Algorithmique - TD1

IUT 1ère Année
Enseignant : Frédéric Koriche

10 septembre 2012

Exercice 1.

Nous allons étudier le problème suivant :
- Données : une série de lettres aimantées sur un tableau
- Résultat : un alignement des lettres par ordre alphabétique

Question 1.1. Considérons tout d'abord la série suivante suivante :

(C) (A) (B) (D)

On veut naturellement obtenir :

(A) (B) (C) (D)

Comment résoudre le problème ? Décomposez votre raisonnement en une série d'instructions simples.

Question 1.2. Considérons à présent la série suivante suivante :

(E) (B) (F) (B)

On veut obtenir :

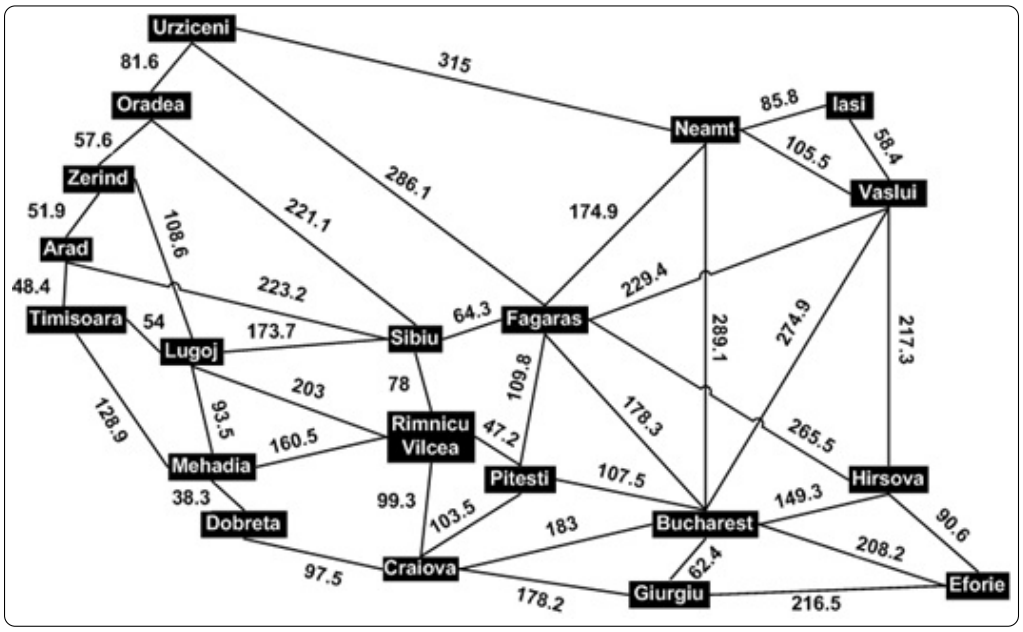
(B) (B) (E) (F)

Votre algorithme est-il correct pour cette série ? Si ce n'est pas le cas, trouver un moyen de résoudre le problème *pour toute* série de lettres.

Exercice 2.

Etudions le problème suivant :
- Données : une carte routière avec des villes connectées par des routes (avec un kilométrage donné) ; une ville de départ et une ville d'arrivée
- Résultat : un itinéraire entre la ville de départ et la ville d'arrivée

Considérons la carte ci-jointe ; on veut partir d'Eforie pour arriver à Arad. Comment trouver un itinéraire ? Décomposez votre raisonnement en une série d'instructions simples.



Exercice 3.

Corrigez les erreurs de l'algorithme ci-dessous afin de résoudre le problème suivant :
- Données : le rayon d'un cercle
- Résultat : la surface du cercle

Algorithme 1: surfaceCercle

```
constante
    réel pi ⟵ 3.14

variable
    réel rayon

début
    lire rayon
    lire pi
    surface ⟵ rayon × rayon × pi
    afficher surface
fin
```

Exercice 4.

Corrigez les erreurs de l'algorithme ci-dessous afin de résoudre le problème suivant :
- Données : deux nombres réels a et b
- Résultat : la permutation de a et b (a possède la valeur de b et *vice versa*)

Algorithme 2: PermuterVariables

```
variables
    réel a
    réel b

début
    lire a
    lire b
    a ⟵ b
    b ⟵ a
    afficher a
    afficher b
fin
```

Exercice 5.

Nous supposons un ensemble de variables définies et initialisées de la manière suivante :

Variable	Type	Valeur
i	entier	50
j	entier	20
x	reel	20.0
a	booléen	vrai
b	booléen	faux

Evaluer chacune des expressions suivantes et donner sa valeur avec son type.

1. $(i + j) \bmod 4$
2. $x/3$
3. $i/3$
4. a et b
5. a ou b
6. non ((a et b) ou a)
7. $i = j$
8. (b ou ($j < i$))

Exercice 6.

Nous cherchons à afficher le périmètre d'un cercle dont le rayon est donné par l'utilisateur. Pour cela, corrigez les erreurs du programme C++ suivant.

Listing 1 – PerimetreDunCercle

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

float rayon, perimetre;
const float pi = 3.14;

main()
{
cout << "Entrer le rayon du cercle: ";
cin >> rayon;
```

```
perimetre = rayon;
perimetre << pi;
perimetre == 2;

cout << "Le perimetre du cercle est: " >> perimetre << endl;
}
```

Exercice 7.

Nous cherchons à calculer la distance entre deux points (2D). La fonction `sqrt` calcule la racine carrée d'un réel. Corriger le programme pour garantir le bon résultat.

Listing 2 – distance

```
#include <iostream>
#include <math.h>
using namespace std;

float x1, y1, x2, y2;
int carreAbs, carreOrd, distance;

main()
{
cout << "Entrer l'abscisse du premier point: ";
cin >> x1;
cout << "Entrer l'ordonnee du premier point: ";
cin >> y1;
cout << "Entrer l'abscisse du second point: ";
cin >> x2;
cout << "Entrer l'ordonnee du second point: ";
cin >> y2;

carreAbs = (x1 + x2) * (x1 + x2);
carreOrd = (y1 + y2) * (y1 + y2);
distance = sqrt(carreAbs + carreOrd);

cout << "La distance est de: " << distance << endl;
}
```

Exercice 8.

Ecrire l'algorithme *HeuresMinutesSecondes* permettant de résoudre le problème suivant :
- Données : un nombre entier n (représentant une quantité de secondes)
- Résultat : la conversion de ce nombre en heures, minutes et secondes.

Par exemple, le nombre 8000 est associé à 2 heures 13 minutes et 20 secondes.

Exercice 9.

Construire un algorithme permettant de résoudre le problème suivant :
- Données : deux cercles C_1 et C_2, chacun étant spécifié par l'abscisse et l'ordonnée de son centre, et son rayon
- Résultat : "vrai" si l'intersection de C_1 et C_2 n'est pas vide (les cercles sont en collision), et "faux" sinon.

Note : cet algorithme est souvent utilisé pour la détection de collisions dans les jeux vidéo.