Introduction, Basics, Conditions, Loops

Moien Makkiyan

Basic Programming
Session 1

November 18, 2024

Overview

Introduction

What is C++?

Definition

C++ is a cross-platform language that can be used to create high-performance applications.
C++ gives programmers a high level of control over system resources and memory.
Some examples that were developed using C++:

- Microsoft Windows
- Google Chrome
- Mozilla Firefox
- Adobe Photoshop
- Grand Theft Auto V (GTA V)

C++ Install IDE

To start using C++, you need two things:

- A text editor, like Notepad, to write C++ code
- A compiler, like GCC, to translate the C++ code into a language that the computer will understand.

Popular IDE's include Code::Blocks, Eclipse, and Visual Studio. These are all free, and they can be used to both edit and debug C++ code.

Your First Program:

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

int main() {
    cout << "Hello World!";
    return 0;
}
```

Don't worry if you don't understand the code above - we will discuss it in detail in later.

C++ Syntax

Code and Explanation

`#include <iostream>` is a header file library that lets us work with input and output objects, such as cout.
`using namespace std` means that we can use names for objects and variables from the standard library.
`int main()`, This is called a function. Any code inside its curly brackets {} will be executed.
`return 0` ends the main function.

Code and Explanation

Don't worry if you don't understand how `#include <iostream>` , `using namespace std` and `return 0` works. Just think of it as something that always appears in your program.

Note

C++ is case-sensitive: `cout` and `Cout` has different meaning.

Note

Note: Every C++ statement ends with a semicolon `;`.

Note

Do not forget to add the closing curly bracket } to actually end the main function.

Omitting Namespace

You might see some C++ programs that runs without the standard namespace library. The using namespace std line can be omitted and replaced with the std keyword, followed by the :: operator for some objects:

```
#include <iostream>

int main() {
  std::cout << "Hello World!";
  return 0;
}
```

C++ Output

Print Text

The cout object, together with the ¡¡ operator, is used to output values and print text. Just remember to surround the text with double quotes (""):

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

int main() {
  cout << "Hello World!";
  return 0;
}
```

You can add as many cout objects as you want. However, note that it does not insert a new line at the end of the output.

Print Numbers

You can also use cout() to print numbers. However, unlike text, we don't put numbers inside double quotes:

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

int main() {
  cout << 3+5;
  return 0;
}
```

New Lines

To insert a new line in your output, you can use the \n character:

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

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

New Lines

Another way to insert a new line, is with the endl manipulator:

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

int main() {
  cout<<"Hello World"<<endl;
  return 0;
}
```

Both \n and endl are used to break lines. However, \n is most used.

New Lines

But what is \n exactly? The newline character (\n) is called an escape sequence, and it forces the cursor to change its position to the beginning of the next line on the screen. This results in a new line.

Examples of other valid escape sequences

\t $\Longrightarrow$ Creates a horizontal tab
\\ $\Longrightarrow$ Inserts a backslash character (\)
\" $\Longrightarrow$ Inserts a double quote character

C++ Comments

Single-line Comments

Single-line comments start with two forward slashes (//).

Multi-line Comments

Multi-line comments start with /* and ends with */.

C++ Variables

Basic Data Types

Data Type	Size	Description
`boolean`	1 byte	Stores true or false values
`char`	1 byte	Stores a single character/letter/number, or ASCII values
`int`	2 or 4 bytes	Stores whole numbers, without decimals
`float`	4 bytes	Sufficient for storing 6–7 decimal digits
`double`	8 bytes	Sufficient for storing 15 decimal digits

Table: Basic Data Types

Variables

Syntax:

```
type variableName = value;
```

Note

If you assign a new value to an existing variable, it will overwrite the previous value.

```
int myNum = 15;  // myNum is 15
myNum = 10;  // Now myNum is 10
```

Note

When you do not want others (or yourself) to change existing variable values, use the const keyword.

```
const int myNum = 15;  // myNum will always be 15
```

float vs. double

The precision of a floating point value indicates how many digits the value can have after the decimal point. The precision of float is only six or seven decimal digits, while double variables have a precision of about 15 digits. Therefore it is safer to use double for most calculations.

