11.4 Les tableaux

SOLUTION DE L'EXERCICE 69:

```
#include<iostream>

using namespace std;
const int MAX_SIZE = 100;

int main()
{

  int t[MAX_SIZE];
  int i;
  int valeur;
  cin >> valeur;
  for (i=0; valeur!=-1;i++)
    {
      t[i]=valeur;
      cin >> valeur;
    }

  for (int j=i-1; j>=0;j--)
```

```
    cout << t[j]<<endl;
}
```

SOLUTION DE L'EXERCICE 70:

```
#include<iostream>

using namespace std;

const int N=10;


int find_max(int t[],int n)
{
  int imax=0;

  for (int i=1;i<n;i++)
    if (t[i]> t[imax])
      imax=i;

  return imax;
}

int main()
{
  int t[N]= {11,4,2,8,1,10,3,5,7,12};
  cout << "Indice du Max ="<< find_max(t,N)<<endl;
}
```

SOLUTION DE L'EXERCICE 71:

```
#include<iostream>

using namespace std;

const int dim = 10;
typedef int vecteur[dim];
void symetrie(vecteur v, int dim);

int main()
{
    vecteur v;
    int n;

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> v[i];

    symetrie(v, n);

    for (int j = 0; j < n; ++j)
        cout << v[j] << " ";
```

```
}

void symetrie(vecteur v, int n)
{
    int save;

    for (int i = 0; i < n / 2; ++i) {
        save = v[i];
        v[i] = v[n - i - 1];
        v[n - i - 1] = save;
    }
}
```

SOLUTION DE L'EXERCICE 72:

```
#include<iostream>

using namespace std;

const int n = 100;

int produit_scalaire(int t[n], int v[n]);

int main()
{
    int a[n], b[n];

    for (int i = 0; i < n; i++) {
        a[i] = i;
        b[i] = i + 1;
    }

    cout << produit_scalaire(a, b);
}

int produit_scalaire(int u[n], int v[n])
{
    int prod = 0;

    for (int i = 0; i < n; i++)
        prod += u[i] * v[i];

    return (prod);
}
```

SOLUTION DE L'EXERCICE 73:

```
// Tassement (non-stable) d'un vecteur

#include<iostream>

using namespace std;
```

```
const int n = 5;
typedef int vecteur[n];

void swap(int &, int &);
void tassement(vecteur);

int main()
{
    vecteur V;
    V[0] = 0;
    V[1] = 3;
    V[2] = 5;
    V[3] = 7;
    V[4] = 2;

    tassement(V);

    for (int i = 0; i < n; ++i)
        cout << V[i] << "-";
    cout << endl;
}

void swap(int &i, int &j)
{
    int tmp = i;
    i = j;
    j = tmp;
}

void tassement(vecteur v)
{
    int i, j;

    for (i = 0; i < n and v[i] != 0; ++i);
    for (j = n - 1; j >= 0 and v[j] == 0; --j);
    while (i < j) {
        swap(v[i], v[j]);
        while (i < n and v[i] != 0)
            ++i;
        while (j >= 0 and v[j] == 0)
            --j;
    }
}
```

SOLUTION DE L'EXERCICE 74:

```
#include<iostream>

using namespace std;

const int n = 10;

typedef int vecteur[n];

void tass_vecteur(vecteur vect);
```

```
int main()
{
    vecteur v;

    for (int i = 0; i < n; i++) {
        cout << "Element #" << i << " : ";
        cin >> v[i];
    }
    tass_vecteur(v);
    for (int i = 0; i < n; i++)
        cout << v[i] << " ";
}

//---------------------------------------------------------------

void tass_vecteur(vecteur vect)
{
    int pos = 0, i = 0;

    while (i < n) {
        if (vect[i] != 0) {
            if (i != pos) {
                vect[pos] = vect[i];
                vect[i] = 0;
            }
            ++pos;
        }
        ++i;
    }
}
```

SOLUTION DE L'EXERCICE 75:

```
#include<iostream>

using namespace std;

const int n = 10;
typedef int vecteur[n];
void impri_vect(vecteur, vecteur);

int main()
{
    vecteur don, pos;

    cout << "introduisez les " << n << " " << "composantes de don";
    for (int i = 0; i < n; ++i)
        cin >> don[i];

    cout << "introduisez les " << n << " " << "composantes de pos ";
    for (int j = 0; j < n; ++j)
        cin >> pos[j];

    impri_vect(don, pos);
```

```
}

void impri_vect(vecteur don, vecteur pos)
{
    cout << "ecriture de don en ordre donné par pos" << endl;
    for (int j = 0; j < n; ++j)
        cout << don[pos[j] - 1] << " ";
}
```

