



# Module C++

FIP 1A CNAM

# Déroulement du module

- 7 modules d'une demie journée (4h)
- 4 premiers modules :
  - 1 à 2h théorie
  - 1 à 2h de TP en salle machine
- Les 3 suivants :
  - Travail en groupe
  - Code review en privé

# Critères d'évaluation

- Notes des Travaux Pratiques individuels
- Note des Travaux Pratiques en groupe
- Note du projet final (coeff. 2)

# Historique du C++

- « C with classes » : 1979
- « *The C++ Programming Language* » publié en 1985 par [Bjarne Stroustrup](#)
- *C++ 98 (1998)*
- *C++11 (2011): Refonte du langage vers le C++ « Moderne »*

# Le C++, une fédération de langages:

- Le C:
  - Préprocesseur,
  - Organisation du code en blocs,
  - types de base, tableaux, pointeurs...

# Le C++, une fédération de langages:

- Les classes :
  - Héritage, polymorphisme,
  - Abstraction,
  - Encapsulation, fonctions virtuelles,
  - Constructeurs, destructeurs...

# Le C++, une fédération de langages:

- La STL (« Standard Template Library »):
  - Comporte des Conteneurs, Itérateurs, des algos spécifiques,
  - A ses propres conventions...

# Le C++, une fédération de langages:

- Les Templates :
  - Programmation générique,
  - Paradigme différent du C++ « classique »



# Les points forts du C++

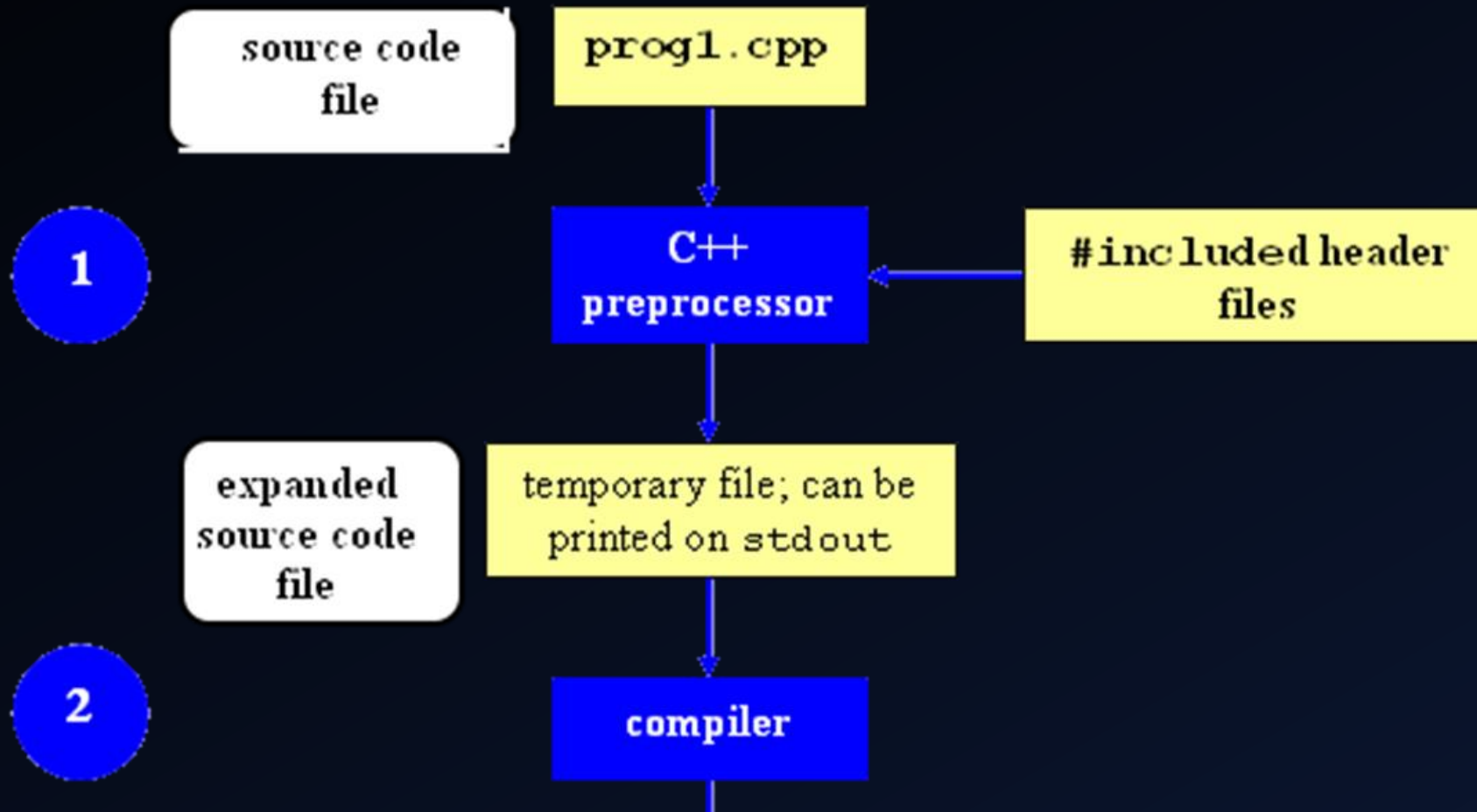
- Performances:
  - 3D,
  - Programmes « Temps réel »,
  - Calculs énergivores
- Portabilité sur les principales plate-formes,
- Code machine « bas-niveau », organisation du code « haut-niveau »



