

R101 - Amphi05 - C++

Alain Casali

Aix Marseille Univ



1 Avant Propos

- Langage de programmation
- Problème du développeur
- La documentation
- Le C++, c'est quoi ?

2 Un premier programme

3 Les commentaires

4 Déclaration, affectation, bloc

5 Entrées / Sorties

6 Opérateur identité et différence

7 Schéma alternatif simple

8 Schéma alternatif complexe

9 Le type booléen

10 Les types entiers

11 Les types réels

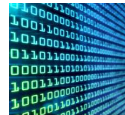
Langage de programmation

En informatique, un langage de programmation est une notation conventionnelle destinée à formuler des **algorithmes** et produire des **programmes informatiques** qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est fait d'un **alphabet**, un **vocabulaire**, des **règles de grammaire**, et des **significations**.

Source : http://fr.wikipedia.org/wiki/Langage_de_programmation



Fichiers



Executable

Problème du développeur (1)



C'est une roue !



On ne re-développe **JAMAIS** une fonctionnalité si elle est déjà fournie par le langage (sauf pour des raisons pédagogiques).

Problème du développeur (2)

Exemple : Trier un tableau d'entiers

- Il existe une trentaine d'algorithmes permettant de trier un tableau d'entiers avec des performances différentes.
- Un des algorithmes les plus efficaces est le tri rapide

Taille du tableau	Temps C++	Temps mon implémentation
1 000	~ 0s	~ 0s
10 000	~ 0s	~ 0s
100 000	~ 0s	~ 0s
1 000 000	~ 0s	~ 13s
10 000 000	~ 6s	~ 1200s

La documentation

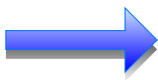
Elle se lit :

- En ligne (elle est à jour) ;
- En anglais (on évite les erreurs de traduction, si traduction il y a) ;

Elle comporte des exemples.

Objectifs

- Début janvier :
 - Être capable de lire une documentation en anglais ;
 - De la comprendre ;
 - D'adapter les exemples fournis à votre problématique.
- Fin du BUT : Être capable d'écrire une documentation technique en anglais ;



Tous les noms de fonctions, de variables, doivent être en anglais !

Le C++, cest quoi ?

- 1 Un langage de programmation ;
- 2 Une surcouche (partiellement) objet du C.

Dernière norme du C++ date de 2020.

Plusieurs compilateurs disponibles :

- g++ (support complet)
- clang (support complet)
- VS2017 (support partiel)

En C++ :

- Tout est appel de fonction ou de méthode (maintenant) ;
- Tout est flux ;
- Vive les références (maintenant) et les pointeurs (S3) ;

- 1 Avant Propos
- 2 Un premier programme**
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

Un premier programme (1)

Quel que soit le langage de programmation utilisé, le premier programme est toujours d'afficher, soit :

- Sur un terminal (une console) ;
- Sur une page web ;
- Sur une interface graphique.

la chaîne de caractères "Hello world!" (modulo quelques petits arrangements)

Un premier programme (2)

```
1  /**
2   *
3   * @file   Hello.cxx
4   *
5   * @author A. Casali
6   *
7   * @date   12/09/2013
8   *
9   **/
10
11 #include <iostream>
12 using namespace std;
13
14 int main ()
15 {
16     ^~^lcout << "Hello␣World!" << endl;
17     ^~^lreturn 0;
18 }
```

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires**
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

Les commentaires

- 1 Commentaire sur une unique ligne : utilisation de `//`

```
//this is a one line remark
```

- 2 Commentaire sur plusieurs lignes : utilisation de `/* */`

```
/* this  
   is a drawn out  
   explanation  
*/
```

- 3 Autre possibilité :

```
/*this is also a one line remark*/
```

Plus de détail : <http://www.stack.nl/~dimitri/doxygen/>

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 **Déclaration, affectation, bloc**
 - Déclaration de variables
 - Affectation
 - Cas des variables constantes
 - Déclaration et initialisation à la volée
 - Bloc et portée de variable
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

Déclaration de variables

Modèle général algorithmique :

```
declarer nomDeVariable : Type ;
```

Modèle général C++ :

```
type varIdent ; //varIdent (Variable Identifier)
```

Exemple :

```
int i ;
```

Affectation

Modèle général algorithmique :

```
nomDeVariable ← valeur ;
```

Modèle général C++ :

```
varIdent = value ;
```

Exemple :

```
i = 10 ;
```

Cas des variables constantes

Modèle général algorithmique :

```
declarer KnomDeVariable : constante type ← valeur ;
```

