Cours d'introduction à la programmation (en C++)
Fonctions

Jean-Cédric Chappelier
Jamila Sam
Vincent Lepetit

Faculté I&C

Notion de réutilisabilité

Pour l'instant : un programme est une séquence d'instructions

☞ mais sans **partage** des parties importantes ou utilisées plusieurs fois

Si une tâche, par exemple :

```
do {
  cout << "Entrez le nombre de points du joueur : ";
  cin >> nb;
} while ((nb < 0) or (nb > 100));
```

doit être exécutée à *plusieurs* endroits dans un plus gros programme

☞ recopie ? **NON !**

Bonne pratique : Ne *jamais dupliquer* de code en programmant :
Jamais de « copier-coller » !

☞ Ce que vous voudriez recopier doit être mis dans une **fonction**

Notion de réutilisabilité (2)

Pourquoi ne jamais dupliquer du code (copier/coller) :

Cela rend le programme

- inutilement long
- difficile à comprendre
- difficile à **maintenir** :
  reporter chaque modification dans *chacune* des copies

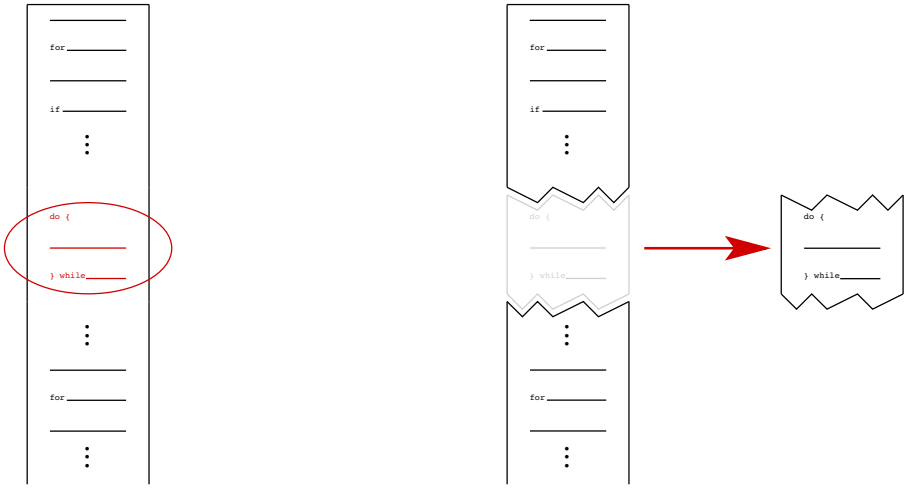
Tout bon langage de programmation fournit donc des moyens pour permettre la **réutilisation** de portions de programmes.

☞ les **fonctions**

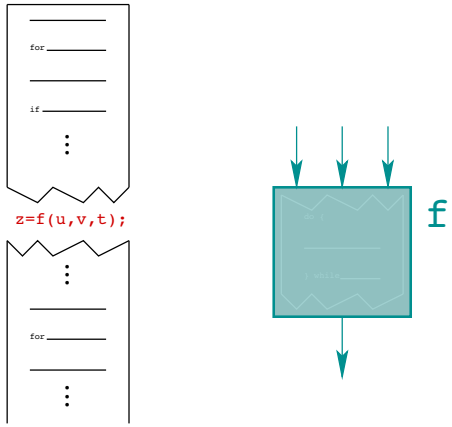
Exemple de fonction

```
int score (double points, double temps_jeu)
{
  int le_score(0);
  if (temps_jeu != 0.0) {
    le_score = 1000 * points / temps_jeu;
  }
  return le_score;
}
```

Notion de réutilisabilité : illustration



Notion de réutilisabilité : illustration



Fonction (en programmation)

fonction = portion de programme réutilisable ou importante en soi

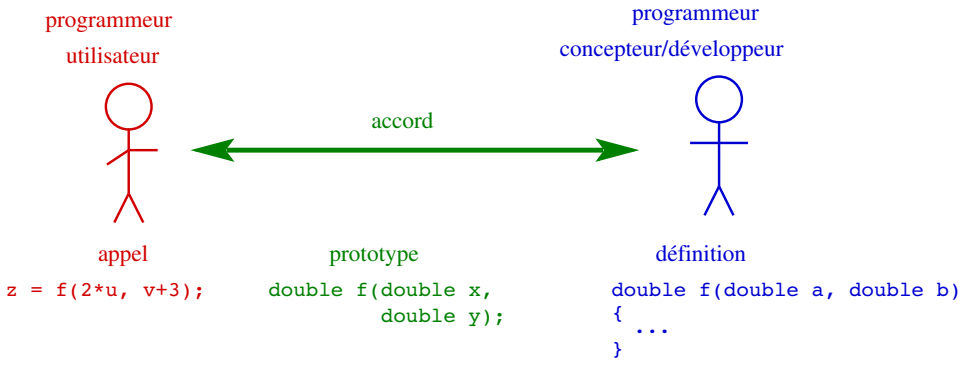
Plus précisément, une fonction est un objet logiciel caractérisé par :

- un corps : la portion de programme à réutiliser ou mettre en évidence, qui a justifié la création de la fonction ;
- un nom : par lequel on désignera cette fonction ;
- des paramètres : (les « *entrées* », on les appelle aussi « *arguments* ») ensemble de variables extérieures à la fonction dont le corps dépend pour fonctionner ;
- un type et une valeur de retour : (la « *sortie* ») ce que la fonction renvoie au reste du programme

L'utilisation de la fonction dans une autre partie du programme se nomme un appel de la fonction.

Les « 3 facettes » d'une fonction

- Résumé / Contrat (« prototype »)
- Création / Construction (« définition »)
- Utilisation (« appel »)



Exemple complet

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

prototype

```
double moyenne(double nombre_1, double nombre_2);

int main()
{
  double note1(0.0), note2(0.0);
  cout << "Entrez vos deux notes : " << endl;
  cin >> note1 >> note2;
  cout << "Votre moyenne est : "
       << moyenne(note1, note2) << endl;
  return 0;
}
```

appel

```
double moyenne(double x, double y)
{
   return (x + y) / 2.0;
}
```

définition