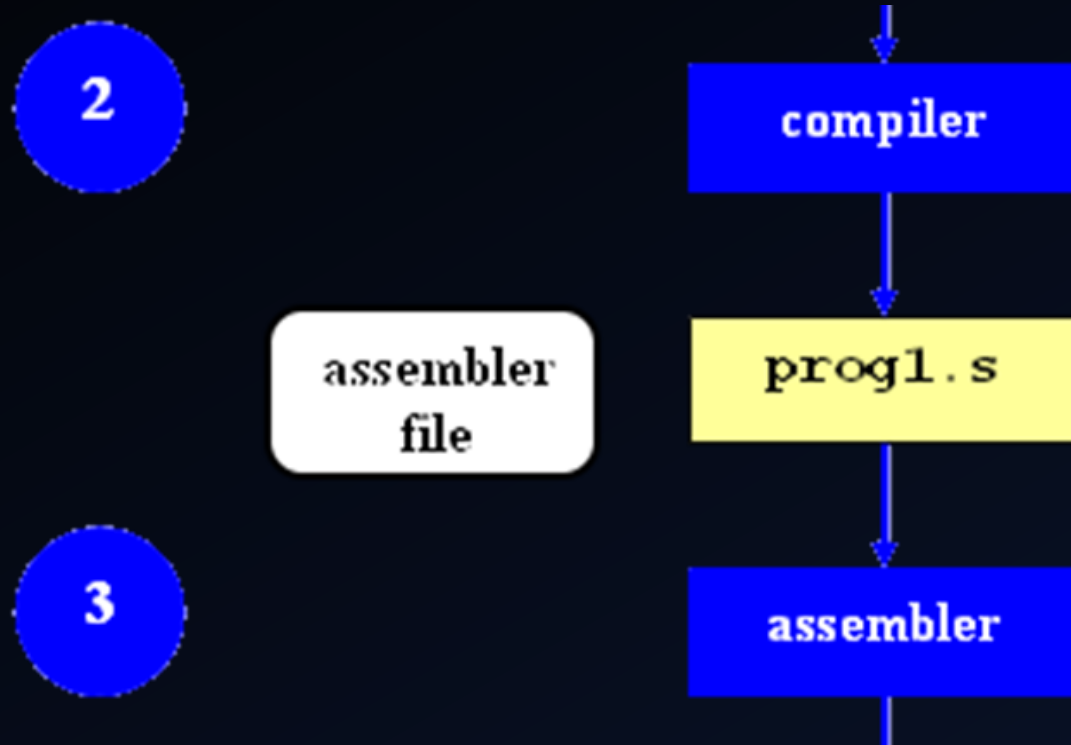
# Programmer en C++

## LES BASES THÉORIQUES

# Les étapes de la construction d'un programme



# Les étapes de la construction d'un programme



# Les étapes de la construction d'un programme

3

object code  
file

