

Chapter 2. Elementary Syntax in C++ Language

1-Introduction

This chapter explores the elementary syntax of the C++ language, an essential component for anyone wishing to develop programming skills with this language. We will cover fundamental aspects, such as comments, variables, different data types, and operators, which allow for data manipulation and the creation of structured programs. The goal is to give you a clear understanding of the basics of C++, allowing you to write your first programs and acquire good practices for programming in C++.

2- Elements of a C++ Program

2- 2-1- The comments:

Comments are essential and very easy to add. They make the program easier to read, and it is recommended to comment on every program. A comment is a text ignored by the compiler.

Note: An excess of comments can diminish their usefulness, as important information may get lost among superfluous details.

There are two types of comments in C++:

- The first is delimited by `/*...*/` and can span multiple lines.

Example: `input_sentence;`

- The second is indicated by `//` and is used for single-line comments.

Example:

- `/*` these comments can span multiple lines. `*/`
- `/*` indicates the beginning of the comment and `*/` indicates the end of the comment or the comment between these two symbols can span multiple lines.

Context: `*/`

- `//` Single-line comments

2-2-The variables

A variable has a name and a type. We know how to name our variables, let's now look at the different types they can have. The computer needs to know the content of its memory, so it is necessary to specify the type of value that the variable will contain. Is it a number, a word, a letter? This must be clearly indicated.

Here is the list of variable types that can be used in C++.

| Nom du type | Ce qu'il peut contenir |
|--------------|--|
| bool | Peut contenir deux valeurs "vrai" (true) ou "faux" (false). |
| char | Une lettre. |
| int | Un nombre entier. |
| unsigned int | Un nombre entier positif ou nul. |
| double | Un nombre à virgule. |
| string | Une chaîne de caractères. C'est-à-dire une suite de lettres, un mot, une phrase. |

Declare a variable

It's time to get down to business and ask the computer to reserve some memory space for us. Technically, this is called variable declaration. We need to tell the computer the type of variable we want, its name, and possibly its value. It's a very simple process.

We indicate things exactly in this order.

Context: TYPE NAME (VALUE); We can also use the same syntax as in the C language:

Context: TYPE NAME=VALUE; input_sentence: TYPE NAME=VALUE;

Example :

```
#include <iostream>
using namespace std;
int main()
{
    int age(16);cin.ignore();
    return 0;
}
```

What happens on line 5 of this program?

The computer sees that we would like to borrow a drawer in its memory with the properties

following:

It can contain integers.

He has a label indicating that his name is age. It contains the value 16.

The table below represents the types of variables with size and range:

| Type | Taille | Intervalle de valeurs |
|-----------------|-----------|--|
| bool | 1 octet | <i>false ou true</i> |
| char | 1 octet | −128 à 127 |
| unsigned char | 1 octet | 0 à 255 |
| short | 2 octets | −32768 à 32767 |
| unsigned short | 2 octets | 0 à 65535 |
| int | 4 octets | −2147483648 à 2147483647 |
| unsigned (int) | 4 octets | 0 à 4294967295 |
| long | 4 octets | −2147483648 à 2147483647 |
| unsigned (long) | 4 octets | 0 à 4294967295 |
| float | 4 octets | $+/- 3.4 * 10^{-38}$ à $3.4 * 10^{38}$ |
| double | 8 octets | $+/- 1.7 * 10^{-308}$ à $1.7 * 10^{308}$ |
| long double | 10 octets | $+/- 1.2 * 10^{-4932}$ à $1.2 * 10^{4932}$ |

Declare without initializing

Now that we have seen the general principle, let's examine the details more deeply. Context:
Now that we have seen the general principle, let's examine the details more closely. When
declaring a variable, your program actually performs two successive operations:

1. He asks the computer for a memory storage area.
2. It assigns the provided value to this area, which is called the initialization of the variable.

These two steps happen automatically, without any intervention from you.

Sometimes, we don't know the value to assign to a variable at the time of declaration. In this case, it is possible to simply allocate the memory without initializing the variable. For this, you simply need to specify the type and name of the variable, without indicating a value.