Modèle général C++ :

```
const type KvarIdent = valeur ;
```

Exemple :

```
const int Ki = 10 ;
```



Tous les noms des constantes doivent commencer par la lettre 'K'

Déclaration et initialisation à la volée

Modèle général algorithmique :

```
declarer nomDeVariable : type ← Valeur ;
```

Modèle général C++ :

- 1 `type varIdent = value ;`
- 2 `type varIdent (value) ;`
- 3 `type varIdent {value(s)} ;`

Exemple :

```
int i = 10 ;  
int j (5) ;  
int k {3} ;
```



Seule la troisième forme permet la déclaration et initialisation à la volée des tableaux.

Bloc et portée de variable

Définition : **bloc**

Un **bloc** est une suite d'instructions entre { }

Propriété : **portée de variable**

Une variable n'existe que dans le bloc dans laquelle est déclarée.

Exemple :

```
{
~ type i ;
~ li = value1 ;
~ ...
~ {
~ ~ type j (value2);
~ ~ //i et j existent
~ ~ li = value3 ;
~ ~ lj = value4 ;
~ }
~ //seul i existe
~ li = value5 ;
~ lj = value6 ; //<- erreur de compilation
}
```

Bloc interne

Bloc externe

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
 - Saisie clavier
 - Affichage écran
 - Passage à la ligne lors d'un affichage console
 - Compression des sorties écran
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

Saisie clavier

Modèle général algorithmique :

```
saisir (nomDeVariable);
```

Modèle général C++ :

```
cin >> varIdent;
```

cin signifie console input

Le symbole » est appelé un **extracteur**.

Exemple :

```
int i;  
cin >> i; //10
```

Prérequis pour utiliser le clavier :

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

Utilisation de la bibliothèque qui gère les flux (stream) d'entrées / sorties (input / output)

Affichage écran

Modèle général algorithmique :

```
afficher (unLiteral);
afficher (nomDeVariable);
```

Modèle général C++ :

```
cout << literalPattern;
cout << varIdent;
```

`cout` signifie console `output`

Le symbole « est appelé un **injecteur**.

Exemple :

<code>cout << 10;</code>	Littéral entier	10
<code>cout << "une_jolie_chaine";</code>	Littéral chaîne de	une jolie chaîne
<code>int i;</code>	caractères	
<code>i = 10;</code>		
<code>cout << i;</code>	Variable	10

Prérequis pour utiliser la console :

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

Utilisation de la bibliothèque qui gère les flux (stream) d'entrées / sorties (input / output)

Passage à la ligne lors d'un affichage console

Modèle général algorithmique :

```
ligne_suivante ;
```

Modèle général C++ :

```
cout << endl ;
```

`endl` signifie **end line**

Le symbole `endl` est appelé un **identificateur**.

Exemple :

```
cout << i ;  
cout << endl ;
```

10



Compression des sorties écran

Il est possible d'injecter plusieurs informations de type différents et donc de compacter l'écriture.

Exemple :

- ```
cout << "Valeur de i : ";
cout << i;
cout << endl;
```
- ```
cout << "Valeur de i : "  
  << i  
  << endl;
```



- ❶ Absence du caractère ';' à la fin de la ligne;
- ❷ Tous les injecteurs sont alignés.

- ```
cout << "Valeur de i : " << i << endl;
```

1 Avant Propos

2 Un premier programme

3 Les commentaires

4 Déclaration, affectation, bloc

5 Entrées / Sorties

6 Opérateur identité et différence

- Opérateur identité
- Opérateur différence

7 Schéma alternatif simple

8 Schéma alternatif complexe

9 Le type booléen

10 Les types entiers

11 Les types réels



# Opérateur identité

## Modèle général algorithmique :

nomDeVariable **vaut** Valeur

## Modèle général C++ :

varIdent **=** value

## Exemple :

i **=** 10

Ou mieux

10 **=** i



On ne compare que des variables et / ou des littéraux du même type.