assembler

prog1.o

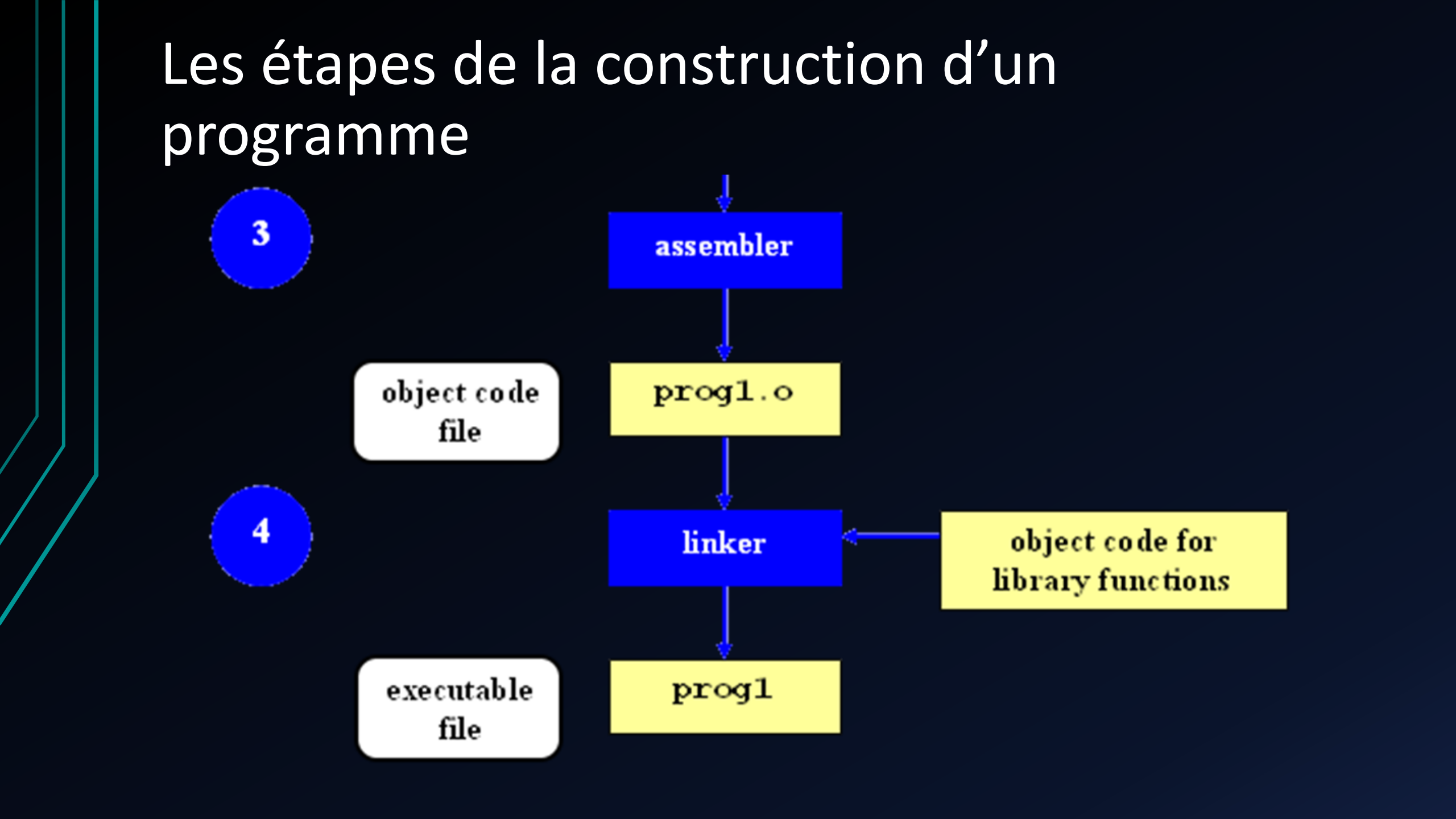
4

executable  
file

linker

object code for  
library functions

prog1



# Les types de données

- Caractères (char, string)
- Entiers (int)
- Nombres décimaux (float)
- Types définis par l'utilisateur.

