Lesson 4 - C++ Variable Types

CSC/ITS – 101(PROGRAMMING 1)

What is C++ variable?

A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C++ is case-sensitive –

What is C++ variable?

There are following basic types of variable in C++ as explained in last chapter –

Type	Description
bool	Stores either value true or false.
char	Typically a single octet (one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.
void	Represents the absence of type.
wchar_t	A wide character type.

Variable Definition in C++

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type, and contains a list of one or more variables of that type as follows –

```
type variable_list;
```

Here, type must be a valid C++ data type including char, w_char, int, float, double, bool or any user-defined object, etc., and variable_list may consist of one or more identifier names separated by commas. Some valid declarations are shown here –

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

Variable Definition in C++

Some examples are –

```
extern int d = 3, f = 5;    // declaration of d and f.
int d = 3, f = 5; // definition and initializing d and f.
short int z = 22;     // definition and initializes z.
char x = 'x';      // the variable x has the value 'x'.
```

Variable Declaration in C++

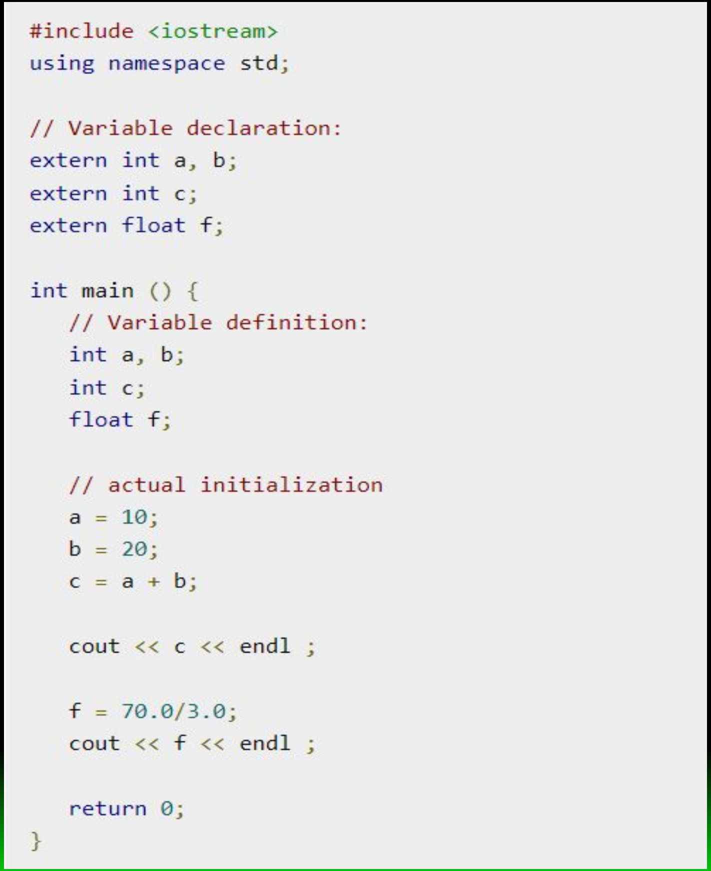
A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation without needing complete detail about the variable. A variable declaration has its meaning at the time of compilation only, compiler needs actual variable definition at the time of linking of the program.

A variable declaration is useful when you are using multiple files and you define your variable in one of the files which will be available at the time of linking of the program. You will use extern keyword to declare a variable at any place. Though you can declare a variable multiple times in your C++ program, but it can be defined only once in a file, a function or a block of code.

Variable Declaration in C++

Example

Try the following example where a variable has been declared at the top, but it has been defined inside the main function –



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

// Variable declaration:
extern int a, b;
extern int c;
extern float f;

int main () {
   // Variable definition:
   int a, b;
   int c;
   float f;

   // actual initialization
   a = 10;
   b = 20;
   c = a + b;

   cout << c << endl ;

   f = 70.0/3.0;
   cout << f << endl ;

   return 0;
}
```

Variable Declaration in C++

When the above code is compiled and executed, it produces the following result –

```
30
23.3333
```

Variable Declaration in C++

Same concept applies on function declaration where you provide a function name at the time of its declaration and its actual definition can be given anywhere else. For example –