TYPE NOM ;

Example of declaration without initialization

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string nomJoueur;
    int nombreJoueurs;
    bool aGagne; //Le joueur a-t-il gagné ?
    cin.ignore() ;
    return 0;
}
```

The goal of this program is to declare a few variables to store information about a player.
Context: The goal of this program is to declare some variables to store information about a player. It declares:

1. string playerName: to store the player's name.
2. int numberOfPlayers: to store the number of players.
3. bool aGagne: to indicate whether the player has won or not.

Next, the line `cin.ignore();` is used to keep the console open. The program does nothing more; it simply declares the variables without using them. It's a basic structure that could be further developed for a more comprehensive program.

Display the value of a variable

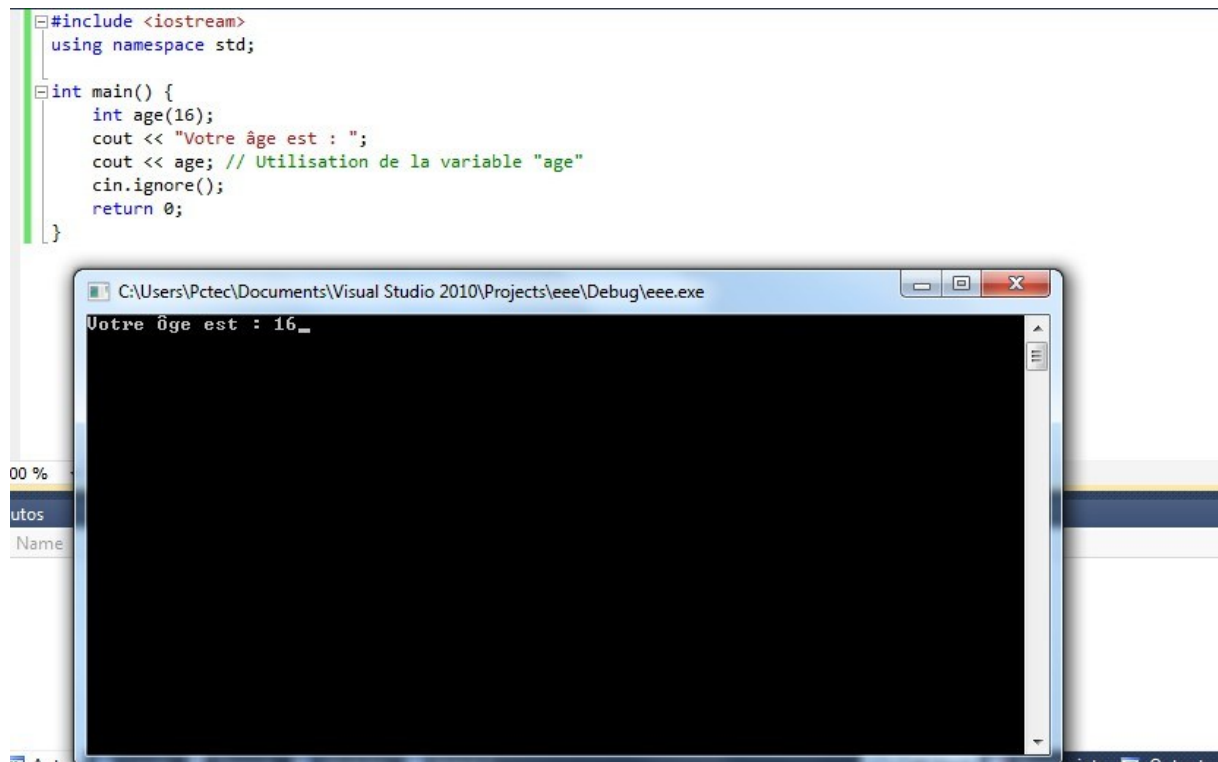
```
cout << âge;
```

Example of display

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