SOLUTION DE L'EXERCICE 76:

SOLUTION DE L'EXERCICE 77:

```
#include<iostream>

using namespace std;

const int MAX = 11;

typedef int vecteur[MAX];

void pascal(vecteur, int);

int main()
{
    int num;
    vecteur vect;

    cout << "Numero de la ligne du triangle a afficher : ";
    cin >> num;
    pascal(vect, num);
    if (num == 0)
      ++num ;
    for (int i = 0; i < num; i++)
        cout << vect[i] << " ";
}

void pascal(vecteur v, int n)
{
    v[0] = 1;
    // On met en place la ligne nř 0
    for (int i = 1; i < MAX; i++)
        v[i] = 0;
    // On calcule les lignes de 1 à n
    for (int i = 1; i <= n; i++)
        for (int j = i; j > 0; j--)
            v[j] = v[j - 1] + v[j];
}
```

SOLUTION DE L'EXERCICE 78:

```
#include<iostream>

using namespace std;

const int n = 10;
typedef int vecteur[n];
void gliss_1(vecteur, int);

int main()
{
    vecteur v;

    for (int i = 0; i < n; ++i)
        cin >> v[i];

    gliss_1(v, n);
    for (int j = 0; j < n; ++j)
        cout << v[j] << " ";
}

void gliss_1(vecteur w, int m)
{
    int save = w[0];

    for (int i = 1; i < m; ++i)
        w[i - 1] = w[i];
    w[m - 1] = save;
}
```

SOLUTION DE L'EXERCICE 79:

```
/*Explication intuitive (!?):
  Cet algorithme se base sur l'existence de cycles : on remplace un
  élément par son voisin k place plus loin à droite.(v[i]=v[(i+k)%n) On
  continue jusqu'à revenir à l'élément initial. On a alors fait un
  cycle. Seul problème, un cycle ne comprend pas forcément tous les
  éléments du tableau. Il faut alors recommencer un nouveau cycle. Or,
  on sait que s'il y a n cycles différents, n cases consécutives du
  tableau appartiennent forcément toutes à des cycles différents.  Par
  conséquent, pour commencer un nouveau cycle, il suffit de commencer
  avec l'élément juste à droite de l'élément initial du cycle
  précédent.(deb++)
*/

#include <iostream.h>
#include <iomanip.h>
const int TAILLE = 100;

void gliss_k_g(int v[TAILLE], int n, int k)
{
  /* variable indiquant l'indice de début du cycle courant */
```

```
  int deb = 0;
  /* cpt du nombre d'éléments déjà déplacés  */
  int cpt = 0;
  /* Quand tous les éléments sont déplacés, on s'arrête. */
  while (cpt < n)
    {
      int i;
      int sav = v[deb];
      /* boucle traitant un cycle */
      for (i = deb; (i + k) % n != deb; i = (i + k) % n)
        {
          v[i] = v[(i + k) % n];
          cpt++;
        }
      v[i] = sav;
      cpt++;
      deb++;
    }
}

int main()
{
  int v[TAILLE];
  int k;
  int n;
  cout << " Quelle taille de tableau désirez-vous?(max : 100)" << endl;
  cin >> n;

  for (int i = 0; i < n; i++) {
    v[i] = i + 1;
    cout << setw(3) << v[i];
  }
  cout << endl;

  cout << "Quel décalage désirez vous ?? " << endl;
  cin >> k;
  gliss_k_g(v, n, k);
  for (int i = 0; i < n; i++)
    cout << setw(3) << v[i];
  cout << endl;
}
```

SOLUTION DE L'EXERCICE 80:

```
#include<iostream>

using namespace std;
const int n = 10, m = 20;

void init_mat_m_n(int tab[m][n]);
void imprime_mat_m_n(int tab[m][n]);

int main()
{
```

```
    int mat[m][n];

    init_mat_m_n(mat);
    imprime_mat_m_n(mat);
}

void init_mat_m_n(int tab[m][n])
{
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            cin >> tab[i][j];
}

void imprime_mat_m_n(int tab[m][n])
{
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            cout << tab[i][j] << " ";
        cout << endl;
    }
}
```

