TD Corrigés : pointeurs

Ces corrigés sont écrits avec les entrées-sorties du *C* et non celles du *C++* : il faut donc remplacer partout les `printf` par des `cout`, les `scanf` par des `cin` et les instructions concernant les fichiers par celles qui leurs correspondent en *C++*. Il faut également ajouter en tête des programmes :

```
#include<iostream>
#include<iomanip>
#include<fstream>
using namespace std;
```

```
*****************************************************************************************
//passage_par_adresse_1.c
#include<stdio.h>
#include<math.h>
int racine(double a,double b,double *x0)
{
  if(a==0) return 0;
  *x0=-b/a;
  return(1);
}
int main()
{
  double p,q,x;
  p=5; q=3;
  if(racine(p,q,&x)==0)
  printf("L'equation %lgx+%lg=0 a zero ou une infinite de racines\n",p,q);
  else if(racine(p,q,&x)==1) printf("L'equation %lgx+%lg=0 a pour racine %lg\n",p,q,x);
  else printf("Resultat anormal\n");
  return 0;
}
*****************************************************************************************
//passage_par_adresse_2.c
#include <stdio.h>
#include <math.h>
void f(double a,double b,double *u)
{
  *u=a*(*u)+b;
}
int main()
{
  int i,N=8;
  double x=3,y=7,un=1;
  for(i=0;i<N;i++)
    {
      printf("i=%d un=%lg\n",i,un);
      f(x,y,&un);
    }
  return 0;
}
*****************************************************************************************
//vecteur_unitaire.c
#include <stdio.h>
#include <math.h>
#define N 2
#define P 3
int f(const double v[],double u[],int n)
```

```
{
  double c,q;
  int i;
  for(c=0,i=0;i<n;i++) c=c+v[i]*v[i];
  q=sqrt(c);
  if(q==0)
    {
      for(i=0;i<n;i++) u[i]=0;
      return 0;
    }
  for(i=0;i<n;i++) u[i]=v[i]/q;
  return(1);
}
int main()
{
  int i;
  double x[N]={1.,2.},xu[N],y[P]={3.,4.,5.},yu[P],z[P]={0.,0.,0.},zu[P];
  printf("f=%d ",f(x,xu,N)); for(i=0; i<N; i++) printf("%lg ",xu[i]);
  printf("\n");
  printf("f=%d ",f(y,yu,P)); for(i=0; i<P; i++) printf("%lg ",yu[i]);
  printf("\n");
  printf("f=%d ",f(z,zu,P)); for(i=0; i<P; i++) printf("%lg ",zu[i]);
  printf("\n");
  return 0;
}
*****************************************************************************
//echantillonnage.c
#include<stdio.h>
#include<math.h>
#define NP 100
double g(double x)
{
  return exp(-x*x/2);
}

void ech(double(*f)(double),double xmin,double xmax,int n,double t[])
{
  double h;
  int i;
  h=(xmax-xmin)/(n-1);
  for(i=0;i<n;i++) t[i]=(*f)(h*i+xmin);
}

int main()
{
int i;
double min=-5,max=5,tab[NP];
FILE *fp;
fp=fopen("echantillonage.res","w");
ech(g,min,max,NP,tab);
for(i=0;i<NP;i++) fprintf(fp,"%lg\n",tab[i]);
return 0;
}
*****************************************************************************
// produit_matrices_tableaux.cpp
// Ce programme ne peut multiplier que des matrices NxM par MxP, les valeurs de M, N et P
// ne pouvant varier dans une même exécution puisque ce sont des constantes
```

```
#include<iostream>
#include<fstream>
using namespace std;
#define N 3
#define M 2
#define P 1
void f(double a[M][N],double b[N][P],double c[M][P])
{
  int i,j,k;
  for(i=0;i<M;i++)
    {
      for(j=0;j<P;j++)
{
  for(c[i][j]=0,k=0;k<N;k++)
    {
      c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }
}
    }
}
int main()
{
  int i,j;
  double a[M][N],b[N][P],c[M][P];
  // On lit les éléments de matrice dans le fichier matrices.dat
  fstream dat("matrices.dat",ios::in);
  for(i=0;i<M;i++) for(j=0;j<N;j++) dat >> a[i][j];
  for(i=0;i<N;i++) for(j=0;j<P;j++) dat >> b[i][j];
  // On affiche ces éléments pour vérification
  for(i=0;i<M;i++) {for(j=0;j<N;j++) {cout << a[i][j] << " ";} cout << endl;}
  cout << endl;