```
int main() {  
    int age(16);  
    cout << "Votre âge est : ";  
    cout << age; // Utilisation de la variable "age"  
    cin.ignore();  
    return 0;  
}
```

Once compiled, this code displays this on the screen:



2-2- The Different Types of Variables

- Integers

The C++ language distinguishes several types of integers.

| TYPE | DESCRIPTION | TAILLE MEMOIRE |
|----------------|------------------------|--|
| int | entier standard signé | 4 octets: $-2^{31} \leq n \leq 2^{31}-1$ |
| unsigned int | entier positif | 4 octets: $0 \leq n \leq 2^{32}$ |
| short | entier court signé | 2 octets: $-2^{15} \leq n \leq 2^{15}-1$ |
| unsigned short | entier court non signé | 2 octets: $0 \leq n \leq 2^{16}$ |
| char | caractère signé | 1 octet : $-2^7 \leq n \leq 2^7-1$ |
| unsigned char | caractère non signé | 1 octet : $0 \leq n \leq 2^8$ |

Some constant characters:

| CARACTERE | |
|-----------|------------------------|
| '\n' | interligne |
| '\t' | tabulation horizontale |
| '\v' | tabulation verticale |
| '\r' | retour chariot |
| '\f' | saut de page |
| '\\' | backslash |
| '\"' | cote |
| '\"' | guillemets |

- The reals

A real is composed:

- with a sign,
- of a mantissa,
- of an exponent,

A number of bits is reserved in memory for each element.

The C++ language distinguishes between 2 types of real numbers:

| TYPE | DESCRIPTION | TAILLE MEMOIRE |
|--------|-----------------------|----------------|
| float | réel standard | 4 octets |
| double | réel double précision | 8 octets |

Basic types and constants

In C++, the basic types are:

Context: – bool : boolean 2, can be true or false,

– char: character (generally 8 bits), which can also be explicitly declared signed (signed char) or unsigned (unsigned char),

– int: integer (16 or 32 bits, depending on the machines), which has the variants short [int] and long [int], all three of which can also be declared unsigned,

– float: real (1 machine word),

– double: real in double precision (2 machine words), and its variant long double (3 or 4 machine words),

2-3-Assignment and expression instructions

Simple assignment

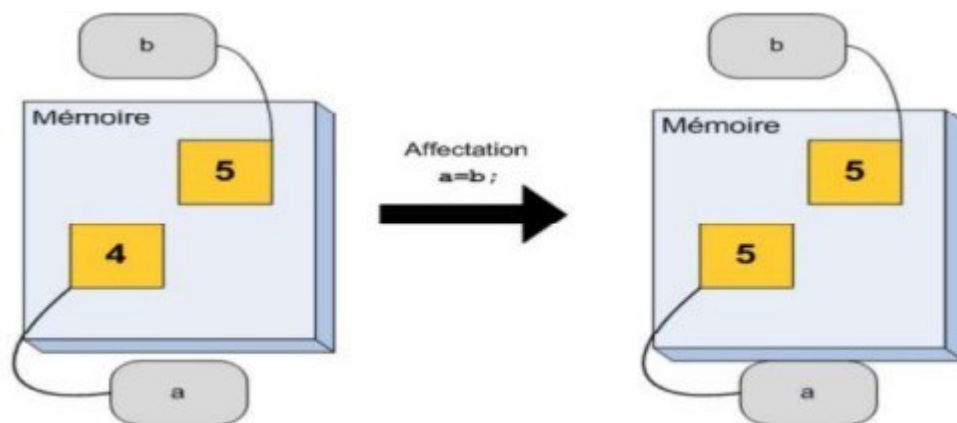
= affectation

It should be noted that the = sign is the assignment operator, not the comparison operator; this sometimes leads to confusion and results in errors that are difficult to discern. Also note that assignment is an expression like any other, meaning it returns a value. It is therefore possible to write: $a = b = c + 2$; this is equivalent to assigning to b the result of the evaluation of $c + 2$, then to a the result of the assignment $b = c + 2$, that is, the value we gave to b. Notice the order of evaluation from right to left.


Example Test of assigning one variable to another

```
#include <iostream>
using namespace std;
int main()
{
    int a(4), b(5); //Déclaration de deux variables.
    cout << "a vaut : " << a << " et b vaut : " << b << endl;
    cout << "Affectation !" << endl;
    a = b; //Affectation de la valeur de 'b' à 'a'.
    cout << "a vaut : " << a << " et b vaut : " << b << endl;

    system("PAUSE");
    return 0;
}
```



Code: Execution



The image shows a screenshot of a terminal window with a black background and white text. The text displays the output of a C++ program. It shows two lines of variable values, a prompt for user input, and the result of a swap operation. The window has a standard macOS-style title bar and a vertical scrollbar on the right side.

