Université Paris Diderot
M2 Modélisation Aléatoire

Programmation C++
Année 2013-2014

TP n°1

Page du cours : `http://www.liafa.univ-paris-diderot.fr/~carton/Enseignement/C++/`
N'oubliez pas d'indenter votre code !
Les exercices marqués d'un * sont (estimés) plus durs et sont à faire en dernier.

Exercice 1 (Mise en bouche)

1. Recopiez le texte suivant dans un fichier que vous nommerez exercice1.cpp. Puis compilez-le avec la commande `g++ exercice1.cpp -o premierExercice -ansi -Wall`. Ensuite exécutez-le avec la commande `./premierExercice`

```
#include<iostream>

using namespace std;

int main(int argc,char** argv)
{
        int d; // une variable entière d
        int i;

        cout<<"****Premier programme****"<<endl;
        cout<<"Saisissez un nombre puis appuyez sur entrée : ";
        cin >> d;

        i=0; // on met 0 dans i
        while(i<=15)
        {
                if( i % d == 0)
                {
                        cout << i << " est divisible par "<< d << endl;
                }
                else
                {
                        cout << i << " n'est pas divisible par "<< d << endl;
                }
                i=i+1;
        }
}
```

2. Modifiez le programme précédent afin qu'il affiche
`*****Division****` au lieu de `*****Premier programme******`
3. Modifiez le programme précédent pour qu'il traite les nombres de 1 à 100 au lieu de 0 à 15.
4. Modifiez le programme précédent afin qu'il demande à l'utilisateur un second nombre n et qu'il traite les nombres de 1 à n.
5. Modifiez le programme précédent afin qu'il affiche pour chaque nombre de 1 à n le reste et le quotient de la division par d des nombres de 1 à n.
Exemple :

```
*******Division*****
.
.
4=0*5+4
5=1*5+0
6=1*5+1
.
.
```

6. * Modifiez le programme précédent afin que les nombres d et n soient lus comme arguments de programme au lieu d'être entrés par l'utilisateur. Utilisez les variables `argv` et `argc` et la fonction `int atoi(char* s)` utilisable avec `#include<cstdlib>`.
   Par exemple :

```
xxxx@xxxx$ ./premierExercice 7 5
*******Division*****
1=0*5+1
2=0*5+2
3=0*5+3
4=0*5+4
5=1*5+0
6=1*5+1
7=1*5+2
```

Exercice 2 Écrivez une fonction `int pgcdr(int a,int b)` qui retourne le pgcd de deux nombres positifs a et b. La fonction doit être récursive. Si les deux nombres sont nuls alors elle doit renvoyer -1.

Exercice 3 Écrivez une fonction `int uniforme(int a,int b)` qui renvoie un nombre aléatoire uniforme compris entre a et b. Utiliser la fonction rand() qui retourne des entiers aléatoires compris entre 0 et `RAND_MAX`. Pour pouvoir utiliser `rand` vous devez ajouter l'instruction `#include<cstdlib>` au début de votre code.

Exercice 4 * Écrire un programme qui prend en argument un nombre entier et qui affiche sa décomposition en facteurs premier. Par exemple :

```
xxx@xxxx$ ./decomp 45
45=3*3*5
xxx@xxxx$
```

Exercice 5 (Sqrt)

1. Écrire une fonction qui prend en entrée n un nombre entier positif et renvoie la partie entière de la racine carré de ce nombre. La fonction doit faire environ $log(n)$ opérations. (Indice : la fonction racine carré est une fonction croissante et continue)
2. * Écrire une fonction `double sqrt(double r,double epsilon)` qui renvoie la racine carré d'un nombre "réel" r à *epsilon* près.

Exercice 6 On veut écrire une classe `Point` qui représente les points d'un plan.

1. La classe `Point` contiendra les attributs suivants :
   – `char nom;`

- `int abscisse;`
- `int ordonne;`
- `void translate(int x,int y);`
- `void affiche();`
- un constructeur `Point(int ab,int ord,char nom);`

Le nom du point est constitué d'un seul caractère. La méthode `affiche` affiche les coordonnés du points.
Par exemple : `A:(1,-3)`
La méthode `translate` translate le point selon le vecteur (x, y).

2. Ajoutez au constructeur un message qui indique lorsqu'il est appelé (et expérimentez ce constructeur).
3. Écrivez un destructeur de classe qui affiche un message lorsqu'il est appelé. (expérimentez-le).

Remarque :
- L' entête `#include<iostream>` doit être présente pour pouvoir utiliser des instructions comme `cout <<` et cin `cin >>`.
- L'entête `#include<cstdlib>` permet d'utiliser `atoi(..)` et `rand()`
- `using namespace std;` est une instruction qui permet de raccourcir les instructions `cout` et `cin`. Sans elle on doit écrire `std::cout` et `std::cin`.