# Opérateur différence

## Modèle général algorithmique :

nomDeVariable `ne_vaut_pas` Valeur

## Modèle général C++ :

varIdent `!=` value

## Exemple :

i `!=` 10

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
  - Schéma alternatif sans condition sinon
  - Schéma alternatif avec condition sinon
  - Schéma alternatif en cascade
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

# Schéma alternatif sans condition sinon

## Modèle général algorithmique :

```
si (condition)
 instruction1 ;
 instruction2 ;
 ...
fsi
```

## Modèle général C++ :

```
if (condition)
{
 instruction1 ;
 instruction2 ;
 ...
}
```

- condition est une expression booléenne ;
- '{' marque le début d'un bloc d'instruction(s) ;
- '}' marque la fin de ce bloc.



Toutes les instructions à l'intérieur d'un même bloc sont alignées

# Schéma alternatif avec condition sinon (1)

## Modèle général algorithmique :

```
si (condition)
 instruction1;
 instruction2;
 ...
sinon
 instruction3;
 instruction4;
 ...
fsi
```

## Modèle général C++ :

```
if (condition)
{
 instruction1;
 instruction2;
 ...
}
else
{
 instruction3;
 instruction4;
 ...
}
```

# Schéma alternatif avec condition sinon (2)

## Exemple :

```
verifierVie ;
verifierArmes ;

si (toutEstOK)
 attaquerBoss ;
~| ...
sinon
 prendrePotion ;
~| ...
fsi
partir ;
```



```
checkLife ();
checkWeapons ();

if (everythingIsOK)
{
 attackBoss ();
~| ...
}
else
{
 takePotion ();
~| ...
}
leave ();
```

# Schéma alternatif en cascade (1)

Modèle général algorithmique :

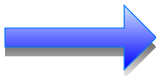
```
si (exprLog1)
 sequ1;
sinon_si (exprLog2)
 sequ2;
sinon_si (exprLog3)
 sequ3;
sinon //tous les autres cas
 sequ4;
fsi
```

```
if (logExp1)
{
 sequ1;
}
else if (logExp2)
{
 sequ2;
}
else if (logExp3)
{
 sequ3;
}
else
{
 sequ4;
}
```

# Schéma alternatif en cascade (2)

## Exemple :

```
si (peuDeVie)
 prendreUnePotion ;
sinon_si (jeSuisUnGuerrier)
 attaquerAvecEpee ;
sinon_si (jeSuisUnMage)
 lancerUnSort ;
sinon
 partir ;
fsi
```



```
if (fewOfLife)
{
 takeAPotion ();
}
else if (IAmAWarrior)
{
 attackWithSword ();
}
else if (IAmAWizard)
{
 castASpell ();
}
else
{
 leave ();
}
```



- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels

# Schéma alternatif complexe (1)

| Condition logique en algo. | Equivalence en C++ | C++14 |
|----------------------------|--------------------|-------|
| ET                         | &                  | &     |
| OU                         |                    |       |
| ET_ALORS                   | &&                 | and   |
| OU_SINON                   |                    | or    |

Exemple :

```
si (beaucoupDeVie OU_SINON jeSuisUnGuerrier)
 Attaquer ;
fsi
```

se traduit par :

```
if (fullOfLife || IAmAWarrior)
{
 attack ();
}

if (fullOfLife or IAmAWarrior)
{
 attack ();
}
```

## Schéma alternatif complexe (2)

### Exemple :

```
si (peuDeVie ET_ALORS jeSuisUnGuerrier)
 attaquer ;
fsi
```

```
if (fewOfLife && IAmAWarrior)
{
 attack ();
}
```

```
if (fullOfLife and IAmAWarrior)
{
 attack ();
}
```



- Les règles de propagation de l'opérateur de négation sont les mêmes qu'en algorithmique (amphi2 T11 -> T14);
- Les tableaux de vérité sont les mêmes qu'en algorithmique (amphi2 T15 & T16).

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
  - Identificateur
  - Valeurs
  - Opérations (opérateurs booléens)
  - Opérations (d'identité)
  - Opérateur de négation
- 10 Les types entiers
- 11 Les types réels

# Identificateur

```
bool
```

## Valeurs

```
true false
```

## Opérations (opérateurs booléens)

Produisent un booléen

Voir transparents précédents

```
bool toBe ;
bool notToBe ;
bool question ;
...

toBe = false ;
notToBe = !toBe ; // true
question = toBe | notToBe ; // always true!
```

# Opérations (d'identité)

Produisent un booléen

== !=

Exemple :

```
if (toBe == notToBe)
{
 cout << "WTF!";
}
```

## Opérateur de négation

Produit un booléen

Modèle général algorithmique :

```
si (NON condition)
 instruction1;
 instruction2;
...
fsi
```