# Les conversions entre types de données

- Les casts à la compilation :

`static_cast<TypeName>()`

- Les casts à l'exécution:

`dynamic_cast<TypeName>()`

- Les méthodes de la bibliothèque standard (STL)

# Vocabulaire

- La Déclaration permet de donner certaines informations au compilateur, par exemple:

<code>extern int x;</code>	<code>// object declaration</code>
<code>std::size_t numDigits(int number);</code>	<code>// function declaration</code>
<code>class Widget;</code>	<code>// class declaration</code>



# Vocabulaire

- La signature est révélée par la déclaration d'une fonction par exemple:

```
std::size_t nombreEleves(int  
number) ;
```

# Vocabulaire

- La définition fournit au compilateur les infos qui lui manquent dans la déclaration. Pour un objet ce sera l'espace mémoire à allouer, pour une fonction ou une fonction template, le corps de la fonction sera aussi fourni.

- Exemples:

```
int x; // Définition d'un objet
```

```
int Somme (int a, int b) // Définition d'une fonction
```

```
{
```

```
    return a + b;
```

```
}
```

# Vocabulaire

- L'Initialisation est le processus consistant à donner à l'objet sa première valeur. Pour les objets définis par l'utilisateur, le constructeur par défaut est appelé.

- Exemples:

```
int x; // Définition d'un objet
```

```
int Somme (int a, int b) // Définition d'une fonction
```

```
{
```

```
    return a + b;
```

```
}
```



# Programmer en C++

## QUELQUES BASES PRATIQUES

# Les pointeurs en C++

- Les pointeurs existent en C++, de par son « héritage » du C.
- Exemple de déclaration de pointeur en C++ :

```
#include <iostream>

int main()
{
    int i = 2;
    int* j = &i;
    std::cout << "i = " << i++ << std::endl;
    std::cout << "j = " << *j << std::endl;
}
```

# Les références en C++

- Les références, comme les pointeurs, sont des variables du type qu'elles permettent de manipuler. Il y a toutefois quelques règles à respecter:
  - Elles ne peuvent jamais être nulles ou pointer vers une valeur nulle.
  - Elles pointent vers le même objet tout au long de leur cycle de vie, on ne peut pas les réassigner.

Lvalues et Rvalues, que représentent-elles?

- En C++ tous les objets sont soit des lvalues soit des rvalues.

# Lvalues et Rvalues, que représentent-elles?

- LValue: un objet dont la ressource ne peut pas être réutilisée. Par exemple:
  - une variable nommée par exemple.
  - une variable dont le scope est plus grand que le bloc en cours
  - un objet qui a un emplacement dans la mémoire.
- Une référence vers une lvalue aura un seul esperluette &



# Lvalues et Rvalues, que représentent-elles?

- Rvalue: un objet dont la ressource peut être réutilisée sans impacter le code autour. Par exemple:
  - le résultat d'un calcul ( `int x = 3+2` ) .
  - Le constructeur par défaut d'un objet (`MaClasse()` `<=` peut être modifié sans conséquence sur le reste du programme)
  - un objet qui n'a pas encore d'emplacement dans la mémoire.
- Une référence vers une rvalue aura deux esperluettes `&&`

# Les tableaux en C++

- Les tableaux sont un héritage du C, ils fonctionnent de la même manière.

```
TYPE NOMDUTABLEAU[TAILLE]
```

```
int tableau[10];
```

- Comme tout héritage du C, on cherchera un maximum à les éviter, et les remplacer par des objets de la librairie standard C++ (STL)

# Afficher dans la console

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
}
```

# Saisir dans la console

```
1  #include <iostream>
2
3  int main()
4  {
5      std::string saisie;
6      std::cin >> saisie;
7
8      std::cout << "Vous avez saisi: " << saisie << std::endl;
9  }
10
```