TD Données

Exercice 1

Corriger l'algorithme en pseudo-code suivant afin de résoudre le problème suivant :

- Données : deux vecteurs p et q dans un espace (Euclidien) à 3 dimensions
- Résultat : la somme des vecteurs p+q

Algorithme 1: SommeDeVecteurs

```
variables
    réel p[3]
    réel q[3]
    réel r[3]

début
    pour i ← 1 à 3 faire
        pour j ← 1 à 3 faire
            r[j] ← p[i] + q
        fin
    fin
fin
```

Exercice 2

Ecrire un algorithme permettant de résoudre le problème suivant :

- Données : deux vecteurs p et q dans un espace (Euclidien) à 3 dimensions
- Résultat : le produit scalaire de p et q

Exercice 3

Pour sa naissance, la grand-mère de Gabriel place une somme de 1000 e sur son compte épargne rémunéré au taux de 2.25% (chaque année le compte est augmenté de 2.25%). Développer un algorithme permettant d'afficher un tableau sur 20 ans associant à chaque anniversaire de Gabriel la somme acquise sur son compte.

Exercice 4

Felix est un fermier qui dispose d'un couple de shadoks capables de se reproduire à vitesse phénoménale. Un couple de shadocks met deux mois pour grandir ; à partir du troisième mois, le couple de shadocks engendre une paire de nouveaux shadocks (qui mettront deux mois pour grandir et donc trois mois pour engendrer une nouvelle paire, etc.). Et surtout, les shadoks ne meurent jamais ! D'après cet exercice le nombre de couples de shadoks Fn à chaque mois n obéit à la loi :

– F1 = 1

– F2 = 1

– Fn = Fn−1 +Fn−2

Développer un algorithme permettant de construire le tableau des couples depuis le premier jusqu'au 20ème mois.

Exercice 5

Corriger le programme C++ suivant afin de résoudre le problème suivant :

- Données : un tableau de 100 entiers, une valeur entière x
- Résultat : le nombre d'occurrences de x dans le tableau

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

int main()
{
int tableau[100];
int i,x,occurrences;

cout << "Entrer votre valeur: ";
cin >> x;
i = 0;
occurrences = 0;
while(tableau[i] != x) i++;
cout << occurrences << endl;
return 0;
}
```

Exercice 6

Nous souhaitons développer un algorithme permettant de rechercher un élément dans un tableau de 100 entiers en partant des deux extrémités. Dans cette perspective, corriger le programme C++ suivant.

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

int main()
{
int tableau[100];
int i,j,x;
bool trouve;

cout << "Entrer votre valeur: ";
cin >> x;
i = 0;
j = 100;
trouve = 0;
do
    {
    trouve = (tableau[i] == x) && (tableau[j] == x);
    i++;
    j--;
    }
while(!trouve);
cout << trouve << endl;
return 0;
}
```

Exercice 7

Ecrire un algorithme permettant de résoudre le problème suivant :

- Données : un tableau *tableau* contenant 100 entiers
- Résultat : “vrai” si les entiers sont consécutifs et “faux” sinon

Rappel : deux entiers x et y sont consécutifs si et seulement si y = x +1.

Exercice 8

Compléter l’algorithme suivant afin de calculer la distance entre deux points p et q entrés par l’utilisateur. Chaque point est représenté par son abscisse et son ordonnée.

```
distanceEntreDeuxPoints
    // Déclaration du type point
    ?
    variables
        point p,q
        réel dist

    début
        ?
    fin
```

Exercice 9

Compléter l’algorithme suivant afin de tester si un triangle entré par l’utilisateur est rectangle ou non. Un triangle est composé de trois points, appelés sommets, et de trois segments qui les relient. Le triangle est rectangle s’il contient un angle droit.

```
distanceEntreDeuxPoints
    // Déclaration du type point
    ?
    // Déclaration du type triangle
    ?
    variables
        triangle T
        booleen estRectangle

    début
        ?
    fin
```

Exercice 10

Ecrire un algorithme permettant de construire le barycentre d’un ensemble de 10 points entrés par l’utilisateur. Le barycentre d’un ensemble de n points est le point B donné par :

$$x_B = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ et } y_B = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Exercice 11

Compléter le code C ci-dessous afin de construire un jeu de 32 cartes. Le jeu est représenté par un tableau de 32 éléments distincts, chacun étant du type Carte.

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

enum Couleur {trefle, carreau, pique, coeur};
enum Face {sept, huit, neuf, dix, valet, dame, roi, as};
struct Carte
    {
    Couleur couleur;
    Face face;
    };

int main()
{
Carte jeu[32];

// Code pour construire le jeu
...

}
```

Exercice 12

Compléter le code C ci-dessous afin de “mélanger” un jeu de 32 cartes (on supposera que le jeu a été construit par le code de l’exercice 5). L’algorithme permute les cartes correspondant à deux indices choisis aléatoirement, et effectue cette opération 100 fois.

```
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

enum Couleur {trefle, carreau, pique, coeur};
enum Face {sept, huit, neuf, dix, valet, dame, roi, as};
struct Carte
    {
    Couleur couleur;
    Face face;
    };

int main()
{
Carte jeu[32];
int i, index1, index2;

// Code pour mélanger le jeu
srand((unsigned)time(0));
for(i = 0; i < 100; i++)
    {
    index1 = (rand() % 32 );
    index2 = (rand() % 32 );

    ...

    }
}
```

Exercice 13

Ecrire une structure en pseudo-code permettant de représenter le type album musical. L’album est décrit par son titre, son groupe, son genre (classique, dance, folk, electro, jazz, musique du monde, pop, rap, reggae, rock, soul, variété), sa date de sortie, sa durée, et son

format (CD, MP3, Vinyle). On supposera que le titre et le groupe font chacun 100 caractères maximum.

Exercice 14

Ecrire un algorithme en pseudo-code permettant de construire une bibliothèque de 100 albums musicaux. Pour chaque album, l'utilisateur saisit son titre, groupe, genre, date de sortie, durée, et format.