

Examen du 15/11/2005

Corrigé

Exercice I

1. Le programme est incorrect, il y a une erreur de type : `*n` ne veut rien dire puisque `n` n'est pas un pointeur.
2. Le programme est correct, il affiche l'entier pointé par `P`, c'est-à-dire `n` donc 1.
3. Le programme est incorrect car on demande à l'ordinateur d'afficher ce qui est pointé par `Q`, or ce pointeur a été initialisé à `P` avant que `P` ne pointe `n` (il pointait donc vers n'importe quoi). Par suite `Q` pointe vers n'importe quoi.
4. Le programme affichera l'adresse du pointeur c'est-à-dire quelque chose d'inintelligible.
5. Le programme est correct, il affiche l'entier pointé par le pointeur pointé par `T`, c'est-à-dire l'entier pointé par `P`, c'est-à-dire `n` donc 1.

Exercice II

Le programme suivant répond aux questions 2, 3 et 5. On traite les deux questions théoriques ensuite.

```
#include <stdio.h>

/* Question 2 */

typedef struct MSR22_Struct { double a, b, c; } MSR22;

/* Question 3 */

void LireMSR22(MSR22* M_Ptr)
    /* Lecture d'une matrice reelle symetrique 2x2
       ENTREE : lecture au clavier des coefficients
       SORTIE : dans l'argument de la fonction, un pointeur sur la matrice
    */
{
    printf("Lecture de la matrice reelle symetrique 2x2\n");
    printf("[ a b ]\n");
    printf("[ b c ]\n");
    printf("Entrer a : \n"); scanf("%lf",&((*M_Ptr).a));
    printf("Entrer b : \n"); scanf("%lf",&((*M_Ptr).b));
    printf("Entrer c : \n"); scanf("%lf",&((*M_Ptr).c));
}
```

```

void AfficherMSR22(MSR22 M)
    /* Affichage d'une matrice reelle symetrique 2x2
       ENTREE : la matrice M
       SORTIE : a l'ecran, affichage de M
    */
{
    printf("[ %10.5lf %10.5lf ]\n",M.a,M.b);
    printf("[ %10.5lf %10.5lf ]\n",M.b,M.c);
}

MSR22 SommeMSR22 (MSR22 M1, MSR22 M2)
    /* Somme de deux matrices reelles symetriques 2x2
       ENTREE : les deux matrices M1 et M2
       SORTIE : M1+M2
    */
{
    MSR22 resultat;
    resultat.a=M1.a+M2.a;
    resultat.b=M1.b+M2.b;
    resultat.c=M1.c+M2.c;
    return(resultat);
}

MSR22 ProdExtMSR22 (MSR22 M, double lambda)
    /* Produit d'une matrice reelle symetrique 2x2 par un scalaire
       ENTREE : la matrice M et le scalaire lambda
       SORTIE : lambda.M
    */
{
    MSR22 resultat;
    resultat.a=lambda*M.a;
    resultat.b=lambda*M.b;
    resultat.c=lambda*M.c;
    return(resultat);
}

/* Question 5 */

int main()
{
    MSR22 M1, M2, S, P;

    LireMSR22(&M1);
    LireMSR22(&M2);

    S=SommeMSR22(M1,M2);
    P=ProdExtMSR22(S,3);

    printf("La somme des deux matrices entrees est\n");
    AfficherMSR22(S);

    printf("3 fois cette somme est\n");
    AfficherMSR22(P);
}

```

```

return (0);
}

```

1. Pour démontrer que l'ensemble E des matrices symétriques réelles 2×2 est un \mathbb{R} -espace vectoriel on montre que c'est un sous-espace vectoriel de l'ensemble des matrices réelles carrées 2×2 .

- E est non vide puisqu'il contient la matrice nulle
- Si

$$M = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \in E \text{ et } M' = \begin{pmatrix} a' & b' \\ b' & c' \end{pmatrix} \in E$$

et si λ est un réel alors

$$M + \lambda M' = \begin{pmatrix} a + \lambda a' & b + \lambda b' \\ b + \lambda b' & c + \lambda c' \end{pmatrix}$$

ce qui est un élément de E .