SOLUTION DE L'EXERCICE 81:

```
#include<iostream>

using namespace std;

const int n = 100;

double trace(double tab[n][n]);

int main()
{
    double mat[n][n];

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            mat[i][j] = 0.0;

    cout << trace(mat);
}

double trace(double tab[n][n])
{
    double res = 0.0;

    for (int i = 0; i < n; i++)
        res += tab[i][i];
    return (res);
}
```

SOLUTION DE L'EXERCICE 82:

```
#include<iostream>

using namespace std;

const int dim = 10;
typedef int matrice[dim][dim];
void produit(matrice C, matrice A, matrice B, int l, int m, int n);
void affiche(matrice A, int l, int m);

int main()
{
    matrice A, B, C;
    int l, m, n;

    cout << "introduisez les dimensions de matrices ";
    cin >> l >> m >> n;
    cout << "matrice A et B " << endl;

    for (int i = 0; i < l; ++i)
        for (int j = 0; j < m; ++j)
            cin >> A[i][j];

    for (int k = 0; k < m; ++k)
        for (int s = 0; s < n; ++s)
            cin >> B[k][s];

    produit(C, A, B, l, m, n);
    affiche(C, l, n);
}

void produit(matrice c, matrice a, matrice b, int l, int m, int n)
{
    for (int i = 0; i < l; ++i) {
        for (int j = 0; j < n; ++j) {
            c[i][j] = 0;
            for (int k = 0; k < m; ++k)
                c[i][j] += a[i][k] * b[k][j];
        }
    }
}

void affiche(matrice A, int l, int n)
{
    for (int i = 0; i < l; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
}
```

SOLUTION DE L'EXERCICE 83:

```
#include<iostream>

using namespace std;

const int n = 3;

typedef int matrice[n][n];

bool symetrie(matrice);
bool asymetrie(matrice);
void init_matrice(matrice);

int main()
{
    matrice a;

    init_matrice(a);
    if (symetrie(a))
        cout << "La matrice est symetrique" << endl;
    else
        cout << "La matrice n''est pas symetrique" << endl;
    if (asymetrie(a))
        cout << "La matrice est antisymetrique" << endl;
    else
        cout << "La matrice n''est pas antisymetrique" << endl;
}

//-----------------------------------------------

bool symetrie(matrice a)
{
    bool sym = true;

    for (int i = 0; i < n and sym; i++)
        for (int j = 0; j < i and sym; j++)
            if (a[i][j] != a[j][i])
                sym = false;
    return (sym);
}

//-----------------------------------------------

bool asymetrie(matrice a)
{
    bool asym = true;

    for (int i = 1; i < n and asym; i++)
        for (int j = 0; j < i and asym; j++)
            if (a[i][j] != -a[j][i])
                asym = false;
    return (asym);
}

//-----------------------------------------------
```

```
void init_matrice(matrice a)
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            cout << "Element #" << i << "," << j << " : ";
            cin >> a[i][j];
        }
}
```

SOLUTION DE L'EXERCICE 84:

```
#include<iostream>

using namespace std;

const int m = 10;
typedef int matrice[m][m];
void init_mat(matrice mat, int n);
void impri_mat(matrice mat, int n);
void rotation1(matrice mat, int n);
void rotation2(matrice mat, int n);



int main()
{
    int n;
    matrice a;

    cin >> n;

    init_mat(a, n);
    rotation1(a, n);
    impri_mat(a, n);

    init_mat(a, n);
    rotation2(a, n);
    impri_mat(a, n);
}

void init_mat(matrice mat, int n)
{
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            cin >> mat[i][j];
}

void rotation1(matrice mat, int n)
{
    for (int i = 1; i <= n / 2; ++i) {
        for (int j = i - 1; j < n - i; ++j) {
            int save = mat[i - 1][j];

            mat[i - 1][j] = mat[j][n - i];
```

```
            mat[j][n - i] = mat[n - i][n - j - 1];
            mat[n - i][n - j - 1] = mat[n - j - 1][i - 1];
            mat[n - j - 1][i - 1] = save;
        }
    }
}

void rotation2(matrice mat, int n)
{
    for (int j = n - 1; n <= 2 * j; --j)
        for (int i = n - j - 1; i <= j - 1; ++i) {
            int save = mat[j][n - i - 1];
            mat[j][n - i - 1] = mat[n - i - 1][n - j - 1];
            mat[n - i - 1][n - j - 1] = mat[n - j - 1][i];
            mat[n - j - 1][i] = mat[i][j];
            mat[i][j] = save;
        }
}

void impri_mat(matrice mat, int n)
{
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j)
            cout << mat[i][j] << " ";
        cout << endl;
    }
}
```