```
a vaut : 4 et b vaut : 5  
Affectation ?  
a vaut : 5 et b vaut : 5
```

The compound assignment that consists of associating an operator with the basic assignment sign '=':

General form:

General form: Variable operator = expression The expression is equivalent to: input_sentence:

Variable operator = expression

The expression is equivalent to:

Context: The expression is equivalent to: Variable=variable operator expression input_sentence:

Variable=variable operator expression

| Opérateur | Exemple | |
|-----------|------------|---------------|
| | Expression | Equivalent |
| += | a += 8 | a = a + 8 |
| -= | a -= 8 | a = a - 8 |
| *= | a *= b | a = a * b |
| /= | a /= b+c | a = a /(b+c) |
| %= | a %= b*c | a = a % (b*c) |

Example :

```
int i=2,j=34;
i += 2;          //i=i+2=4
j *= 10;         //j=j*10=340
j *= (i+1);      //j=j*(i+1)=340*5=1700
i+= j= 3;        //j=3, i=i+j=4+3=7
```

Increment and Decrement Operators ++ Increment __ Decrement

These operators, which can only be applied to scalar types, can be used in two ways: in principle, if they prefix a variable, it will be incremented (or decremented) before use in the rest of the expression; if they postfix it, it will only be modified after use.

Example 1:

Context: `A++ ; // A is equal to A+1`

`A++ ; A=A+1 ;` or `A+=1` are equivalent. `A-- ; A=A-1 ;` or `A-=1` are equivalent.

Example 2:

Table 2-1

| Opérateur | Exemple | Description |
|------------------------|--------------------|------------------------------|
| ++ (incrémentation) | <code>M=++a</code> | <code>a = a + 1 , M=a</code> |
| | <code>M=a++</code> | <code>M=a, a = a + 1</code> |
| -- (Décrémentation) | <code>M=--a</code> | <code>a = a - 1 , M=a</code> |
| | <code>M=a--</code> | <code>M=a, a = a - 1</code> |

2-4- THE OPERATORS IN C++

In C++ programming, an operator is a symbol that instructs the compiler to perform an operation.

Context: In C++ programming, an operator is a symbol that instructs the compiler to perform an operation. There are several types of operations:

Operation operator:

Mathematical operators:

There are five operations addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) A small summary table The table below summarizes the mathematical operators.

| Opération | Symbole | Exemple |
|----------------|---------|-------------------|
| Addition | + | résultat = a + b; |
| Soustraction | - | résultat = a - b; |
| Multiplication | * | résultat = a * b; |
| La division | / | résultat = a / b; |
| modulo | % | résultat = a %b; |

Exemple1 :

19.0 / 5.0 vaut 3.8,
19 / 5 vaut 3,
19 % 5 vaut 4.

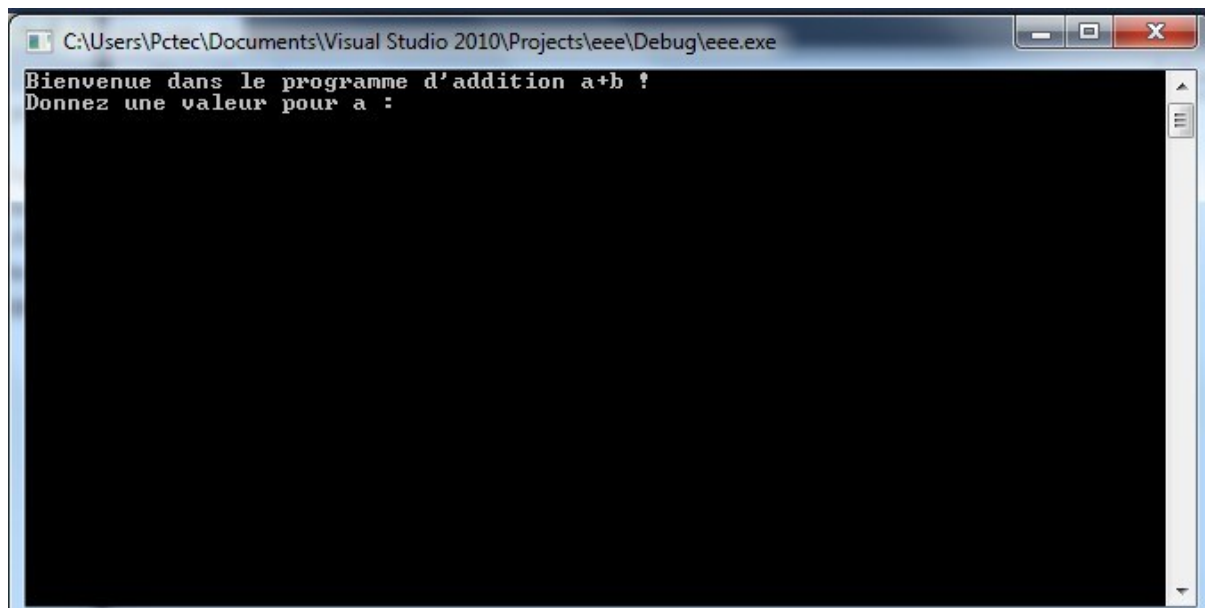
Exemple 2 : The addition of two numbers:

