
    int boules_blanches = urne*urne;  
    int boules_noires = urne*urne*urne;  
    int boules_rouges = urne*urne*urne*urne;
```

```

int boules = boules_blanches + boules_noires + boules_rouges;

int boules_rouges_tirees = 0;

/*
Convention :
Les boules rouges sont numerotees de 1 a boules_rouges
Les boules noires sont numerotees de boules_rouges+1 a boules_rouges+boules_noires
Les boules blanches sont numerotees de boules_rouges+boules_noires+1 a boules
*/

for (i=0;i<=2;i++) {
    tirage = aleatoire(boules);
    if (tirage<=boules_rouges) boules_rouges_tirees++;
}

return (boules_rouges_tirees==3);
}

int main() {
    int i;
    int reussi = 0;

    for (i=1;i<=NbEssais;i++)
        if (experience_reussie()) reussi++;

    printf("Probabilite : %f\n",(float)reussi/NbEssais);

    return 0;
}

```

Exercice III

1. On considère le type suivant

```

typedef struct Suite_struct {
    double a, b, u0, u1; }
Suite;

```

et la fonction suivante

```

Suite CreerSuite (double a, double b, double u0, double u1)
/* ENTREE : les coefficients a, b et les conditions initiales u0, u1
   SORTIE : la suite correspondante */
{
    Suite u;

    /* Verification de la condition  $a*a+4*b>0$  */
    assert(a*a+4*b>0);

    /* Construction de la suite */

```

```

    u.a = a;
    u.b = b;
    u.u0= u0;
    u.u1= u1;

    return u;
}

```

2.

```

Suite SommeSuiteParticulier (Suite u, Suite v)
/* ENTREE : deux suites avec les memes coefficients a et b
   SORTIE : leur somme */
{
    Suite w;

    assert((u.a==v.a)&&(u.b==v.b));

    w.a = u.a;
    w.b = u.b;
    w.u0= u.u0 + v.u0;
    w.u1= u.u1 + v.u1;

    return w;
}

```

3. La somme de deux suites de type `Suite` n'est pas forcément de ce type (ce qui peut aussi se dire de la manière suivante : l'addition n'est pas une loi interne de l'ensemble de ces suites). Pour le montrer, il suffit de prendre un contre-exemple et de raisonner par l'absurde.

Soit $(u_n)_{n \in \mathbb{N}}$ et $(v_n)_{n \in \mathbb{N}}$ définies par

$$\begin{cases} u_0 = 1 \\ u_1 = 2 \\ u_{n+1} = 2 u_n \end{cases}$$

ainsi $a = 2$ et $b = 0$ ce qui simplifiera les calculs, et

$$\begin{cases} v_0 = 1 \\ v_1 = 3 \\ v_{n+1} = 3 v_n \end{cases}$$

ce qui correspond à $a = 3$ et $b = 0$.

La somme de ces suites, notée $(w_n)_{n \in \mathbb{N}}$ s'écrit alors

$$w_n = 2^n + 3^n$$

puisque les suites $(u_n)_{n \in \mathbb{N}}$ et $(v_n)_{n \in \mathbb{N}}$ sont des suites géométriques dont il est facile d'obtenir le terme général.

Supposons que $(w_n)_{n \in \mathbb{N}}$ soit de type `Suite` alors il existe deux réels (constants) a et b tels que $w_{n+1} = aw_n + bw_{n-1}$ pour tout entier naturel $n \geq 1$. Donc,

$$2^{n+1} + 3^{n+1} = 2^n a + 3^n a + 2^{n-1} a + 2^{n-1} b + 3^{n-1} b$$

Comme cette égalité est vraie pour tout entier naturel $n \geq 1$, elle est vraie pour $n = 1$, $n = 2$ et $n = 3$.

Pour $n = 1$, on obtient $4 + 9 = 2a + 3a + b + b$, ce qui équivaut à

$$5a + 2b = 13$$

Pour $n = 2$, on obtient $8 + 27 = 4a + 9a + 2b + 3b$, ce qui équivaut à

$$13a + 5b = 35$$

Le système

$$\begin{cases} 5a + 2b = 13 \\ 13a + 5b = 35 \end{cases}$$

est équivalent à $a = 5$ et $b = -6$. on a alors

$$w_{n+1} = 5w_n - 6w_{n-1}$$

donc $w_4 = 5w_3 - 6w_2$ ainsi

$$2^4 + 3^4 = 5 \times (2^3 + 3^3) - 6 \times (2^2 + 3^2)$$

donc $97 = 27$ **ce qui est faux.**

4.

