

# Structures de base [bs] Résumé de cours

Université de Haute Alsace

Unisciel  algoprogram  Version 12 mai 2018

## Table des matières

<b>1 C++ - Résumé de cours</b>	<b>1</b>
1.1 Le langage . . . . .	1
1.2 Variables, types et valeurs . . . . .	1
1.3 Déclarations . . . . .	2
1.4 Structure générale . . . . .	3
1.5 Interactions avec l'extérieur . . . . .	3
1.6 Expressions algébriques . . . . .	4
1.7 Affectation interne . . . . .	5

## 1 C++ - Résumé de cours

### 1.1 Le langage



#### Identifiant

Séquence de lettres (A...Z, a...z), de chiffres (0...9) ou du caractère souligné (\_).  
Il doit commencer par une lettre ou un souligné.



#### La casse

Le langage est *case-sensitif* et les accentués ne sont pas des lettres anglophones.  
Ainsi `cout`, `Cout` et `COU`T réfèrent trois identifiants différents et `coût` n'est pas autorisé.

### 1.2 Variables, types et valeurs



#### Variable

Élément informatique qu'un programme peut manipuler.  
Décrite par :

- Un **identifiant** unique qui la désigne.

- Un **type** qui définit de quel « genre » est l'information associée.
- Une **valeur** qui doit respecter le type.



### Types intégrés

Domaine	Algorithmique	Équivalent C++
$\mathbb{Z}$	Entier	int
$\mathbb{R}$	Réel	double
$\mathbb{B}$	Booléen	bool
$\mathbb{A}$	Caractère	char
$\mathbb{T}$	Chaîne	#include <string> string



### C++ : Le type Chaîne

Ce n'est pas un type élémentaire.

On utilisera néanmoins le type `string` qui le défini.



### Littéraux

- **Entier** : Suite de chiffres éventuellement préfixé par un signe (+ ou -). S'il y a deux chiffres ou plus, il **ne doit pas** commencer par 0.
- **Réel** : S'écrit en notation décimale ou en notation scientifique.
- **Booléen** : Identifie le **Vrai** (mot-clé `true`) et le **Faux** (mot-clé `false`).
- **Caractère** : Se place entre quotes (').
- **Chaîne** : Se place entre guillemets (").

## 1.3 Déclarations



### Déclaration de variables

```
TypeVar nomVar;  
TypeVar nomVar1, nomVar2, ...;
```



### Initialisation d'une variable

```
TypeVar nomVar = valeur; // héritée du C  
TypeVar nomVar(valeur); // notation objet du C++
```



### Définition de constante

```
const TypeConst nomConst = expression; // notation impérative  
const TypeConst nomConst(expression); // notation objet
```

## 1.4 Structure générale

C/C++

### C99/C++ : Commentaire orienté ligne

```
... // rend le reste de la ligne non-exécutable
```

C/C++

### Commentaire orienté bloc

```
/*
rend le code entouré non exécutable...
(hérité du C)
*/
```

C/C++

### Bloc

```
{
  instruction1;
  instruction2;
  ...
}
```

C++

### Structure générale

```
#include <des_trucs_utiles>
using namespace std;
déclaration_des_objets_globaux
déclarations_et_définitions_de_fonctions_utiles
int main()
{
  corps_du_programme
}
```



### C/C++ : La fonction main

Quelques règles :

- Respectez la casse (m minuscule) ainsi que les parenthèses.
- Chaque programme possède une fonction `main()`.
- En l'absence de fonction `main()`, le programme ne démarre pas.

## 1.5 Interactions avec l'extérieur

C++

### Pour les utiliser

```
#include <iostream>
```

C++

### Saisie de données

```
cin >> nomVar1 >> nomVar2 >> ... >> nomVarN;
```



## Affichage de résultats

```
cout << expr1 << expr2 << ... << exprN; // SANS retour de ligne
cout << expr1 << ... << exprN << endl; // AVEC retour de ligne
```

## 1.6 Expressions algébriques



### Expression, opérandes, opérateurs

Éventuellement accompagnés de parenthèses, une **expression** est une séquence « bien formée » (au sens de la syntaxe) d'**opérandes** (valeurs littérales, variables ou expressions) et d'**opérateurs** destinée à l'évaluation.



### Opérateurs arithmétiques

Opérateur Mathématique	Signification	Équivalent C/C++
+	(unaire) valeur	+a
-	(unaire) opposé	-a
+	addition	a + b
-	soustraction	a - b
*	multiplication	a * b
/	division décimale	a / b
div	division entière	a / b
mod	modulo (reste de la division entière)	a % b



### C/C++ : Que vaut a / b ?

La division s'effectue sur :

- $\mathbb{N}$  : si **a et b** sont entiers (division entière)
- $\mathbb{R}$  : si **a ou b** sont réels (division réelle)



### Ordre de priorité des opérateurs arithmétiques

Comme en mathématique :

1. Les opérateurs unaires (+, -) (priorité la plus élevée)
2. L'opérateur d'exponentiation () (s'il existe)
3. Les opérateurs multiplicatifs (\*, /, div, mod)
4. Les opérateurs additifs (+, -) (priorité la plus basse)

La règle d'associativité s'applique en cas d'ambiguïté entre opérateurs du même ordre de priorité.



### Règle de promotion

Pour qu'une opération numérique binaire (+, -, \*, /) puisse s'effectuer, il faut que ses deux opérandes soient du **même type** ou d'un **type compatible**. Lorsque ce n'est pas le cas, il y a **promotion** de l'opérande de type le plus faible vers le plus grand.

**C/C++ : Écriture des littéraux réels**

Écrivez-les explicitement avec un point pour les distinguer des littéraux entiers.

**Fonctions mathématiques**

Elles agissent sur des paramètres à valeurs réelles et donnent un résultat réel.

**Pour les utiliser**

```
#include <cmath>
```

## 1.7 Affectation interne

**Affectation interne**

```
nomVar = expression;
```