```
#include <iostream>
using namespace std;
int main()
{
    double a(0), b(0); //Déclaration des variables utiles
    cout << "Bienvenue dans le programme d'addition a+b !" << endl;
    cout << "Donnez une valeur pour a : "; //Demande du premier nombre

    cin >> a;
    cout << "Donnez une valeur pour b : "; //Demande du deuxième nombre

    cin >> b;
    double const resultat(a + b); //On effectue l'opération
    cout << a << " + " << b << " = " << resultat << endl; //On affiche le
    résultat
    system("PAUSE");
    return 0;
}
```

After execution



You just need to give a value for the variable a, and b, and you will get the sum of a+b.

Mathematical functions: they are declared in `<math.h>`. Context: Mathematical functions: they are declared in `<math.h>`. There are notably the following functions, with a double parameter and a double result:

- floor: (resp.ceil): integer part by default
- fabs: absolute value
- sqrt : square root
- pow: power (pow(x,y) returns x^y)
- exp, log, log10
- Trigonometric functions: sin, cos, tan, asin, acos, atan,
- Hyperbolic functions: sinh, cosh, tanh: Return respectively the hyperbolic sine, the hyperbolic cosine, and the hyperbolic tangent.

Example

Here is a similar program that uses the `cmath` library to perform a mathematical operation (e.g., calculate the power) instead of a simple addition:

```
#include <iostream>
#include <cmath> // Inclure cmath pour les fonctions mathématiques
using namespace std;

int main()
{
    double base(0), exposant(0); // Déclaration des variables utiles
    cout << "Bienvenue dans le programme de calcul de puissance base^exposant !" <<
endl;

    // Demande de la valeur de la base
    cout << "Donnez une valeur pour la base : ";
    cin >> base;

