Algorithmique - Correction du TD4

IUT 1ère Année

19 décembre 2012

1 Les énumérations et structures

Exercice 1. 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.

```
structure Point
    réel x
    réel y

variables
    Point p, q
    réel dist

début
    dist ← sqrt((p.x − q.x) ∗ (p.x − q.x) + (p.y − q.y) ∗ (p.y − q.y))
fin
```

distance entre deux points

Exercice 2. 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.

Note : dans la correction on utilise la fonction distance définie plus haut. Rappelons qu'il ne faut pas hésiter à *réutiliser* les fonctions ou procédures que vous avez déjà construites. Observons que dans la correction, la valeur du booléen `estRectangle` est donné par une expression de comparaison.

```
// Déclaration du type Point
structure Point
    réel x
    réel y

// Déclaration du type Triangle
structure Triangle
    Point a
    Point b
    Point c


variables
    Triangle T
    réel d_ab, d_ac, d_bc
    booleen estRectangle

début
    d_ab ← distance(T.a, T.b)
    d_bc ← distance(T.b, T.c)
    d_ac ← distance(T.a, T.c)
    estRectangle ← faux
    si d_ab > d_bc et d_ab > d_ac alors
        estRectangle ← d_ab = sqrt(d_ac + d_bc)
    si d_ac > d_ab et d_ac > d_bc alors
        estRectangle ← d_ab = sqrt(d_ab + d_bc)
    si d_bc > d_ab et d_bc > d_ac alors
        estRectangle ← d_bc = sqrt(d_ab + d_ac)
    afficher estRectangle
fin
```

Triangle rectangle

Exercice 3. 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 :

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

```
structure Point
    réel x
    réel y

variables
    Point ensemble[10], barycentre

début
    barycentre.x ← 0
    barycentre.y ← 0
    pour i de 0 à 9 faire
        barycentre.x ← barycentre.x + ensemble[i].x
        barycentre.y ← barycentre.y + ensemble[i].y
    fin
    barycentre.x ← barycentre.x/10.0
    barycentre.y ← barycentre.y/10.0
fin
```

Barycentre d'un ensemble de points

Exercice 4. 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];
int x = 0;
for(int i = 0; i < 8; i++)
    for(int j = 0; j < 4; j++)
        {
        jeu[x].face     = (Face)i;
        jeu[x].couleur  = (Couleur)j;
        x++;
        }
}
```

Exercice 5. 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).

```
#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;
srand((unsigned)time(0));
for(i = 0; i < 100; i++)
    {
    index1 = (rand() % 32 );
    index2 = (rand() % 32 );
    Carte temp  = jeu[index1];
    jeu[index1] = jeu[index2];
    jeu[index2] = temp;
    }
}
```

Exercice 6. 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 ici que le titre et le groupe sont des chaînes (on peut utiliser aussi un tableau de 100 caractères).

```
énumération Format {CD,MP3,Vinyle}
énumération Genre {classique, dance, folk, electro, jazz, musique du monde, pop, rap, reggae, rock, soul, variété}
structure Album
    chaîne titre
    chaîne groupe
    Genre genre
    entier date
    entier durée
    Format format
```

Structure Album Musical

Exercice 7. 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.

```
énumération Format {CD,MP3,Vinyle}
énumération Genre {classique, dance, folk, electro, jazz, musique du monde, pop, rap, reggae, rock, soul, variété}
structure Album
    chaîne titre
    chaîne groupe
    Genre genre
    entier date
    entier durée
    Format format

variables
    Album bibliothèque[100]
    entier i, g, f

début
    pour i de 0 à 99 faire
        afficher "Titre : "
        lire bibliothèque[i].titre
        afficher "Groupe : "
        lire bibliothèque[i].groupe
        afficher "Genre : (0 classique, 1 dance, 2 folk, 3 electro, 4 jazz, 5 musique du monde, 6 pop, 7 rap, 8 reggae, 9 rock, 10
        soul, 11 variété)"
        lire g
        bibliothèque[i].genre ← (Genre) g
        afficher "Date (année) : "
        lire bibliothèque[i].date
        afficher "Durée (mn) : "
        lire bibliothèque[i].durée
        afficher "Format : (0 CD, 1 MP3, 2 Vinyle)"
        lire f
        bibliothèque[i].genre ← (Format) f
    fin
fin
```

Bibliothèque d'albums

2 Les Fonctions

Exercice 8. Dans le programme ci-dessous, indiquer la portée des différentes variables.