Scientific Numbers

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1; // Output: 35000
cout << d1; // Output: 120000
```

C++ User Input

User Input

You have already learned that cout is used to output (print) values. Now we will use cin to get user input.
cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).
In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

```
int x;
cout << "Type a number: "; // Type a number and press enter
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

Good To Know

cout is pronounced "see-out". Used for output, and uses the insertion operator (<<)

cin is pronounced "see-in". Used for input, and uses the extraction operator (>>)

C++ Operators

Operators Table

Operator	Name	Description	Example
+	Addition	Adds together two values	`x + y`
-	Subtraction	Subtracts one value from another	`x - y`
*	Multiplication	Multiplies two values	`x * y`
/	Division	Divides one value by another	`x / y`
%	Modulus	Returns the division remainder	`x % y`
++	Increment	Increases the value of a variable by 1	`++x`
--	Decrement	Decreases the value of a variable by 1	`--x`

Table: Basic Operators

Assignment Operators Table

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^3

Table: Assignment Operators

Comparison Operators Table

Operator	Name	Example
==	Equal to	`x == y`
!=	Not equal	`x != y`
>	Greater than	`x > y`
<	Less than	`x < y`
>=	Greater than or equal to	`x >= y`
<=	Less than or equal to	`x <= y`

Table: Comparison Operators

Logical Operators Table

Operator	Name	Description	Example
`&&`	Logical and	True if both statements are true	`x < 5 && x < 10`
`||`	Logical or	True if one of the statements is true	`x < 5 || x < 4`
`!`	Logical not	Reverses the result	`!(x < 5 && x < 10)`

Table: Logical Operators

C++ Math

Other functions, such as sqrt (square root), round (rounds a number) and log (natural logarithm), can be found in the <cmath> header file:

```
// Include the cmath library
#include <cmath>

cout << sqrt(64);
cout << round(2.6);
cout << log(2);
```

A list of all math functions can be found in this page.

C++ Conditions

Conditions and If Statements

C++ has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

Syntax

`if` syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

`else` syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

Syntax

`else if` syntax (Use the else if statement to specify a new condition if the first condition is false):

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is
  // false and condition2 is true
} else {
  // block of code to be executed if the condition1 is
  // false and condition2 is false
}
```

Short Hand If Else

It can be used to replace multiple lines of code with a single line, and is often used to replace simple if else statements:

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Example:

```
int time = 20;
if (time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
```

```
int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
cout << result;
```

Switch Statements

Use the switch statement to select one of many code blocks to be executed.

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Example:

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Switch Statements

The switch expression is evaluated once. The value of the expression is compared with the values of each case. If there is a match, the associated block of code is executed

```
int day = 4;
switch (day) {
  case 1:
    cout << "Monday"; break;
  case 2:
    cout << "Tuesday"; break;
  case 3:
    cout << "Wednesday"; break;
  case 4:
    cout << "Thursday"; break;
  case 5:
    cout << "Friday"; break;
}
// Outputs "Thursday" (day 4)
```

Break and Default Keyword

break

When C++ reaches a break keyword, it breaks out of the switch block. This will stop the execution of more code and case testing inside the block. When a match is found, and the job is done, it's time for a break. There is no need for more testing. A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

default

The default keyword specifies some code to run if there is no case match.

Break and Default Keyword

```
int day = 4;
 switch (day) {
   case 6:
     cout << "Today is Saturday";
     break;
   case 7:
     cout << "Today is Sunday";
     break;
   default:
     cout << "Looking forward to the Weekend";
 }
 // Outputs "Looking forward to the Weekend"
```

C++ Loops

While Loop

The while loop loops through a block of code as long as a specified condition is true.
Syntax:

```
while (condition) {
   // code block to be executed
 }
```

While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true. Syntax:

```
do {
   // code block to be executed
 }
 while (condition);
```

While Loop

Example:

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}
while (i < 5);
```

For Loop

Syntax:

```
for (statement 1; statement 2; statement 3) {
   // code block to be executed
 }
```

Statement 1 is executed (one time) before the execution of the code block.
Statement 2 defines the condition for executing the code block.
Statement 3 is executed (every time) after the code block has been executed.

For Loop

Example:

```
for (int i = 0; i < 5; i++) {
   cout << i << "\n";
 }
```

Statement 1 sets a variable before the loop starts (int i = 0).
Statement 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.
Statement 3 increases a value (i++) each time the code block in the loop has been executed.

C++ Break and Continue

Break

The break statement can also be used to jump out of a loop.

Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

C++ Break and Continue

Break Example:

```
int i = 0;
 while (i < 10) {
   cout << i << "\n";
   i++;
   if (i == 4) {
     break;
   }
 }
```

C++ Break and Continue

Continue Example:

```
int i = 0;
 while (i < 10) {
   if (i == 4) {
     i++;
     continue;
   }
   cout << i << "\n";
   i++;
 }
```

Questions?

Any Questions?

Feel free to reach out!

You can access the codes for this session here!

Email: moien.makkiyan@gmail.com

Telegram: @moein_makkiyan