5. Le produit de deux matrices symétriques peut ne pas être une matrice symétrique comme le montre le contre exemple suivant

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, M' = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}. \quad MM' = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$$

Il n'est donc pas possible de faire la fonction `ProduitMSR22`.

Exercice III

1. Il s'agit d'une série de Riemann. Cette série est convergente si et seulement si $s > 1$.
2. La fonction suivante convient

```

double Zeta (double s)
/* Fonction Zeta de Riemann
Necessite la bibliotheque math.h
ENTREE : un reel s
SORTIE : Zeta(s) si s>1, le reel -1 pour signaler une erreur sinon
*/
{
double res;
int n;

if (s>1) {
res = 0;
for (n=1;n<=1000;n++)
res+=1/pow(n,s);
return(res);
}
else
return(-1);
}

```

QS. Soit $A \geq 2$ un entier, on a¹

$$\forall n \in \llbracket 1, A \rrbracket, \forall x \in [n-1, n], \frac{1}{n^s} \leq \frac{1}{x^s}$$

donc

$$\forall n \in \llbracket 1, A \rrbracket, \frac{1}{n^s} \leq \int_{n-1}^n \frac{1}{x^s}$$

donc

$$\sum_{n=N+1}^A \frac{1}{n^s} \leq \sum_{n=N+1}^A \int_{n-1}^n \frac{1}{x^s}$$

donc

$$\sum_{n=N+1}^A \frac{1}{n^s} \leq \int_N^A \frac{1}{x^s}$$

donc

$$\sum_{n=N+1}^A \frac{1}{n^s} \leq \frac{1}{1-s} \left(\frac{1}{A^{s-1}} - \frac{1}{N^{s-1}} \right)$$

lorsque A tend vers $+\infty$, il vient

$$\sum_{n=N+1}^{+\infty} \frac{1}{n^s} \leq \frac{1}{s-1} \frac{1}{N^{s-1}}$$

l'erreur commise en approximant

$$\sum_{n=1}^{+\infty} \frac{1}{n^s} \quad \text{par} \quad \sum_{n=1}^N \frac{1}{n^s}$$

est donc inférieur à $\frac{1}{1-s} \frac{1}{N^{s-1}}$. Pour retourner un résultat dont la précision est au moins de p (c'est-à-dire telle que l'erreur commise soit inférieure à p), il suffit de choisir un N tel que

$$\frac{1}{s-1} \frac{1}{N^{s-1}} \leq p$$

Comme $s > 1$, cette équation équivaut à

$$N^{s-1} \geq \frac{1}{(s-1)p}$$

c'est-à-dire

$$N = \text{E} \left(\left(\frac{1}{(s-1)p} \right)^{\frac{1}{s-1}} \right) + 1$$

Au moyen de cette étude on propose le programme suivant :

```
double ZetaPrecise (double s, double p)
/* Fonction Zeta de Riemann
Necessite la bibliotheque math.h
ENTREE : un reel s, la precision p
SORTIE : Zeta(s) si s>1, le reel -1 pour signaler une erreur sinon
*/
{
```

¹On remarque que cette inégalité est conforme à l'interprétation géométrique selon laquelle l'aire comprise entre la courbe, l'axe des abscisses et les droites d'équation $x = n$ et $x = n + 1$ est supérieure à l'aire du rectangle dont les sommets sont $(n-1; 0)$, $(n; 0)$, $(n-1; 1/n^s)$, $(n-1; 1/n^s)$.

```
double res;
int n, N;

if (s>1) {
    N = floor(pow(1/(p*(s-1)),1/(s-1)))+1;
    printf("N=%d\n",N);
    res = 0;
    for (n=1;n<=N;n++)
        res+=1/pow(n,s);
    return(res);
}
else
    return(-1);
}
```