  for(i=0;i<N;i++) {for(j=0;j<P;j++) {cout << b[i][j] << " ";} cout << endl;}
  cout << endl;
  // On calcule le produit
  f(a,b,c);
  // On affiche le résultat du produit
  for(i=0;i<M;i++)
    {
      for(j=0;j<P;j++)
{
  cout << c[i][j] << " ";
}
      cout << endl;
    }
  return 0;
}
*****************************************************************************
// produit_matrices_tableaux_pointeurs.cpp
// Ce programme peut multiplier successivement des matrices quelles qu'elles soient
// dans une même exécution puisque m, n et p sont des variables
#include<iostream>
#include<fstream>
#include<iomanip>
#include<bibli_fonctions.h>
using namespace std;
void f (double **a,double **b,double **c,int m,int n,int p)
{
```

```
  int i,j,k;
  for(i=0;i<m;i++)
    {
      for(j=0;j<p;j++)
{
  for(c[i][j]=0,k=0;k<n;k++)
    {
      c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }
}
    }
}
int main()
{
  int i,j,m,n,p;
  double **a,**b,**c;
  fstream dat("matrices.dat",ios::in);
  //-------------------------------------------------------------------------
  // On traite un premier couple de matrices
  m=2; n=3; p=1;
  a=D_2(m,n); b=D_2(n,p); c=D_2(m,p);
  // On lit les valeurs des éléments de matrice dans le fichier matrices.dat
  for(i=0;i<m;i++) for(j=0;j<n;j++) dat >> a[i][j];
  for(i=0;i<n;i++) for(j=0;j<p;j++) dat >> b[i][j];
  // On affiche les matrices pour vérification
  cout << "Premier couple de matrices" << endl << endl;
  for(i=0;i<m;i++) {for(j=0;j<n;j++) {cout << setw(3) << a[i][j] << " ";} cout << endl;}
  cout << endl;
  for(i=0;i<n;i++) {for(j=0;j<p;j++) {cout << setw(3) << b[i][j] << " ";} cout << endl;}
  cout << endl;
  // On calcule le produit
  f(a,b,c,m,n,p);
  // On affiche la matrice produit
  for (i=0;i<m;i++)
    {
      for (j=0;j<p;j++)
{
  cout << setw(3) << c[i][j] << " ";
}
      cout << endl;
    }
  // On libère la mémoire allouée pour les tableaux-pointeurs
  f_D_2(a,m,n); f_D_2(b,n,p); f_D_2(c,m,p);
  cout << endl << endl;
  //-------------------------------------------------------------------------
  // On traite un second couple de matrices de dimensions différentes des précédentes
  m=3; n=4; p=2;
  a=D_2(m,n); b=D_2(n,p); c=D_2(m,p);
  // On lit les valeurs des éléments de matrice dans le fichier matrices.dat
  for(i=0;i<m;i++) for(j=0;j<n;j++) dat >> a[i][j];
  for(i=0;i<n;i++) for(j=0;j<p;j++) dat >> b[i][j];
  // On affiche les matrices pour vérification
  cout << "Second couple de matrices" << endl << endl;
  for(i=0;i<m;i++) {for(j=0;j<n;j++) {cout << setw(3) << a[i][j] << " ";} cout << endl;}
  cout << endl;
  for(i=0;i<n;i++) {for(j=0;j<p;j++) {cout << setw(3) << b[i][j] << " ";} cout << endl;}
  cout << endl;
```

```
  // On calcule le produit
  f(a,b,c,m,n,p);
  // On affiche la matrice produit
  for (i=0;i<m;i++)
    {
      for (j=0;j<p;j++)
{
  cout << setw(3) << c[i][j] << " ";
}
      cout << endl;
    }
  // On libère la mémoire allouée pour les tableaux-pointeurs
  f_D_2(a,m,n); f_D_2(b,n,p); f_D_2(c,m,p);
  //-------------------------------------------------------------------------
  return 0;
}
*****************************************************************************
// matrices.dat
1 2 1
0 1 -1

1
-1
2

1 2 3 4
5 6 7 8
9 10 11 12

-1 2
3 -4
-5 6
7 -8
*****************************************************************************
//derivee.c
#include<stdio.h>
double f1(double x)
{
  return(1./(1.+x*x));
}
double f2(double x)
{
  return(x*(x-1.));
}
double deriv(double (*f)(double),double x,double h)
{
  double d;
  d=(*f)(x-2*h)-8*(*f)(x-h)+8*(*f)(x+h)-(*f)(x+2*h); d=d/12/h;
  return(d);
}
int main()
{
  int i;
  double x,h;
  x=1.37; h=10;