```
// function declaration
int func();
int main() {
   // function call
   int i = func();
}
// function definition
int func() {
   return 0;
}
```

Lvalues and Rvalues

There are two kinds of expressions in C++ –

lvalue – Expressions that refer to a memory location is called "lvalue" expression. An lvalue may appear as either the left-hand or right-hand side of an assignment.

rvalue – The term rvalue refers to a data value that is stored at some address in memory. An rvalue is an expression that cannot have a value assigned to it which means an rvalue may appear on the right- but not left-hand side of an assignment.

Lvalues and Rvalues

Variables are lvalues and so may appear on the left-hand side of an assignment. Numeric literals are rvalues and so may not be assigned and can not appear on the left-hand side. Following is a valid statement –

```
int g = 20;
```

But the following is not a valid statement and would generate compile-time error –

```
10 = 20;
```

Lesson 5 - Variable Scope in C++

CSC/ITS – 101(PROGRAMMING 1)

What is a scope?

A scope is a region of the program and broadly speaking there are three places, where variables can be declared –

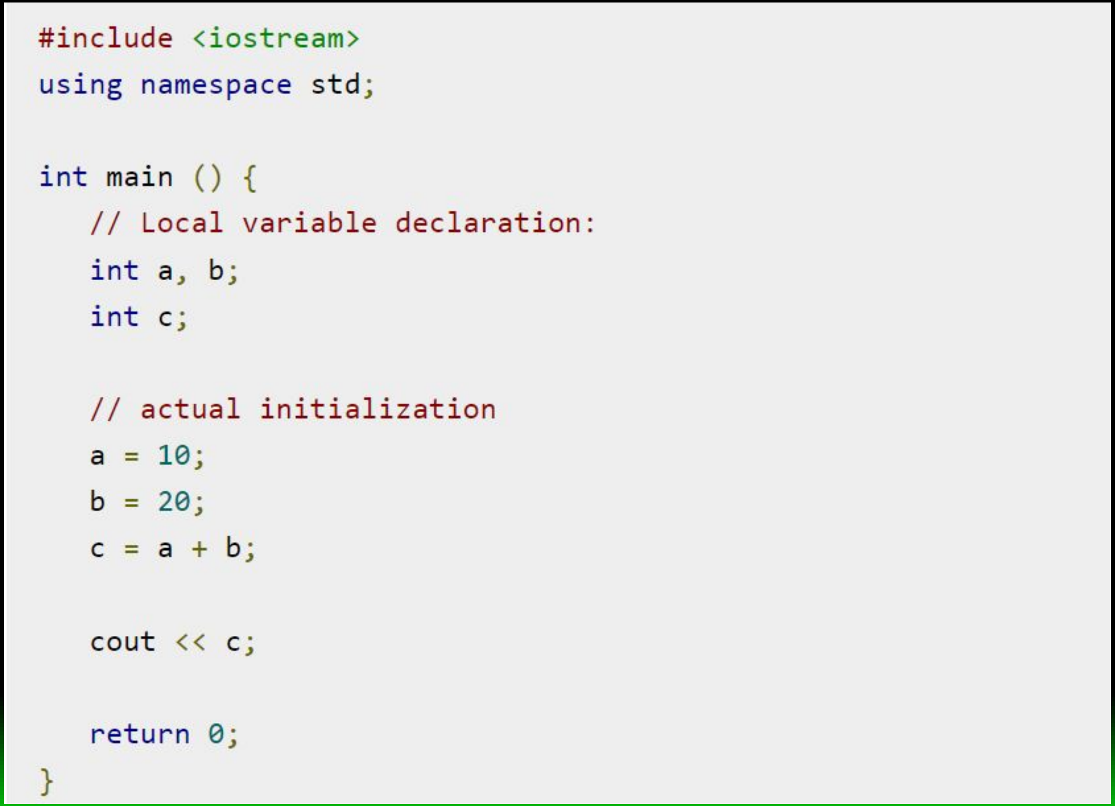
- Inside a function or a block which is called local variables,
- In the definition of function parameters which is called formal parameters.
- Outside of all functions which is called global variables.

We will learn what is a function and it's parameter in subsequent chapters. Here let us explain what are local and global variables.

Local Variables

Variables that are declared inside a function or block are local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. Following is the example using local variables –

Local Variables



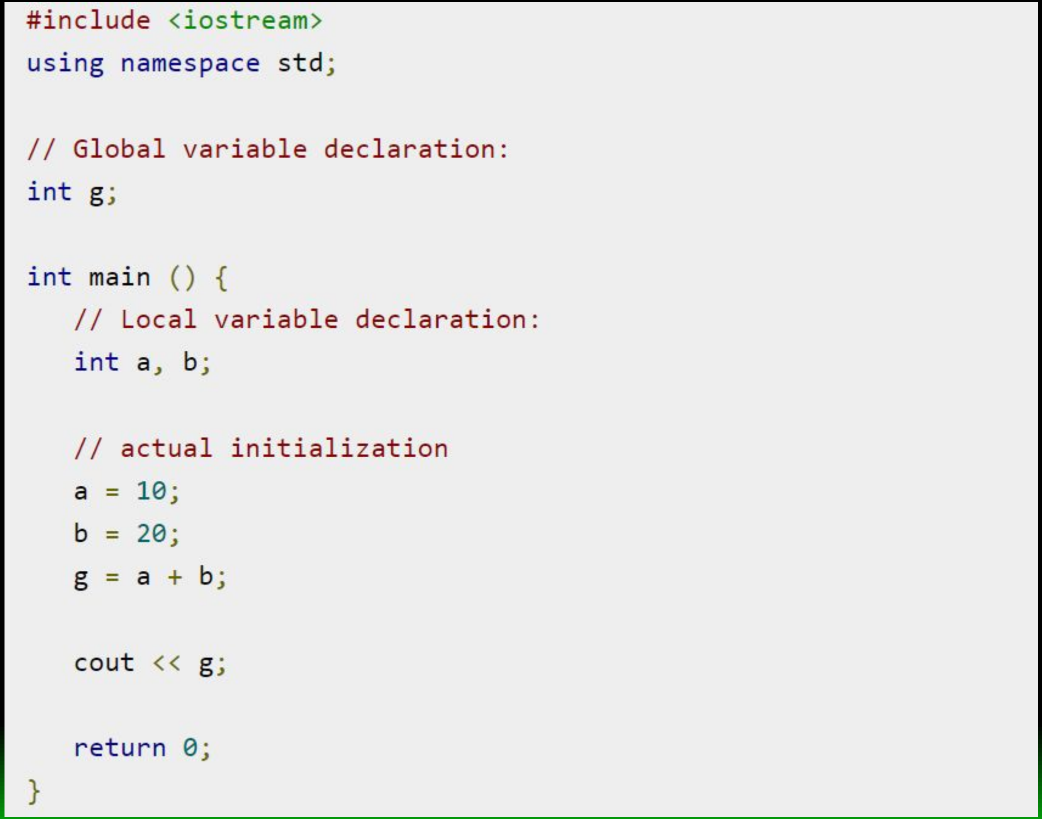
```
#include <iostream>
using namespace std;

