semaine 4 Tutoriels
MOOC Init Prog C++

Les tutoriels sont des exercices qui reprennent des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice « classique ».

Semaine 4 : 1. Calcul de moyenne

> Cet exercice correspond à l'exercice « *pas à pas* » page 27 de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est de reprendre l'exemple du cours illustrant l'utilisation de fonctions : le calcul de la moyenne de 2 nombres.

Dans votre éditeur favori, commencez par ouvrir un nouveau fichier, par exemple `moyenne.cc` , puis préparez la « coquille vide » de base de votre programme :

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

int main()
{

    return 0;
}
```

Écrivez maintenant dans le `main`, les instructions nécessaires pour demander 2 notes à l'utilisateur et afficher la moyenne

Essayez de le faire par vous même sans regarder la solution. (solution page suivante).

Solution :

La façon que nous avions jusqu'ici de réaliser un tel programme ressemble à peut prêt à ceci :

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

int main ()
{
  double note1, note2;
  cout << "Entrez vos deux notes :" << endl;
  cin >> note1 >> note2;
  cout << "Votre moyenne est : "
       << (note1 + note2)/2.0 << endl;

  return 0;
}
```

Vous avez peut être utilisé des `int` au lieu des `double` ici. C'est très bien : cela va aussi.
Cela signifie juste que pour vous une note est un nombre entier, alors que pour moi elle peut être non entière (demi-point).

Notez que les parenthèses lors du calcul de la moyenne sont obligatoire puisque la division est prioritaire sur l'addition.

De plus, dans le cas où les notes sont des entiers, il est recommandé de diviser par 2.0 plutôt que par 2, pour assurer que le résultat sera un double et non pas un entier !
Revoir si nécessaire la leçon sur les expressions et opérateurs, si vous ne comprenez pas cette remarque.

Continuons l'exercice pour maintenant introduire des fonctions

Imaginons que ce programme rudimentaire soit amené à grandir et à être appliqué au calcul de plusieurs moyennes différentes (dans ce même programme).

Il peut alors être judicieux de sortir le calcul de la moyenne du corps de la fonction `main`, de manière à ce qu'il puisse être **réutilisé** par d'autres parties du programme.

Nous voulons donc définir une **fonction** qui prenne deux `double`s comme paramètres, et retourne le résultat sous forme de `double`.

Quel est alors le prototype de cette fonction ?
(sans regarder `;-)`) (solution page suivante).

Solution :

Le prototype de cette fonction est le suivant :

```
double moyenne (double nombre1, double nombre2);
```

Notez que les noms des paramètres sont ici arbitraires (et même optionnels, on pourrait en effet même écrire

```
double moyenne (double, double);
```

Par contre on doit absolument donner ces noms dans la *définition* de la fonction (qui va suivre)).

Écrivez maintenant la **définition** de cette fonction, c'est-à-dire décrivez dans son **corps** ce qu'elle doit faire.

(solution page suivante).

Solution :

La définition est à peine plus compliquée que le prototype. Il suffit en effet ici de recopier une ligne de notre code d'origine.

Le corps de la fonction calcule la moyenne des paramètres, et retourne le résultat du calcul. Bien que les deux solutions ci-dessous soit équivalentes, nous préférerons la deuxième qui est plus courte et plus efficace :

```
// solution 1
double moyenne(double nombre1, double nombre2)
{
  // déclaration et initialisation d'une variable
  double moy((nombre1 + nombre2) / 2.0);

  // la fonction renvoie la valeur de la variable moy
  return moy;
}

// solution 2
double moyenne (double nombre1, double nombre2)
{
  // la fonction renvoie directement la valeur
  // il n'est pas nécessaire d'utiliser une variable intermédiaire

  return (nombre1 + nombre2) / 2.0;
}
```

Notez qu'utiliser les mêmes noms de paramètres pour le prototype et pour la définition n'est pas obligatoire. Nous le recommandons cependant pour plus de clarté.

Conclusion

Il nous reste plus maintenant qu'à assembler les morceaux, c'est-à-dire faire **appel** à la fonction dans le programme principal `main`.

