

Exercices (full)

University of Geneva
www.miralab.ch

Exercice 3 : while

45

Faire un programme qui affiche une somme ajoutant 2 à chaque fois, jusqu'à un résultat maximum donnée par l'utilisateur (la somme sera calculée par additions successives)

Exemple :

pour 7 → affichage de 0, 2, 4, 6

pour 4 → affichage de 0, 2, 4



University of Geneva
www.miralab.ch

LCI
Introduction au langage C

Exercice 3 : pas à pas

46

Déclaration des variables

```
int max = limite donnée par l'utilisateur
int somme = somme cumulée
```

Algorithme

```
somme ← 0
saisir max
faire tant que (somme <= max)
    afficher somme
    somme ← somme + 2
```

University of Geneva
www.miralab.ch

LCI
Introduction au langage C

Exercice 3 : solution

47

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

void main()
{
    int somme=0;
    int max;
    cout << "nombre maximal : ";
    cin >> max;
    while (somme<=(max-2))
    {
        somme+=2;
        cout << somme << endl;
    }
}
```

University of Geneva
www.miralab.ch

LCI
Introduction au langage C

Exercice 4 : struct

48

1. Définir un type de structure permettant de représenter un point de l'espace (float) et un index (int). + qqs exemples d'initialisation et d'accès.
2. Définir une structure et un type de structure représentant les informations suivantes :
 - Le nom (string)
 - Un identifiant (int)
 - L'âge (float)
 + qqs exemples d'initialisation et d'accès.

University of Geneva
www.miralab.ch

LCI
Introduction au langage C

Exercice 4 : solution

49

- ```
typedef struct point {
 float x, y, z;
 int index;
};

struct point p1 = { 3, 4.5, 0, 0}
struct point p2;
p2.x = 3; p2.y = 4.5; p2.z = 0; p2.index = 0;
int dx = p1.x - p2.x;
```
- ```
struct quic {
    string nom;
    int identifiant;
    float age;
};

struct quic moi = { ``Marabou``, 1, 5};
cout << moi.nom << endl;
```

University of Geneva
www.miralab.ch

LCI
Introduction au langage C

Exercice 5

50

Tout pixel d'une image est caractérisé par un couple de coordonnées (x,y). On peut donc calculer des distances entre pixels. Les distances les plus courantes sont (pour deux pixels P(xp,yp) et Q(xq,yq)):

- distance de Manathan : $d1(P,Q) = |xp - xq| + |yp - yq|$
- distance euclidienne : $d2(P,Q) = [(xp - xq)^2 + (yp - yq)^2]^{1/2}$
- distance de l'échiquier : $dinf(P,Q) = \text{Max}(|xp - xq|, |yp - yq|)$

Ces distances sont reliées par la propriété :

$$dinf(P,Q) \leq d2(P,Q) \leq d1(P,Q)$$

Écrire un programme définissant les structures nécessaires définir les points et les fonctions permettant de calculer les distances de **Manathan** et **euclidienne** (echiquier en option ☺) + un petit programme *main()* utilisant ces fonctions.

PS: <math.h> fournit 2 fonctions intéressantes : abs() et sqrt() ☺

Exercice 5 : solution

51

```
#include <iostream>
#include <math.h>

using namespace std;

struct point2d {
    float x,y;
};

struct point2d p,q;

float distManathan() {
    return ((float)abs(p.x - q.x) + abs(p.y - q.y));
}

float distEuclidienne() {
    float dx = p.x - q.x;
    float dy = p.y - q.y;

    return ((float)sqrt(dx*dx + dy*dy));
}

float distEchiquier() {
    float dx = abs(p.x - q.x);
    float dy = abs(p.y - q.y);

    if (dx > dy)
        return dx;
    return dy;
}

int main() {
    cout << "Saisir les coordonnees de p : ";
    cin >> p.x >> p.y;
    cout << "Saisir les coordonnees de q : ";
    cin >> q.x >> q.y;

    cout << endl << "Resultats : " << endl;

    cout << "distance de Manathan = " << distManathan() << endl;
    cout << "distance Euclidienne = " << distEuclidienne() << endl;
    cout << "distance de l'echiquier = " << distEchiquier() << endl;

    return 0;
}
```