int main () {
   // Local variable declaration:
   int a, b;
   int c;

   // actual initialization
   a = 10;
   b = 20;
   c = a + b;

   cout << c;

   return 0;
}
```

Global Variables

Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the life-time of your program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. Following is the example using global and local variables –

Global Variables

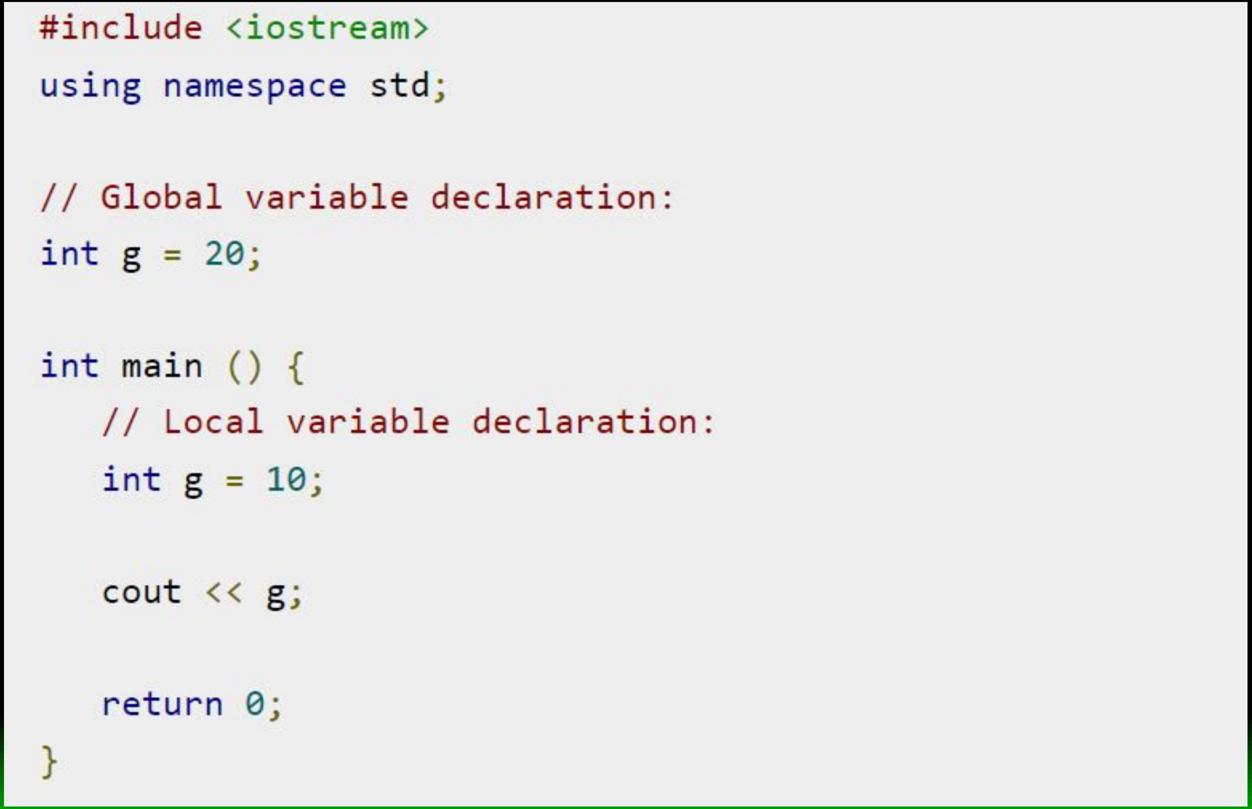


```
#include <iostream>
using namespace std;

// Global variable declaration:
int g;

int main () {
   // Local variable declaration:
   int a, b;

   // actual initialization
   a = 10;
   b = 20;
   g = a + b;

   cout << g;

   return 0;
}
```

Global and Local Variables

A program can have same name for local and global variables but value of local variable inside a function will take preference. For example –

Global and Local Variables



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

// Global variable declaration:
int g = 20;

int main () {
   // Local variable declaration:
   int g = 10;

   cout << g;

   return 0;
}
```

Lesson 6 - C++ Constants/Literals

CSC/ITS – 101(PROGRAMMING 1)

Constants/Literals

Constants refer to fixed values that the program may not alter and they are called literals.

Constants can be of any of the basic data types and can be divided into Integer Numerals, Floating-Point Numerals, Characters, Strings and Boolean Values.

Again, constants are treated just like regular variables except that their values cannot be modified after their definition.

Integer Literals

An integer literal can be a decimal, octal, or hexadecimal constant. A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal.

An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.

Here are some examples of integer literals –

Integer Literals

```
212       // Legal
215u      // Legal
0xFeeL    // Legal
078       // Illegal: 8 is not an octal digit
032UU     // Illegal: cannot repeat a suffix
```

Integer Literals

Following are other examples of various types of Integer literals –

```
85        // decimal
0213      // octal
0x4b      // hexadecimal
30        // int
30u       // unsigned int
30l       // long
30ul      // unsigned long
```

Floating-point Literals

A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form.

While representing using decimal form, you must include the decimal point, the exponent, or both and while representing using exponential form, you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

Floating-point Literals

Here are some examples of floating-point literals –

```
3.14159      // Legal
314159E-5L   // Legal
510E         // Illegal: incomplete exponent
210f         // Illegal: no decimal or exponent
.e55         // Illegal: missing integer or fraction
```

Boolean Literals

There are two Boolean literals and they are part of standard C++ keywords –

A value of true representing true.

A value of false representing false.

You should not consider the value of true equal to 1 and value of false equal to 0.

Character Literals

Character literals are enclosed in single quotes. If the literal begins with L (uppercase only), it is a wide character literal (e.g., L'x') and should be stored in wchar_t type of variable . Otherwise, it is a narrow character literal (e.g., 'x') and can be stored in a simple variable of char type.