Cela se fait simplement en mettant le nom de la fonction à appeler avec les paramètres voulus à l'endroit où l'on veut utiliser la valeur qu'elle retourne. Par exemple ici :

```
cout << "Votre moyenne est : " << moyenne(note1, note2) << endl;
```

Souvenez-vous qu'une fonction doit avoir été prototypée avant de pouvoir être utilisée. Comme nous voulons utiliser cette fonction à l'intérieur de la fonction `main()` ; le prototype doit donc apparaître avant celle-ci dans le fichier. La définition par contre peut se trouver n'importe où après le prototype.
On arrive donc au résultat suivant : (solution page suivante).

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

// prototype
double moyenne (double nombre1, double nombre2);

int main()
{
  double note1, note2;
  cout << "Entrez vos deux notes :" << endl;
  cin >> note1 >> note2;
  cout << "Votre moyenne est : "
       << moyenne(note1, note2) << endl;

  return 0;
}

// définition
double moyenne(double nombre1, double nombre2)
{
  return (nombre1 + nombre2) / 2.0;
}
```

Semaine 4 : 2. Somme récursives

Cet exercice correspond à l'exercice « *pas à pas* » page 79 de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

On veut écrire le programme `somme.cc` qui calcule de façon récursive la somme des *n* premiers entiers (positifs).
« *De façon récursive* » signifie que le calcul de la somme pour *n* fait appel au calcul de la somme pour des nombres plus petits que *n*. Dans ce cas précis (somme), on utilisera le fait que la somme des *n* premiers entiers vaut celles des *n-1* premiers entiers, à laquelle on ajoute *n*.
Mathématiquement, on noterait : $S_n = S_{n-1} + n$

Cet exercice reprend pas à pas les différentes étapes pour y parvenir.

1. Ouvrez le fichier (vide) `somme.cc` dans votre éditeur favori.
2. Préparez la « coquille » vide de base accueillant votre programme :

   (solution page suivante).

Solution :

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

int main()
{
  return 0;
}
```

---

3. Prototypez la fonction `somme`.

   (solution page suivante).

Solution :

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

int somme(int n); // prototype

int main()
{
  return 0;
}
```

---

4. Définissez maintenant cette fonction (c.-à-d. écrivez son corps) : on utilisera ici le fait que la somme des *n* premiers entiers vaut celles des *n-1* premiers entiers, à laquelle on ajoute *n* comme expliqué plus haut.

   (solution page suivante).

Solution :

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

int somme(int n); // prototype

int main()
{
  return 0;
}

// définition de la fonction somme
int somme(int n)
{
  if (n <= 0) // condition d'arrêt
    return 0;
  else
    return (n + somme(n-1));
}
```

---

5. Il ne reste plus qu'à compléter le `main` pour demander un nombre et calculer la somme.

   (solution page suivante).

Solution :

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

int somme(int n); // prototype

int main()
{
      int n(0);
      cout << "Entrez un nombre entier : ";
      cin >> n;
      cout << "S(" << n << ")=" << somme(n) << endl;
  return 0;
}

// définition de la fonction somme
...
```

---

6. On peut pour finir raffiner pour boucler tant que l'utilisateur veux continuer.

   (solution page suivante).

Solution :

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

int somme(int n); // prototype

int main()
{
  char rep('n');

  do {

    int n(0);
    cout << "Entrez un nombre entier : ";
    cin >> n;
    cout << "S(" << n << ")=" << somme(n) << endl;

    do {
      cout << "Voulez-vous recommencer [o/n] ? ";
      cin >> rep;
    } while ((rep != 'o') and (rep != 'n'));

  } while (rep == 'o');

  return 0;
}

// définition de la fonction somme
...
```

Une dernière remarque (*complément*) : il serait en fait préférable d'utiliser ici des `unsigned int` plutôt que de simples `int`, vu que nous travaillons avec des entiers exclusivement positifs. Ce sujet (`unsigned int`) n'est abordé que dans les *compléments* du cours, pas dans la matière principale.
