Algorithmique - TD1 Correction

IUT 1ère Année
Enseignant : Frédéric Koriche

10 septembre 2012

Note : plusieurs approches sont possibles pour résoudre les exercices 1 et 2

Exercice 1.

Algorithme : trierLettresMagnétiques

```
Mettre en bas du tableau une lettre spéciale Ω (indiquant "fin")
tant que il reste des lettres en haut du tableau faire
    Prendre la première lettre H
    Soit B la première lettre en bas du tableau
    tant que B < H et B n'est pas la lettre Ω faire
        B devient la lettre suivante
    fin
    Insérer H juste avant B
fin
Retirer Ω du bas du tableau
```

Exercice 2.

Algorithme : cheminDansUneCarte

```
Soit W = ∅ l'ensemble des villes parcourues
Soit v la ville de départ
tant que v n'est pas la ville d'arrivée faire
    Prendre la ville w la plus proche de v qui n'est pas dans W
    Dessiner le segment (v, w)
    Mettre w dans W
    v devient w
fin
```

Exercice 3.

Algorithme 3: surfaceCercle

constante
  réel pi ⟵ 3.14

variable
  réel rayon
  réel surface

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

Exercice 4.

Algorithme 4: permuterVariables

variables
  réel a
  réel b
  réel c

début
  lire a
  lire b
  c ⟵ b
  b ⟵ a
  a ⟵ c
  afficher a
  afficher b
fin

Exercice 5.

Expression	Type	Valeur
$(i + j) \bmod 4$	**entier**	2
$x/3$	**réel**	6.66
$i/3$	**entier**	16
a et b	**booléen**	faux
a ou b	**booléen**	vrai
non ((a et b) ou a)	**booléen**	faux
$i = j$	**booléen**	faux
(b ou ($j < i$))	**booléen**	vrai

Exercice 6.

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 = perimetre * pi;
perimetre = perimetre * 2;

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

Exercice 7.

Listing 2 – distance

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

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

main()
{
cout << "Entrer l'absisse du premier point: ";
cin >> x1;
cout << "Entrer l'ordonnee du premier point: ";
cin >> y1;
cout << "Entrer l'absisse 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.

Algorithme 5: heuresMinutesSecondes

```
variables
    réel duree
    réel heures
    réel minutes
    réel secondes

début
    afficher "Entrer une durée : "
    lire duree
    heures ⟵ duree / 3600
    duree ⟵ duree mod 3600
    minutes ⟵ duree / 60
    secondes ⟵ duree mod 60
    afficher heures "h, " minutes "mn, " secondes "s"
fin
```

Exercice 9.

Algorithme 6: sontEnCollision

```
variables
    réel x1, y1, r1
    réel x2, y2, r2
    réel carreAbs, carreOrd, distance ; booléen résultat

début
    afficher "Entrer l'absisse, l'ordonnée et le rayon du premier cercle"
    lire x1
    lire y1
    lire r1
    afficher "Entrer l'absisse, l'ordonnée et le rayon du second cercle"
    lire x2
    lire y2
    lire r2
    carreAbs ⟵ (x1 − x2) × (x1 − x2)
    carreOrd ⟵ (y1 − y2) × (y1 − y2)
    distance ⟵ racine(carreAbs + carreOrd)
    résultat ⟵ (distance < r1 + r2)
    afficher "En collision : " résultat
fin
```