Character Literals

A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t'), or a universal character (e.g., '\u02C0').

There are certain characters in C++ when they are preceded by a backslash they will have special meaning and they are used to represent like newline (\n) or tab (\t).

Character Literals

A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t'), or a universal character (e.g., '\u02C0').

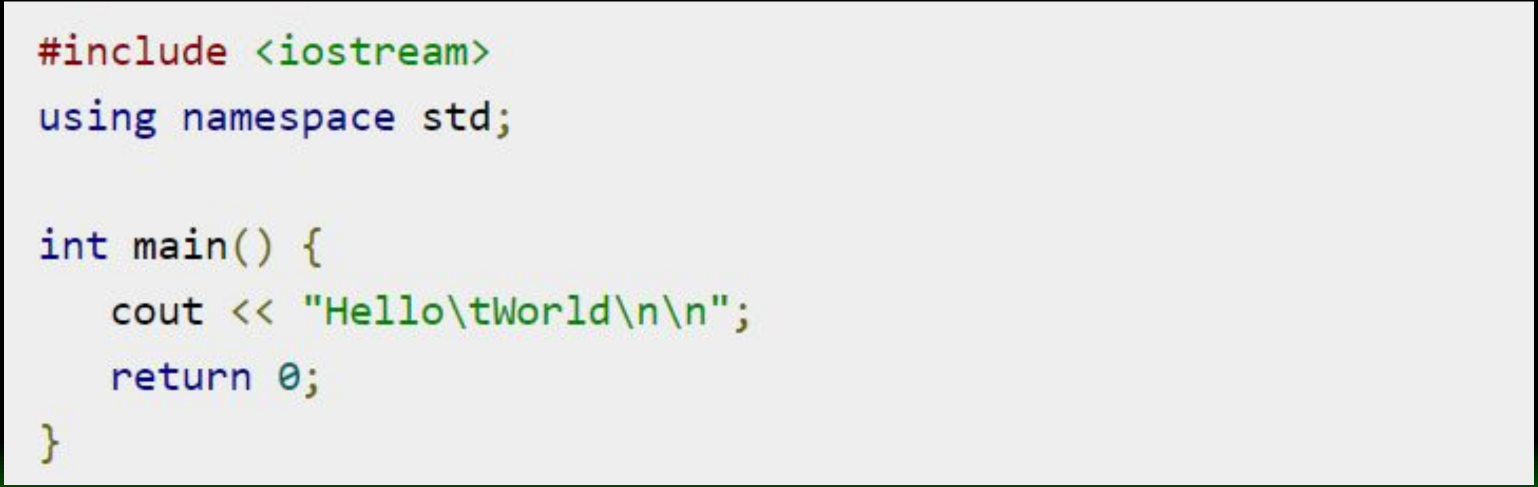
There are certain characters in C++ when they are preceded by a backslash they will have special meaning and they are used to represent like newline (\n) or tab (\t).

Escape Sequences

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\ooo	Octal number of one to three digits
\xhh . . .	Hexadecimal number of one or more digits