```
1  #include <cmath>
2  using namespace std;
3
4  int a = 100;
5
6  float sumCarres(int n)
7  {
8  int x = 0;
9  for(int i = 0; i < n; i++)
10     {
11     float y = i * i;
12     x += y;
13     }
14 return x;
15 }
16
17 int main()
18 {
19 int x = sumCarres(a);
20 for(int j = 0; j < 100; j++)
21     {
22     float y = sqrt(i);
23     x -= y;
24     }
25 }
```

- `a` : [4,25]
- `n` : [7,15]
- `x` : [8,15] ∪ [19,25]
- `i` : [9,15]
- `y` : [11,13] ∪ [22,24]
- `j` : [20,25]

Notons que le programme ne compilera pas puisque la portée de la variable locale `i` ne couvre pas la ligne 22.

Exercice 9. Corriger les erreurs du programme ci-dessous afin de retourner une moyenne de 100 entiers.

```
using namespace std;

typedef int TableauEntiers [100];

void moyenne(TableauEntiers& tab)
{
float m = 0;
for(int i = 0; i < 100; i++)
    m += tab[i];
return m/100.0;
}

int main()
{
TableauEntiers tab;
cout << moyenne(tab);
}
```

Exercice 10. En se basant sur les albums de musique vus dans l'exercice 6, écrire une fonction en pseudo-code permettant de résoudre le problème suivant :

- Entrée : une bibliothèque de 100 albums et un genre de musique
- Sortie : le nombre d'albums du genre g apparaissant dans la bibliothèque.

```
énumération Format {CD,MP3,Vinyle}

énumération Genre {classique, dance, folk, electro, jazz, musique du monde, pop, rap, reggae, rock, soul, variété}

structure Album
    chaîne titre
    chaîne groupe
    Genre genre
    entier date
    entier durée
    Format format

type Album Bibliotheque [100]

entier occurrences(Bibliotheque bib, Genre g)
début
    entier i
    entier n ← 0
    pour i de 1 à 99 faire
        n ← n+ (bib[i].genre = g)
    retourner n
fin
```

Fonction pour le nombre d'albums d'un genre particulier

Exercice 11. En se basant à nouveau sur les albums de musique, écrire une fonction en pseudo-code permettant de résoudre le problème suivant :

- Entrée : une bibliothèque de N albums et un titre t d'album
- Sortie : l'index du premier album dans la bibliothèque correspondant au titre t ; la valeur N est retournée s'il n'existe aucun album intitulé par t dans la bibliothèque.

```
énumération Format {CD,MP3,Vinyle}

énumération Genre {classique, dance, folk, electro, jazz, musique du monde, pop, rap, reggae, rock, soul, variété}

structure Album
    chaîne titre
    chaîne groupe
    Genre genre
    entier date
    entier durée
    Format format

type Album Bibliotheque [N]

entier recherche(Bibliotheque bib, chaîne t)
début
    entier i ← 0
    booleen trouvé ← 0
    tant que (non trouvé) et i < N faire
        trouvé ← bib[i].titre = t
        si non trouvé alors i ← i + 1
    retourner i
fin
```

Fonction pour la recherche d'un album selon son titre

Exercice 12. En se basant sur l'exercice 10, écrire une fonction en C++ permettant de résoudre le problème suivant :
- Entrée : une bibliothèque de 100 albums de musique
- Sortie : un tableau avec le nombre d'albums par genre

Pour des raisons d'efficacité, on utilise dans la correction une chaîne de caractères pour le genre.

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

enum Format {CD,MP3, Vinyle};

struct Album{
string titre;
string groupe;
string genre;
int    date;
int    durée;
Format format;
};

typedef int Repartitions [12];
typedef Album Bibliotheque [100];

void classification(const Bibliotheque& bib, Repartitions& rep)
{
string genres [12] = {
"classique",
"dance",
"folk",
"electro",
"jazz",
"musique du monde",
"pop",
"rap",
"reggae",
"rock",
"soul",
"variete"};

int i = 0, j = 0;

for(j = 0; j < 12; j++)
    rep[j] = 0;

for(i = 0; i < 100; i++)
    {
    j = 0;
    bool trouve = 0;
    while(!trouve && j < 12)
        {
        trouve = (genres[j] == bib[i].genre);
        if(!trouve) j++;
        }
    if(j < 12)
        rep[j]++;
    }
}
```

Exercice 13. En se basant sur le type `Point` vu dans l'exercice 1, construire une fonction permettant de résoudre le problème suivant :

- Entrée : un ensemble (représentée par un tableau) de 100 points
- Sortie : la plus grande distance entre deux points de l'ensemble

La correction est faite ici en C++ et se sert de la déclaration de la fonction `distance` construite dans l'exercice 1.

```
#include <string>
#include <cmath>
using namespace std;

struct Point{
float x;
float y;
};

typedef Point TableauDePoints [100];

// Déclaration de la fonction distance
float distance(const Point& p, const Point& q);

//Définition de la fonction distanceMax
float distanceMax(const TableauDePoints& tab)
{
float dist = 0, max = 0;

for(int i = 0; i < 99; i++)
    for(int j = 0; j < 100; j++)
        {
        dist = distance(tab[i],tab[j]);
        if(dist > max) max = dist;
        }

return max;
}
```

Exercice 14. En se basant sur la structure de `Carte` donnée dans l'exercice 4, construire une fonction permettant de determiner si un joueur de Poker possède une paire. La fonction reçoit en entrée une *main* (un tableau de cinq cartes différentes) et affiche vrai si et seulement si les cinq cartes possèdent au moins une paire. Rappelons qu'une paire est formée par deux cartes de même face (ex : deux valets).