  printf("Derivees :\n\n");
  for(i=1;i<=20;i++)
```

```
    {
      h/=10;
      printf("1/(1+x*x) : x=%lg   h=%lg   valeur approchee : %20.10lg
                              valeur exacte : %20.10lg\n",x,h,deriv(f1,x,h),-2*x/(1+x*x)/(1+x*x));
      printf("x*(x-1)  :  x=%lg   h=%lg   valeur approchee : %20.10lg
                              valeur exacte : %20.10lg\n\n",x,h,deriv(f2,x,h),2*x-1);
    }
  return 0;
}
*******************************************************************************
//pendule_euler_sans_fonctions.c
#define N 2
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
  int i,j,np; double q[N],qp[N],t,dt,tfin;
  const double alpha=10;
  FILE *fich;
  // Methode d'Euler sans fonctions
  fich=fopen("pendule_euler_sf.res","w");
  t=0; q[0]=2; q[1]=0; np=10001; tfin=100; dt=(tfin-t)/(np-1);
  for(i=1;i<=np;i++)
    {
      fprintf(fich,"%lg %lg %lg\n",t,q[0],q[1]);
      qp[0]=q[1]; qp[1]=-alpha*sin(q[0]);
      for(j=0;j<N;j++) q[j]=q[j]+dt*qp[j];
      t+=dt;
    }
  fclose(fich);
  return 0;
}
*******************************************************************************
//pendule_euler_avec_fonctions.c
#define N 2
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
const double alpha=10;
/* DEFINITION DE LA OU DES EQUATIONS DIFFERENTIELLES */
void sd1(double q[],double t,double qp[],int n)
{
  qp[0]=q[1]; qp[1]=-alpha*sin(q[0]);
}
/* DEFINITON DE LA OU DES METHODES D'INTEGRATION */
void euler_vec(double q[],double qp[],double dt,int n)
{
  int i;
  for(i=0; i<n; i++) q[i]=q[i]+dt*qp[i];
}
/* LECTURE DES DONNEES ET IMPRESSION DES RESULTATS */
int main()
{
  int i,np; double q[N],qp[N],t,dt,tfin;
  FILE *fich;
  // Methode d'Euler avec fonctions
```

```
  fich=fopen("pendule_euler_af.res","w");
  t=0; q[0]=2; q[1]=0; np=10001; tfin=100; dt=(tfin-t)/(np-1);
  for(i=1;i<=np;i++)
    {
      fprintf(fich,"%lg %lg %lg\n",t,q[0],q[1]);
      sd1(q,t,qp,N);
      euler_vec(q,qp,dt,N);
      t+=dt;
    }
  fclose(fich);
  return 0;
}
*****************************************************************************
//pendule_rk4.c
#define N 2
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <bibli_fonctions.h>
const double alpha=10;
/* DEFINITION DE LA OU DES EQUATIONS DIFFERENTIELLES */
void sd1(double q[],double t,double qp[],int n)
{
  qp[0]=q[1]; qp[1]=-alpha*sin(q[0]);
}
/* LECTURE DES DONNEES ET IMPRESSION DES RESULTATS */
int main()
{
  int i,np; double q[N],t,dt,tfin;
  FILE *fich;
  // Methode RK4
  fich=fopen("pendule_rk4.res","w");
  t=0; q[0]=2; q[1]=0; np=1001; tfin=100; dt=(tfin-t)/(np-1);
  for(i=1;i<=np;i++)
    {
      fprintf(fich,"%lg %lg %lg\n",t,q[0],q[1]);
      rk4(sd1,q,t,dt,N);
      t+=dt;
    }
  fclose(fich);
  return 0;
}
*****************************************************************************
//div_rot_champ.c
#include<stdio.h>
#include<math.h>
#define N 3
void f(double *x,double *v)
{
  v[0]=2*x[0]+x[1]+x[2]; v[1]=x[0]+2*x[1]+x[2]*x[2]; v[2]=x[0]*x[1]+x[1]*x[1]*x[2]+x[2];
}
void dp(void (*cha)(double *x,double *v),double *x0,double h,double *div,double *rot)
{
  int i,j,k;
  double der[N][N],vp[N],vm[N];
  for(j=0;j<N;j++)
    {
```

```
      x0[j]=x0[j]+h; (*cha)(x0,vp); x0[j]=x0[j]-2*h; (*cha)(x0,vm); x0[j]=x0[j]+h;
      for(i=0;i<N;i++) der[i][j]=(vp[i]-vm[i])/2/h;
    }
  for(*div=0,i=0;i<N;i++) {*div+=der[i][i]; j=(i+1)%N; k=(i+2)%N; rot[i]=der[k][j]-der[j][k];}
}
int main()
{
  double x[N],h,div,rot[N];
  x[0]=1; x[1]=2; x[2]=3; h=1e-10;
  dp(f,x,h,&div,rot);
  printf("%lg %lg %lg %lg %lg %lg %lg\n",x[0],x[1],x[2],div,rot[0],rot[1],rot[2]);
  return 0;
}
*******************************************************************************
```