Escape Sequences

Following is the example to show a few escape sequence characters –



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

int main() {
   cout << "Hello\tWorld\n\n";
   return 0;
}
```

String Literals

String literals are enclosed in double quotes. A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.

You can break a long line into multiple lines using string literals and separate them using whitespaces.

String Literals

Here are some examples of string literals. All the three forms are identical strings.

"hello, dear"

"hello, \n

dear"

"hello, " "d" "ear"

Defining Constants

There are two simple ways in C++ to define constants –

Using #define preprocessor.

Using const keyword.

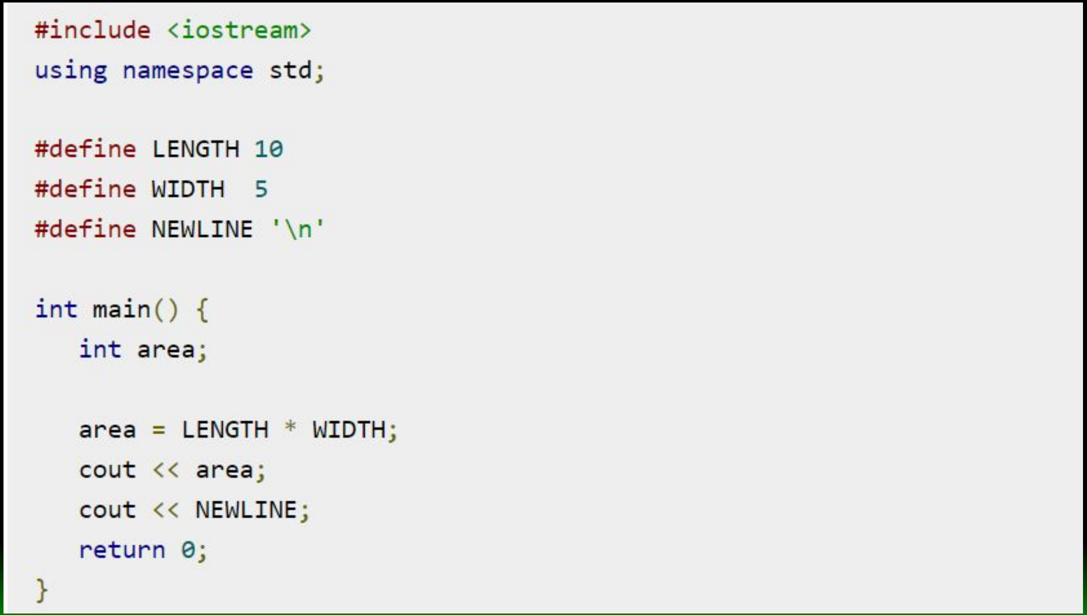
The #define Preprocessor

Following is the form to use #define preprocessor to define a constant –

```
#define identifier value
```

The #define Preprocessor

Following example explains it in detail –



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

#define LENGTH 10
#define WIDTH  5
#define NEWLINE '\n'

int main() {
   int area;

   area = LENGTH * WIDTH;
   cout << area;
   cout << NEWLINE;
   return 0;
}
```

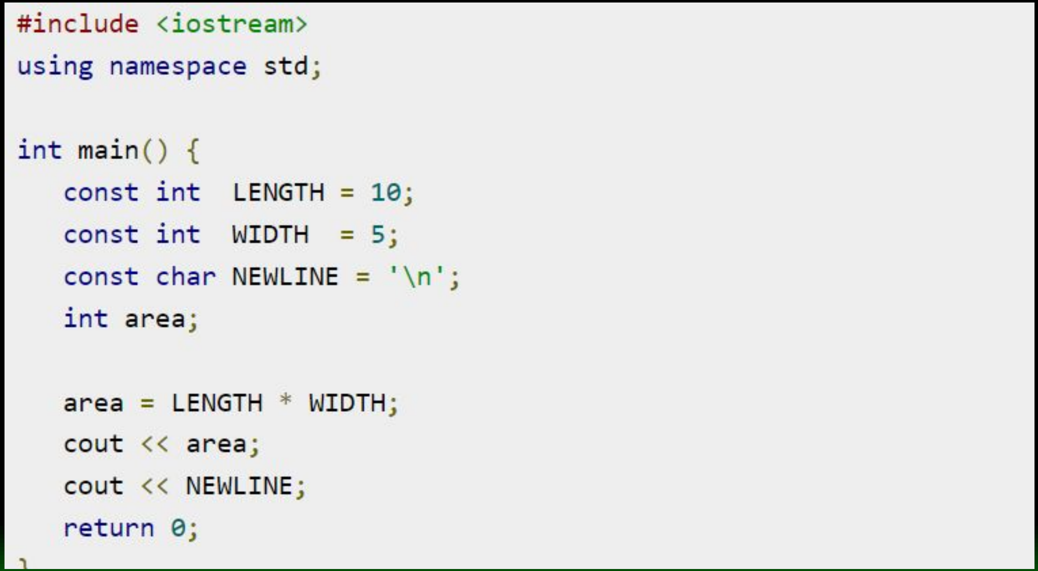
The const Keyword

You can use const prefix to declare constants with a specific type as follows –

```
const type variable = value;
```

The const Keyword

Following example explains it in detail –



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

int main() {
   const int  LENGTH = 10;
   const int  WIDTH  = 5;
   const char NEWLINE = '\n';
   int area;

   area = LENGTH * WIDTH;
   cout << area;
   cout << NEWLINE;
   return 0;
}
```

QUESTIONS?