Dans la correction, la fonction est implémentée en C++ (même principe pour le pseudo-code).

```
#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;
    };

typedef Carte Main [5];

bool testerPaire(const Main& m)
```

```
{
bool paire = 0;
int  i = 0;
while(!paire && i < 4)
    {
    int j = i + 1;
    while(!paire && j < 5)
        {
        paire = (m[i].face == m[j].face);
        j++;
        }
    i++;
    }
return paire;
}
```

Exercice 15*. En se basant sur les albums de musique vus dans l'exercice 6, écrire une procédure permettant de ranger par ordre alphabétique des albums saisis au fur et à mesure par l'utilisateur. Si besoin, on pourra se servir de la déclaration de la procédure `permuter` vue en cours (pour les entiers).

Dans la correction, la procédure est écrite en C++ et utilise une procédure `saisir` qui permet à l'utilisateur de rentrer un nouvel album.

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

enum Format {CD,MP3,Vinyle};
struct Album{
string titre;
string groupe;
string genre;
int    date;
int    duree;
Format format;
};

typedef Album Bibliotheque [100];

// Declaration d'une procédure permettant de saisir un album
void saisir(Album& album);

// Definition de la procédure copier
void copier(const Album& original, Album& copie)
{
copie.titre    = original.titre;
copie.groupe   = original.groupe;
copie.genre    = original.genre;
copie.date     = original.date;
copie.duree    = original.duree;
copie.format   = original.format;
}

// Définition de la procédure trier
void trier(Bibliotheque& bib)
{
for(int i = 0; i < 100; i++)
    {
```

```
    Album album;
    saisir(album);

    int j = i;
    while((j > 0) && (bib[j-1].titre > album.titre))
        {
        copie(bib[j-1],bib[j]);
        j--;
        }
    copie(album,tableau[j]);
    }
}
```

Exercice 16*. En se basant sur toutes les fonctions et procédures définies pour les albums de musique, écrire une fonction permettant de rechercher de manière dichotomique un album dans une bibliothèque rangée par ordre alphabétique.

Dans la correction, la procédure est écrite en C++, retourne l'index de l'album s'il est présent, et 100 (la taille de la bibliothèque) s'il est absent.

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

enum Format {CD,MP3, Vinyle};
struct Album{
string titre;
string groupe;
string genre;
int    date;
int    duree;
Format format;
};

typedef Album Bibliotheque [100];

// Définition de la fonction de recherche
int rechercher(const Bibliotheque& bib, const string& titre)
{
int gauche  = 0, droite  = 99, milieu;
bool trouve = 0;

do  {
    milieu = (gauche + droite)/2;
    trouve = (titre == bib[milieu].titre);
    if(titre > bib[milieu].titre) gauche = milieu + 1;
    if(titre < bib[milieu].titre) droite = milieu - 1;
    }
while(gauche <= droite && !trouve);

if(trouve)
    return milieu;
return -1;
}
```

Exercice 17*. En se basant sur la structure de `Carte` donnée dans l'exercice 4, construire une procédure permettant de trier une main au Poker. L'algorithme part de cinq cartes rangées n'importe comment, et les tries selon leur face puis leur couleur.
Dans la correction, la procédure est implémentée en C++ et utilise la fonction `swap` de la STL (utiliser le header algorithm) qui permute deux objets de la même structure ou même classe.

```
#include <algorithm>
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;
    };

typedef Carte Main [5];

void trier(Main& m)
{
for(int i = 0; i < 4; i++)
    for(int j = i + 1; j < 5; j++)
        {
        if((m[j].face > m[i].face) || ((m[j].face == m[i].face) && (m[j].couleur > m[j].couleur)))
            swap(m[i],m[j]);
        }
}
```

Exercice 18. Construire un algorithme permettant de determiner si un joueur de Poker possède une quinte. Rappelons qu'une quinte est formée par cinq cartes dont les faces se suivent (ex : neuf de pique, huit de trèfle, valet de cœur, dix de pique, dame de carreau). On pourra, si besoin, se servir de toutes les fonctions définies pour le jeu de 32 cartes.

```
#include <algorithm>
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;
    };

typedef Carte Main [5];

// Déclaration de la fonction trier
void trier(Main& m);

// Définition de la procédure testerQuinte
bool testerQuinte(Main& m)
{
trier(Main);

bool quinte = vrai;
int i = 0;
while(quinte && i < 4)
    {
    quinte = (m[i+1].face == m[i].face + 1);
    i++;
    }
return quinte;
}
```