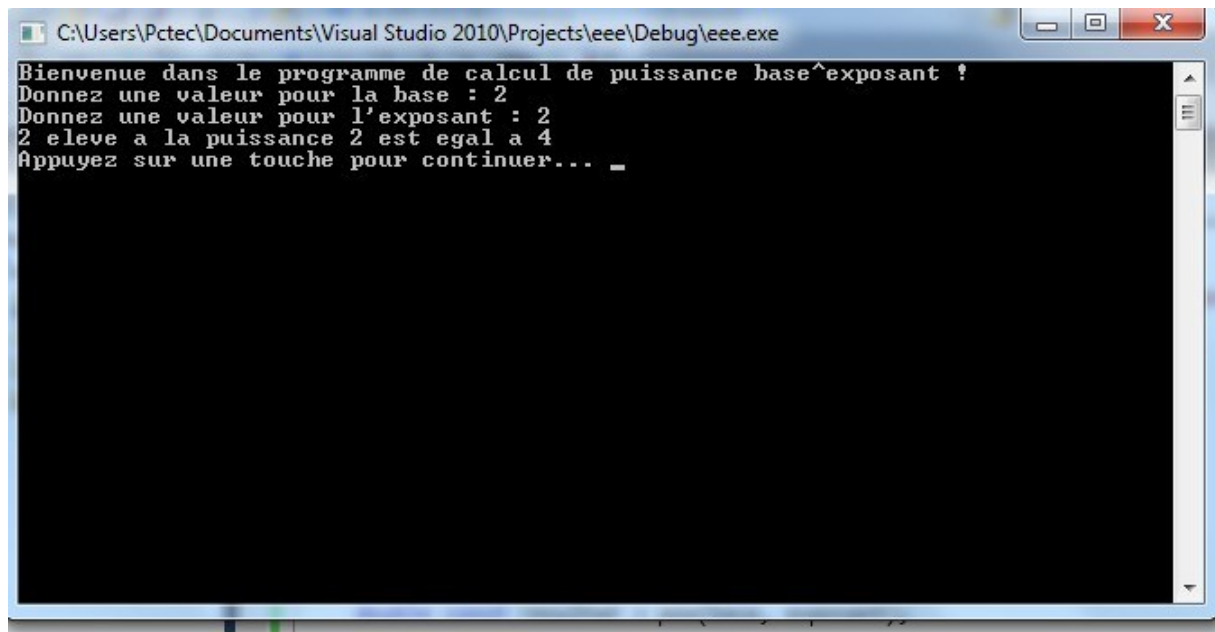
    // Demande de la valeur de l'exposant
    cout << "Donnez une valeur pour l'exposant : ";
    cin >> exposant;

    // Calcul de la puissance
    double const resultat = pow(base, exposant);

    // Affichage du résultat
    cout << base << " élevé à la puissance " << exposant << " est égal à " << resultat
<< endl;

    // Pause avant la fermeture
    system("pause");
    return 0;
}
```

If we choose the value 2 for the exponent and for the base, we will have