```
double TermeSuite (Suite u, int n)
/* ENTREE : une suite u et un entier naturel n
   SORTIE : u(n) */
{
  assert (n>=0);

  if (n==0) return u.u0;
  else if (n==1) return u.u1;
  else {

    int p = 1;
    int aux1 = u.u0; /* contient u_{p-1} */
    int aux2 = u.u1; /* contient u_p */
    int suivant;    /* contiendra u_{p+1} */

    while (p<n) {
      /* u_{p+1} = u_p + u_{p-1} */
      suivant = aux1 + aux2;

      /* On incremente p de 1 donc on decale */
      p++;
      aux1 = aux2;
      aux2 = suivant;
    }

    return aux2; /* puisque aux2 contient u_p */
  }
}
```

5. Le discriminant du trinôme $X^2 - aX - b$ est $\Delta = a^2 + 4b$ ce qui est strictement positif, par hypothèses. Ainsi le trinôme admet deux racines réelles distinctes r_1 et r_2 .

Considérons (H_n) l'hypothèse $u_n = Ar_1^n + Br_2^n$.

- (H_0) est vraie si l'on pose $A + B = u_0$, ce que l'on fait.
- (H_1) est vraie si l'on pose $Ar_1 + Br_2 = u_1$, ce que l'on peut faire car $r_1 \neq r_2$ entraîne la non-nullité du déterminant

$$\begin{vmatrix} A & B \\ Ar_1 & Br_2 \end{vmatrix}$$

- Supposons (H_n) vraie et $n \geq 1$ alors

$$u_n = Ar_1^n + Br_2^n$$

or $u_{n+1} = au_n + bu_{n-1}$ donc

$$u_{n+1} = a(Ar_1^n + Br_2^n) + b(Ar_1^{n-1} + Br_2^{n-1})$$

ainsi

$$u_{n+1} = (ar_1 + b)Ar_1^{n-1} + (ar_2 + b)Br_2^{n-1}$$

mais r_1 est solution de $X^2 - aX - b = 0$ donc $ar_1 + b = r_1^2$, de même pour r_2 donc $ar_2 + b = r_2^2$; il vient donc

$$u_{n+1} = Ar_1^{n+1} + Br_2^{n+1}$$

ce qui donne (H_{n+1}) .

xs Le prédicat de convergence de la suite s'écrit simplement en vérifiant que r_1 et r_2 sont tels que $1 < r_1 \leq 1$ et $-1 < r_2 \leq 1$. En cas d'indétermination, celle-ci se lève en factorisant la valeur absolue la plus grande, on montre alors la divergence sans difficulté.

```
void r1r2 (Suite u, double* r1_P, double* r2_P)
/* ENTREE : Une suite u (premier argument)
   SORTIE : les racines du trinome associe (pointeurs vers) via les arguments 2 et 4
*/
{
    double delta = u.a*u.a+4*u.b;
    double sqrtdelta = sqrt(delta);
    *r1_P = (u.a-sqrtdelta)/2;
    *r2_P = (u.a+sqrtdelta)/2;
}

int estConvergente (Suite u)
/* Predicat de convergence */
{
    double r1, r2;
    r1r2(u,&r1,&r2);
    return ((r1>-1)&&(r1<=1)&&(r2>-1)&&(r2<=1));
}
```

6. Dans suite.h on met:

```
typedef struct Suite_struct {
    double a, b, u0, u1; }
Suite;

Suite CreerSuite (double a, double b, double u0, double u1);
Suite SommeSuiteParticulier (Suite u, Suite v);
double TermeSuite (Suite u, int n);
int estConvergente (Suite u);
```

Dans suite.c on met

```
#include <stdio.h>
#include <math.h>
#include <assert.h>
#include "suite.h"

Suite CreerSuite (double a, double b, double u0, double u1)
/* ENTREE : les coefficients a, b et les conditions initiales u0, u1
   SORTIE : la suite correspondante */
{
...

TOUTES LES FONCTIONS DEFINIES CI DESSUS

...
double r1, r2;
r1r2(u,&r1,&r2);
return ((fabs(r1)<=1)&&(fabs(r2)<=1));
}

7.

#include <stdio.h>
#include <math.h>
#include <assert.h>
#include "suite.h"

int main() {
Suite u1, u2, u;

u1 = CreerSuite(1,1,0.2,0.8);
u2 = CreerSuite(1,1,0.8,0.2);
u = SommeSuiteParticulier(u1,u2); /* u est maintenant la suite de Fibonacci */

printf("u(5)=%f\n",TermeSuite(u,5));
if (estConvergente(u)) printf("Cette suite converge\n");
else printf("Cette suite diverge\n");

return 0;
}
```