Modèle général C++ :

```
if (!condition)
{
 instruction1;
 instruction2;
 ...
}
```

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
  - Identificateur
  - Valeurs
  - Opérations
  - Opérations (de comparaison)
  - Opérations (d'identité)
  - Les différents types d'entiers
- 11 Les types réels

# Identificateur

`int`

## Valeurs

Sous-ensemble des entiers mathématiques

## Opérations

Produisent un entier

`+` `-` `*` `/` `%`



- Troncature du reste de la division ;
- L'opérateur modulo (`%`) renvoie le reste de la division entière.

## Exemple :

|                               |   |                                   |   |
|-------------------------------|---|-----------------------------------|---|
| <code>int x;</code>           |   | <code>x = (x + 2) / 4 * 3;</code> | 0 |
| <code>cout &lt;&lt; x;</code> | 0 | <code>cout &lt;&lt; x;</code>     |   |
| <code>x = 1;</code>           |   | <code>x = 1;</code>               |   |
| <code>cout &lt;&lt; x;</code> | 1 | <code>x = (x + 2) * 3 % 4;</code> |   |
|                               |   | <code>cout &lt;&lt; x;</code>     | 0 |



# Priorité des opérateurs

## Opérations (de comparaison)

Produisent un booléen

&lt;   &lt;=   &gt;   &gt;=

## Opérations (d'identité)

Produit un booléen

==   !=

Exemple :

```

int nb1;
nb1 = 12;
int nb2;
nb2 = 5;
int nb3;
nb3 = nb1 / nb2; // nb3 vaut 2

if (4 == nb3)
{
 cout << "nb3 vaut 4";
}
else if (nb3 >= 2)
{
 cout << ("nb3 >= 2");
}

```

# Les différents types d'entiers

| Type                              | Min                        | Max                        | #octets |
|-----------------------------------|----------------------------|----------------------------|---------|
| <code>int</code>                  | -2 147 483 648             | 2 147 483 647              | 4       |
| <code>unsigned [int]</code>       | 0                          | 4 294 967 295              | 4       |
| <code>short [int]</code>          | -32 768                    | 32 767                     | 2       |
| <code>unsigned short [int]</code> | 0                          | 65 535                     | 2       |
| <code>long long [int]</code>      | -9 223 372 036 854 770 000 | 9 223 372 036 854 770 000  | 8       |
| <code>unsigned long</code>        | 0                          | 18 446 744 073 509 551 615 | 8       |
| <code>long [int]</code>           |                            |                            |         |

- 1 Avant Propos
- 2 Un premier programme
- 3 Les commentaires
- 4 Déclaration, affectation, bloc
- 5 Entrées / Sorties
- 6 Opérateur identité et différence
- 7 Schéma alternatif simple
- 8 Schéma alternatif complexe
- 9 Le type booléen
- 10 Les types entiers
- 11 Les types réels
  - Identificateur
  - Valeurs
  - Opérations
  - Opérations (d'identité)
  - Les différents types de réels
  - Égalité entre de deux réels

# Identificateur

`float`

## Valeurs

Sous-ensemble des réels mathématiques

## Opérations

Produisent un réel

+ - \* /



L'opérateur division (/) renvoie la division réelle.

Produisent un booleen

< <= > >=

## Opérations (d'identité)

Produit un booleen

= !=

# Les différents types de réels

| Type                       | #octets |
|----------------------------|---------|
| <code>float</code>         | 4       |
| <code>double</code>        | 8       |
| <code>double double</code> | 12      |

Plus il y a d'octets, plus la précision est grande (voir cours d'architecture).

# Égalité entre de deux réels



On ne compare jamais deux réels avec l'opérateur d'égalité, même si mathématiquement le résultat est juste !

Exemple : La comparaison suivante a de fortes chances d'être fausse :

```
float four ;
four = 4.0 ;
float squareOfFour ;
squareOfFour = ... // Computation of four
if (2.0 == squareOfFour)
{
 cout << "the_value_of_the_square_root_of_4_is_2" << endl ;
}
```

Il est préférable de remplacer le test d'égalité par :

```
float eps = 0.000001 ;
if (abs (2.0 - squareOfFour) < Eps)
{
 cout << "the_value_of_the_square_root_of_4_is_2" << endl ;
}
```