```
C:\Users\Pctec\Documents\Visual Studio 2010\Projects\eee\Debug\eee.exe
Bienvenue dans le programme de calcul de puissance base^exposant !
Donnez une valeur pour la base : 2
Donnez une valeur pour l'exposant : 2
2 eleve a la puissance 2 est egal a 4
Appuyez sur une touche pour continuer... _
```

Relational operators

There are six (06) of them:

1. Equal to ==,
2. Different from (!=)
3. Greater than (>)
4. Greater than or equal to (>=)
5. Less than (<)
6. Less than or equal to (<=)

| Symbole | Signification |
|---------|-------------------------|
| == | Est égal à |
| > | Est supérieur à |
| < | Est inférieur à |
| >= | Est supérieur ou égal à |
| <= | Est inférieur ou égal à |
| != | Est différent de |

Exemple :

`Int a=27 ; b=19 ;`

`a==b ; // this operation gives the value`

`a<b ; // this operation gives the value True`

The input/output operators

The input operation is symbolized by `>>`: reading the value of from the keyboard, and the output operation is symbolized by: `<<`, displaying the value of on the screen.

Example:

`int a; cin >> //to enter a value for A A++; cout << A; //display the result.`

Here is a table presenting all the C++ operators (some will only be used in later chapters):

Here is a table presenting all the operators in C++ (some will be covered in later chapters):

| Catégorie | Opérateurs |
|-------------------------|---|
| Résolution de portée | :: (portée globale - unaire) :: (portée de classe - binaire) |
| Référence | () [] -> . |
| Unaire | + - ++ -- ! ~ * & sizeof cast dynamic_cast static_cast reinterpret_cast const_cast new new[] delete delete[] |
| Sélection | ->* .* |
| Arithmétique | * / % |
| Arithmétique | + - |
| Décalage | << >> |
| Relationnels | < <= > >= |
| Relationnels | == != |
| Manipulation de bits | & |
| Manipulation de bits | ^ |
| Manipulation de bits | |
| Logique | && |
| Logique | |
| Conditionnel (ternaire) | ? : |
| Affectation | = += -= *= /= %= &= ^= = <<= >>= |
| Séquentiel | , |

Exercise

Context: Write a C++ program that does the following:

1. Declare an integer variable named age.
2. Initialize age with the value 30.
3. Use a single-line comment to explain the initialization.
4. Display the value of age on the console.

Solution

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

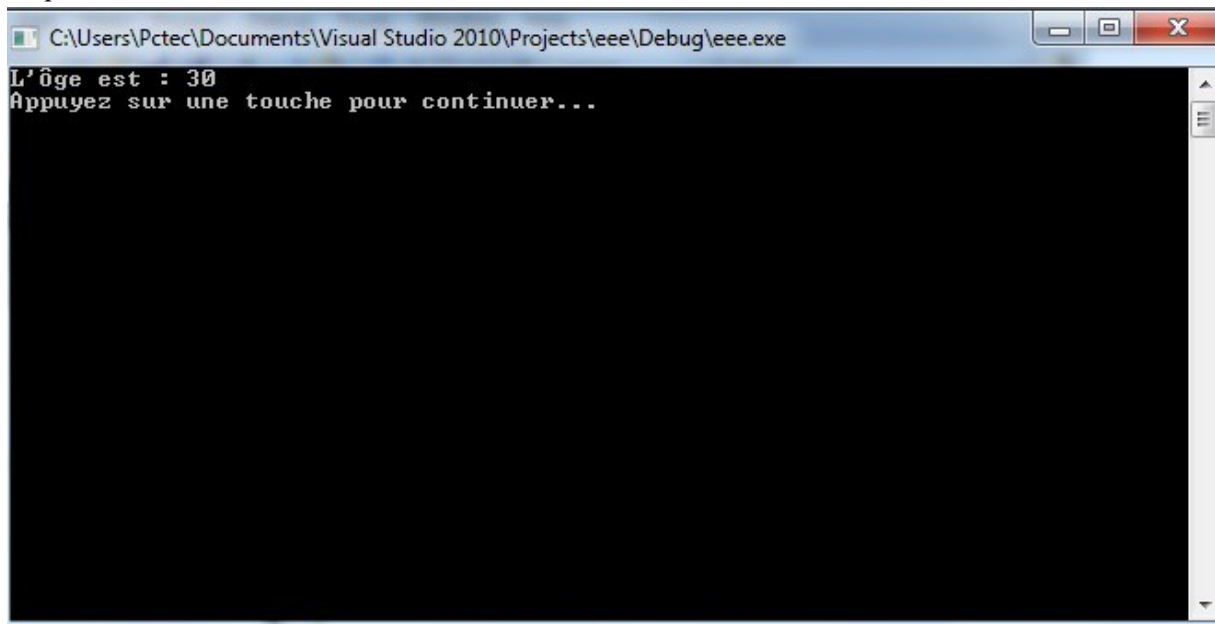
int main() {
    int age = 30; // Initialisation de la variable age avec la valeur 30

    // Affichage de la valeur de age
    cout << "L'âge est : " << age << endl;
    system("pause");
    return 0;
}
```

Context: Explanations:

1. Declaration and initialization of the variable: `int age = 30;` declares an integer variable `age` and initializes it with the value 30.
2. Comment: `// Initializing the age variable with the value 30` is a single-line comment that explains the initialization.
3. Displaying the value:
Context: 3. Displaying the value: `cout << "The age is: " << age << endl;` displays the message "The age is: 30" on the console.

Expected result after execution



: