

Fonction & SOLID

C'est quoi une fonction ?

Les fonctions sont affichées en **jaune** dans l'IDE et permettent d'**effectuer une suite d'instructions**.

Exemple la fonction SayHelloWorld() sert à afficher "Hello World !" dans la console.

```
void SayHelloWorld()
{
    cout << "Hello World !";
}
```

Paramètre

Un paramètre est une **variable** de type **par défaut** ou **complexe** que l'on peut insérer dans la fonction afin de pouvoir **utiliser une ou plusieurs variables initialisées en-dehors de la fonction**.

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

void AskQuestion(const string& _question);

int main()
{
    const string& _question = "Quel âge as-tu ?";
    AskQuestion(_question);
}

void AskQuestion(const string& _question)
{
    cout << _question;
}
```

Ici la fonction AskQuestion() a pour paramètre une chaîne de caractère _question et affiche la chaîne de caractère "Quel âge as-tu ?" à l'utilisateur.

Fonction & SOLID

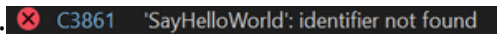
Utilisation

Pour utiliser une fonction **trois choses sont à savoir**.

La première est la **Définition** de celle-ci elle doit **se trouver en dehors de toute fonction** et doit comporter un **Type de Retour** (qu'il soit **par défaut** ou **complexe** peu importe), un **Nom de Fonction** et un **Body** contenant des **instructions**.

Ici la Fonction **se nomme** **SayHelloWorld()**, elle a pour **type de retour void** (ne renvoie pas de valeur) et **affiche "Hello World !"**.

```
void SayHelloWorld()
{
    cout << "Hello World !";
}
```

La deuxième est le **'Forward'** de la fonction, pour pouvoir appeler une fonction en C++ nous **devons informer l'IDE que la fonction peut être appelé plus tard sans quoi il ne va pas la reconnaître tel une fonction et afficher un message d'erreur**. 

Pour 'Forward', il suffit de placer la référence de la fonction en haut de votre fichier comme ci-contre :

```
#include <iostream>
using namespace std;
void SayHelloWorld();
void SayHelloWorld()
{
    cout << "Hello World !";
}
```

'Forward' de la Fonction
SayHelloWorld()

La troisième est l'**Appel** de la fonction. Nous pouvons **appeler une fonction depuis une autre fonction ou depuis la fonction main()**.

SayHelloWorld()
est **appelé depuis**
autre fonction
DoSomething()

```
#include <iostream>
void SayHelloWorld();
void DoSomething();

int main()
{
    DoSomething();
}

void DoSomething()
{
    SayHelloWorld();
}

void SayHelloWorld()
{
    cout << "Hello World !";
}
```

SayHelloWorld()
est **appelé**
depuis le main()

```
#include <iostream>
void SayHelloWorld();

int main()
{
    SayHelloWorld();
}

void SayHelloWorld()
{
    cout << "Hello World !";
}
```

Fonction & SOLID

Renvoie de Valeurs

Une fonction peut **renvoyer une valeur** grâce au **mot clé return**. Pour pouvoir renvoyer une valeur il faut spécifier le **type de valeur à renvoyer au type de retour de la fonction**. Attention **return** ne **peux renvoyer qu'une valeur à la fois** !

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

int Sums(const int _value1, const int _value2);

int main()
{
    const int _value1 = 2, _value2 = 5;
    const int _result = Sums(_value1, _value2);
}

int Sums(const int _value1, const int _value2)
{
    int _result = _value1 + _value2;
    return _result;
}
```

Ici la fonction Sums fait la somme de deux valeurs et renvoie le résultat. Le résultat renvoyer est 7.

Principe SOLID

S - Single Responsibility Principle

Une fonction doit être dédiée à une **SEUL utilité** et doit **pouvoir être réutilisable** afin d'**éviter les répétitions et surcharger le main()**. Sur les petits projets, on pourrait penser que ce principe est inutile mais sur des projets plus conséquents avec beaucoup de fonctions ce principe devient primordiale **pour que le code soit lisible et structuré**.

Avec fonction Sums() réutilisable que l'on appelle à n'importe quel moment

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

int Sums(const int _value1, const int _value2);

int main()
{
    int _result = Sums(5, 3);
    cout << _result;

    _result = Sums(4, 8);
    cout << _result;

    _result = Sums(15, 1);
    cout << _result;
}

int Sums(const int _value1, const int _value2)
{
    const int _result = _value1 + _value2;
    return _result;
}
```

Sans fonction Sums() aucune fonction réutilisable

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

int main()
{
    int _result = 5 + 3;
    cout << _result;

    _result = 4 + 8;
    cout << _result;

    _result = 15 + 1;
    cout << _result;
}
```

Fonction & SOLID

Convention

Afin d'avoir un code structuré et agréable à lire **chaque fonction doit avoir un nom qui précise exactement ce qu'il fait** sans quoi vous allez perdre du temps à savoir ce que font les fonctions et vous allez faire perdre du temps aux autres qui vont devoir essayer de comprendre ce que vous avez produit.



```
int AskQuestionAndRetrieveInt(const string& _question)
{
    int _result;
    cout << _question;
    cin >> _result;
    return _result;
}
```



```
int DoSomethingForMe(const string& _question)
{
    int _result;
    cout << _question;
    cin >> _result;
    return _result;
}
```