MOOC Init. Prog. C++
Exercices semaine 4

Exercice 11 : prototypes (niveau 1, puis 2 pour les points 4 & 5)

> Cet exercice correspond à l'exercice n°12 (pages 31 et 210)
> de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Écrivez un programme `proto.cc` dans lequel vous définissez une fonction `demander_nombre()` respectant le prototype suivant :

```
int demander_nombre();
```

Cette fonction doit demander à l'utilisateur un entier et le renvoyer.

1. Placez la **définition** de la fonction avant le `main()`.
   Faites appel à la fonction dans le `main()` et affichez le résultat renvoyé.
2. Que se passe-t-il si l'on déplace la **définition** de la fonction `demander_nombre()` *après* le `main()` et que l'on recompile le programme ?
   (faites-le et vérifiez si le compilateur réagit comme vous vous y attendiez).
3. Ajoutez ensuite le **prototype** de la fonction `demander_nombre()` *avant* `main()` et recompilez le programme.
4. Modifiez maintenant la fonction pour qu'elle prenne 2 arguments : les bornes minimales et maximales entre lesquelles l'utilisateur doit donner le nombre.
   La fonction doit boucler tant que l'utilisateur ne donne pas un chiffre valide.
5. Raffinez encore le programme de sorte que si la borne maximale est inférieure ou égale à la borne minimale, elle ne soit pas prise en compte (c'est-à-dire que l'on peut entrer n'importe quel chiffre plus grand que la borne minimale).

Exercice 12 : passage des paramètres (niveau 1)

Cet exercice correspond à l'exercice n°13 (pages 31 et 212)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Écrivez le programme `echange.cc` dans lequel vous devez écrire (prototype + définition) une fonction `echange` qui :

- accepte deux arguments de type entier ;
- échange les valeurs de ces deux arguments.

Vous essayerez votre fonction avec le code `main` suivant :

```
int main()
{
    int i(10);
    int j(55);

    cout << "Avant: i=" << i << " et j=" << j << endl;
    echange(i,j);
    cout << "Après: i=" << i << " et j=" << j << endl;

    return 0;
}
```

Si votre fonction est correcte, le programme affichera:

```
Avant: i=10 et j=55
Après: i=55 et j=10
```

Exercice 13 : La fonction cosinus (définition et appel de fonction, niveau 2)

Cet exercice correspond à l'exercice n°14 (pages 32 et 213)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est d'écrire un programme `cos.cc` qui calcule une approximation de la fonction cosinus *cos(x)* (pour *x* dans [0, 2*pi])

Méthode

Pour calculer *cos(x)*, on utilisera son développement en série :

$$\cos(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{2i}}{(2i)!} = 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \dots$$

Tâches

Écrire les fonctions suivantes (qui seront ensuite appelées depuis le `main()`) :

- `double factorielle(int k)` qui calcule la factorielle d'un nombre *k* (et la retourne au format `double`)
  Utilisez pour cela une boucle `for`.
- `double somme_partielle(double x, int N)` qui calcule la somme partielle de la série au rang *N* (somme des *N* premiers termes de la série ci-dessus).

Ensuite, dans le `main()` : demander une valeur de *N* à l'utilisateur (vous pouvez utiliser ici la fonction `demander_nombre` de l'exercice 2), puis demander une valeur de *x* et tant que l'utilisateur n'entre pas 0.0, le programme calcule et affiche la valeur approchée du cosinus du nombre entré, en utilisant les fonctions précédentes..

Remarques :

- La fonction factorielle renvoie facilement des nombres très grands, qui dépassent les capacités de précision de l'ordinateur. Pour un ordre de grandeur, avec le type `double`, la plus grande valeur permise sera 170 (170! = 7.25742e+306).
  La valeur retournée pour des nombre supérieurs à ces valeurs limites sera `Infinity`.
  Évitez donc de tester le programme avec de trop grandes valeurs de N.
- Vous pouvez raffiner votre programme de sorte à garantir que *N* soit inférieur ou égal à 85 (par exemple en utilisant la borne maximale dans `demander_nombre`).
- Notez que pour afficher plus de décimales dans le résultat, vous pouvez utiliser `cout.precision`, par exemple : `cout.precision(12);`.

Exercice 14 : nombres de Fibonacci (fonctions récursives, niveau 1)

Cet exercice correspond à l'exercice n°33 (pages 82 et 254)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Les nombres de Fibonnaci sont définis par la suite :

```
F(0) = 0
F(1) = 1
F(n) = F(n-1)+ F(n-2) avec n>1
```

Le but de cet exercice est d'écrire un programme qui calcule la valeur de `F(n)` selon la définition récursive précédente.

Dans le fichier `fibonacci.cc`, prototypez puis définissez la fonction

```
int Fibonacci(int n)
```

qui calcule la valeur de `F(n)` de manière récursive (cette fonction devra donc faire appel à elle-même) sans utiliser de structure de boucle (for, while, etc...) et sans aucune variable locale.

Pour comparaison, voici une manière itérative (i.e. non récursive) de calculer les `n` premier termes de la suite :

```
int FibonacciIteratif(int n)
{
  int Fn(0);    // stocke F(i)  , initialisé par F(0)
  int Fn_1(Fn); // stocke F(i-1), initialisé par F(0)
  int Fn_2(1);  // stocke F(i-2), initialisé par F(-1)

  if (n > 0)
    for (int i(1); i <= n; ++i) {
      Fn   = Fn_1 + Fn_2;    // pour n>=1 on calcule F(n)=F(n-1)+F(n-
      Fn_2 = Fn_1;           // et on decale...
      Fn_1 = Fn;
    }
  return Fn;
}
```

Note : la méthode récursive est coûteuse en temps de calcul, ne la lancez pas pour des nombres trop élevés (disons supérieurs à 40).

Exemple de déroulement

```
Entrez un nombre entier compris entre 0 et 40 : 0
Méthode itérative :
    F(0) = 0
```

```
Méthode récursive :
    F(0) = 0
Voulez-vous recommencer [o/n] ? o
Entrez un nombre entier compris entre 0 et 40 : 1
Méthode itérative :
    F(1) = 1
Méthode récursive :
    F(1) = 1
Voulez-vous recommencer [o/n] ? o
Entrez un nombre entier compris entre 0 et 40 : 2
Méthode itérative :
    F(2) = 1
Méthode récursive :
    F(2) = 1
Voulez-vous recommencer [o/n] ? o
Entrez un nombre entier compris entre 0 et 40 : 3
Méthode itérative :
    F(3) = 2
Méthode récursive :
    F(3) = 2
Voulez-vous recommencer [o/n] ? o
Entrez un nombre entier compris entre 0 et 40 : 7
Méthode itérative :
    F(7) = 13
Méthode récursive :
    F(7) = 13
Voulez-vous recommencer [o/n] ? n
```
