

Tâches professionnelles :	T7.2	Réaliser une maquette, un prototype logiciel/matériel
Compétences du référentiel	C4.4	Développer un module logiciel.
Savoirs - Savoir faire	S4.7	Langages de programmation C++

*Table des matières*

1	Déclaration des variables.....	2
2	Déclaration et Initialisation de variables .....	3
3	Déduction de Type : auto et decltype .....	4
4	Introduction à STRING .....	4
5	Un peu d'anglais... ..	6
6	Exercices .....	6

{ C / C++ }

# 1 DECLARATION DES VARIABLES

C++ est un langage fortement typé : chaque variable pour être déclarée avec son type avant sa première utilisation. Cela informe le compilateur de la taille à réserver en mémoire pour cette variable et comment interpréter sa valeur. La syntaxe pour déclarer une nouvelle variable en C++ est simple : nous allons simplement écrire le type suivi par le nom de la variable (son identifiant).

Par exemple :

```
1 int a;
2 float mynumber;
```

Ce sont deux déclarations de variables valides. Le premier déclare une variable de type `int` avec l'identifiant `a`. Le second déclare une variable de type `float` avec l'identifiant `mynumber`. Une fois déclarée, les variables `a` et `mynumber` peuvent être utilisés dans le reste du programme.

Si on doit déclarer plus d'une variable du même type, elles peuvent toutes être déclarées dans une déclaration unique en séparant leurs identifiants par des virgules.

Par exemple :

```
int a, b, c;
```

Cela déclare trois variables (`a`, `b` et `c`), chacune de type `int`, et cela a exactement la même signification que :

```
1 int a;
2 int b;
3 int c;
```

L'exemple de calcul mental donné au début du cours précédent pourrait se programmer avec le code C++ suivant.

```
1 // operating with variables
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
8     // declaring variables:
9     int a, b;
10    int result;
11
12    // process:
13    a = 5;
14    b = 2;
15    a = a + 1;
16    result = a - b;
17
18    // print out the result:
19    cout << result;
20
21    // terminate the program:
22    return 0;
23 }
```

4

Le résultat est 4.

Ne soyez pas inquiet si à part les déclarations de variables le reste du programme semble un peu étrange. La majeure partie sera expliquée plus en détail dans les prochains chapitres.

Nous n'allons pas encore saisir ce programme car il peut encore être amélioré.

## 2 DECLARATION ET INITIALISATION DE VARIABLES

Lorsque les variables dans l'exemple ci-dessus sont déclarées, elles ont une valeur indéterminée jusqu'à ce qu'elles se voient attribuer une valeur pour la première fois. Mais il est possible pour une variable d'avoir une valeur spécifique à partir du moment où elle est déclarée. Ceci est appelé *l'initialisation* de la variable.

En C ++, il y a trois façons d'initialiser les variables. Elles sont tous équivalentes et proviennent de l'évolution du langage au fil des ans.

La première, connue sous le nom **initialisation en style C** (parce qu'elle est héritée du langage C), se fait de la manière suivante: **type identifieur = initial\_value;**

Par exemple, pour déclarer une variable de type `int` appelée `x` et l'initialiser à une valeur de zéro à partir du moment où elle est déclarée, nous pouvons écrire :

```
int x = 0;
```

Une deuxième méthode, connue sous le nom **initialisation par constructeur**, introduite par le langage C ++, se fait de la manière suivante: **type identifieur (initial\_value);**

Par exemple:

```
int x (0);
```

Enfin, une troisième méthode, connue sous le nom d' **initialisation uniformisée**, a été introduite par la révision de la norme C ++, en 2011: **type identifieur {initial\_value};**

Par exemple:

```
int x {0};
```

Les trois façons d'initialiser des variables sont valides et équivalentes en C ++. :

```
1 // initialization of variables
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
8     int a=5;           // initial value: 5
9     int b(3);          // initial value: 3
10    int c{2};           // initial value: 2
11    int result;         // initial value undetermined
12
13    a = a + b;
14    result = a - c;
15    cout << result;
16
17    return 0;
18 }
```

6

Vous allez saisir et tester ce programme

### 3 DEDUCTION DE TYPE : AUTO ET DECLTYPE

Quand une nouvelle variable est initialisée, le compilateur peut comprendre automatiquement ce qu'est le type de la variable au moment de l'initialisation. Pour cela, il suffit d'utiliser **auto** comme spécificateur de type pour la variable.

```
1 int foo = 0;
2 auto bar = foo; // the same as: int bar = foo;
```

Ici, **bar** est déclarée comme ayant un type **auto** ; par conséquent, le type de **bar** est le type de la valeur utilisée pour l'initialiser : dans ce cas, il utilise le type de **foo**, qui est **int**.

Les variables qui ne sont pas initialisées peuvent également utiliser le type déduction avec le spécificateur **decltype** :

```
1 int foo = 0;
2 decltype(foo) bar; // the same as: int bar;
```

Ici, la variable **bar** est déclarée comme ayant le même type que **foo**.

