Algorithmique - TD4

IUT 1ère Année

10 octobre 2014

Les exercices marqués avec une étoile sont un peu plus difficiles. Il est fortement recommandé de travailler chez soi afin de progresser sur ces exercices.

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.

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

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.

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

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 :

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

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];

// Code pour construire le jeu
...

}
```

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). 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 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 que le titre et le groupe font chacun 100 caractères maximum.

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.

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 }
```

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

```
using namespace std;

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

int main()
{
int tab[100];
cout << moyenne(int 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.

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.

Dans cet exercice, nous utiliserons l'opérateur = pour déterminer si deux chaînes de caractères de même taille sont identiques. En d'autres termes, étant données deux chaînes de caractères c et c' de même taille, l'expression booléenne $c = c'$ est vraie si et seulement si c et c' sont identiques.

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

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

On pourra se servir de la déclaration de la fonction `dist` construite dans l'exercice 1.

Exercice 14. En se basant sur la structure de `Carte` donnée dans l'exercice 4, construire une fonction en pseudo-code 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).

Exercice 15*. En se basant sur les albums de musique vus dans l'exercice 6, écrire une procédure en pseudo-code 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).

L'opérateur $\leq$ sera utilisé pour comparer les titres d'albums par ordre alphabétique. Etant données deux chaînes de caractères c et c' de même taille, l'expression booléenne $c \leq c'$ est vraie si et seulement si c est classée avant c' selon l'ordre alphabétique.

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.

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.

Si besoin on pourra se servir de la déclaration de la procédure `permuter`. Etant données deux cartes c et c', c est rangée avant c' si (1) la face de c est plus petite que la face de c' (selon l'ordre des faces donné par le type énuméré `Face`), ou (2) la face c est égale à la face de c' mais la couleur de c est plus petite que la couleur de c' (selon l'ordre des couleurs donné par le type énuméré `Couleur`).

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.