**auto** et **decltype** sont des fonctionnalités puissantes récemment ajoutées au langage. Mais elles sont destinées à être utilisées soit lorsque le type ne peut pas être obtenue par d'autres moyens, ou lorsque leur utilisation améliore la lisibilité du code. Les deux exemples ci-dessus ne correspondent probablement à aucun de ces cas d'utilisation. En fait, ils diminuent probablement la lisibilité, puisque, lors de la lecture du code, il faut rechercher le type de **foo** pour savoir le type de **bar**.

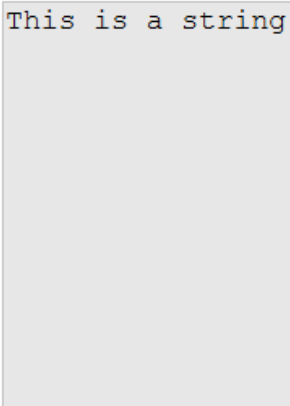
### 4 INTRODUCTION A STRING

Les types fondamentaux sont les briques les plus élémentaires. L'un des principaux points forts du langage C++ est son riche ensemble de types composés, construits à partir des types fondamentaux.

Un exemple de type composé est la classe **string**. Les variables de ce type sont capables de stocker des séquences de caractères (de **char**), tels que des mots ou des phrases. Une caractéristique très utile !

Une première différence avec les types de données fondamentales est que, pour déclarer et utiliser des objets (des variables) de ce type **string**, le programme doit inclure l'en-tête où le type est défini dans la bibliothèque standard ( en-tête **<string>** )

```
1 // my first string
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main ()
7 {
8     string mystring;
9     mystring = "This is a string";
10    cout << mystring;
11    return 0;
12 }
```



Comme vous pouvez le voir dans l'exemple précédent, les chaînes peuvent être initialisées avec toute chaîne littérale valide, comme des variables de type numériques peuvent être initialisées avec toute valeur numérique littérale valide.

Comme pour les types fondamentaux, tous les formats d'initialisation sont valables avec les chaînes de caractères (string) :

```
1 string mystring = "This is a string";
2 string mystring ("This is a string");
3 string mystring {"This is a string"};
```

La chaîne de classe est un *type composé*. Comme vous pouvez le voir dans l'exemple ci-dessous, les *types composés* sont utilisés de la même manière que les *types fondamentaux* : la même syntaxe est utilisée pour déclarer des variables, les initialiser, modifier leur valeur, etc.

Vous allez saisir et tester ce programme sur le site de <http://cpp.sh/> ou <https://repl.it/languages/cpp11>

<pre>1 // my first string 2 #include &lt;iostream&gt; 3 #include &lt;string&gt; 4 using namespace std; 5 6 int main () 7 { 8     string mystring; 9     mystring = "This is the initial string content"; 10    cout &lt;&lt; mystring &lt;&lt; endl; 11    mystring = "This is a different string content"; 12    cout &lt;&lt; mystring &lt;&lt; endl; 13    return 0; 14 }</pre>	<pre>This is the initial string content This is a different string content</pre>
--	--

En tant que type composé l'objet **string** possède un ensemble de méthodes (d'outils) qui facilitent la manipulation des chaînes de caractères (voir [http://en.cppreference.com/w/cpp/string/basic\\_string](http://en.cppreference.com/w/cpp/string/basic_string) ).

On peut transformer un nombre en string :

**to\_string(12)** va transformer le nombre 12 en chaîne "12". **to\_string** fonctionne avec des doubles, etc.

On peut aussi concaténer des string comme ceci :

**string maChaine = "La valeur de la température est : " + to\_string(temp) + "°C " ;**

Vous allez saisir et tester le programme suivant :

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main()
7 {
8     string maChaine = "Hasparren";
9     cout << maChaine.length() << endl;
10    cout << maChaine.front() << endl;
11    cout << maChaine.back() << endl;
12    cout << maChaine.at(0) << endl;
13    cout << maChaine.at(1) << endl;
14    cout << maChaine.at(2) << endl;
15
16    system("pause"); // dans windows garde la fenêtre console ouverte en fin d'exécution
17    return 0;
18 }
```

Saisissez sur les lignes ci-dessous le résultat du programme et déduire l'utilité de la méthode (outil) fournie par **string**

- ..... : length sert à : .....
- ..... : front sert à : .....
- ..... : back sert à : .....
- ..... : at(2) sert à : .....

## 5 UN PEU D'ANGLAIS...

Chaîne de caractères

String

En tête

Header

## 6 EXERCICES

Ecrire un programme qui utilise deux variables de type string appelées monNom et monPrenom.

En même temps que vous les déclarez il faut les initialiser avec vos nom et prénom (par la méthode uniformisée).

A partir de là le programme travaille sur ces deux variables.

Votre programme doit ensuite produire la sortie suivante (avec vos nom et prénom à vous bien sûr).

```
Bonjour Arnaud Etchegoyhen  
Etchegoyhen ça a 11 lettres  
La 4ème lettre de votre nom est : h  
Vos initiales sont : AE
```