4.10 — Introduction to if statements

ALEX OCTOBER 18, 2021

Consider a case where you're going to go to the market, and your roommate tells you, "if they have strawberries on sale, buy some". This is a conditional statement, meaning that you'll execute some action ("buy some") only if the condition ("they have strawberries on sale") is true.

Such conditions are common in programming, as they allow us to implement conditional behavior into our programs. The simplest kind of conditional statement in C++ is called an if statement. An **if statement** allows us to execute one (or more) lines of code only if some condition is true.

The simplest if statement takes the following form:

```
if (condition) true_statement;
```

For readability, this is more often written as following:

```
if (condition)
    true_statement;
```

A **condition** (also called a **conditional expression**) is an expression that evaluates to a Boolean value.

If the condition of an if statement evaluates to Boolean value true, then true_statement is executed. If the condition instead evaluates to Boolean value false, then true_statement is skipped.

A sample program using an if statement

Given the following program:

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    if (x == 0)
        std::cout << "The value is
zero\n";

    return 0;
}
```

Here's output from one run of this program:

```
Enter an integer: 0
The value is zero
```

Let's examine how this works in more detail.

First, the user enters an integer. Then the condition x == 0 is evaluated. The equality operator (==) is used to test whether two values are equal. Operator== returns true if the operands are equal, and false if they are not. Since x has value 0, and 0 == 0 is true, this expression evaluates to true.

Because the condition has evaluated to true, the subsequent statement executes, printing The value is zero.

Here's another run of this program:

```
Enter an integer: 5
```

In this case, x == 0 evaluates to false. The subsequent statement is skipped, the program ends, and nothing else is printed.

Warning

If statements only conditionally execute a single statement. We talk about how to conditionally execute multiple statements in lesson 7.2 -- If statements and blocks (https://www.learncpp.com/cpp-tutorial/if-statements-and-blocks/).

If-else

Given the above example, what if we wanted to tell the user that the number they entered was non-zero?

We could write something like this:

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    if (x == 0)
        std::cout << "The value is zero\n";
    if (x != 0)
        std::cout << "The value is non-
zero\n";

    return 0;
}
```

Or this:

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    bool zero { (x == 0) };
    if (zero)
        std::cout << "The value is zero\n";
    if (!zero)
        std::cout << "The value is non-
zero\n";

    return 0;
}
```

Both of these programs are more complex than they need to be. Instead, we can use an alternative form of the if statement called if-else. If-else takes the following form:

```
if (condition)
    true_statement;
else
    false_statement;
```

If the condition evaluates to Boolean true, true_statement executes. Otherwise false_statement executes.

Let's amend our previous program to use an if-else.

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    if (x == 0)
        std::cout << "The value is zero\n";
    else
        std::cout << "The value is non-
zero\n";

    return 0;
}
```

Now our program will produce the following output:

```
Enter an integer: 0
The value is zero
```

```
Enter an integer: 5
The value is non-zero
```

Chaining if statements

Sometimes we want to check if several things are true or false in sequence. We can do so by chaining an if statement to a prior if-else, like so:

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    if (x > 0)
        std::cout << "The value is
positive\n";
    else if (x < 0)
        std::cout << "The value is
negative\n";
    else
        std::cout << "The value is zero\n";

    return 0;
}
```

The less than operator (<) is used to test whether one value is less than another. Similarly, the greater than operator (>) is used to test whether one value is greater than another. These operators both return Boolean values.

Here's output from a few runs of this program:

```
Enter an integer: 4
The value is positive
```

```
Enter an integer: -3
The value is negative
```

```
Enter an integer: 0
The value is zero
```

Note that you can chain if statements as many times as you have conditions you want to evaluate. We'll see an example in the quiz where this is useful.

Boolean return values and if statements

In the previous lesson (4.9 -- Boolean values (https://www.learncpp.com/cpp-tutorial/boolean-values/)), we wrote this program using a function that returns a Boolean value:

```
#include <iostream>

// returns true if x and y are equal, false otherwise
bool isEqual(int x, int y)
{
    return (x == y); // operator== returns true if x equals y, and false
otherwise
}

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    std::cout << "Enter another integer: ";
    int y {};
    std::cin >> y;

    std::cout << std::boolalpha; // print bools as true or false

    std::cout << x << " and " << y << " are equal? ";
    std::cout << isEqual(x, y); // will return true or false

    return 0;
}
```

Let's improve this program using an if statement:

```
#include <iostream>

// returns true if x and y are equal, false otherwise
bool isEqual(int x, int y)
{
    return (x == y); // operator== returns true if x equals y, and false
otherwise
}

int main()
{
    std::cout << "Enter an integer: ";
    int x {};
    std::cin >> x;

    std::cout << "Enter another integer: ";
    int y {};
    std::cin >> y;

    if (isEqual(x, y))
        std::cout << x << " and " << y << " are equal\n";
    else
        std::cout << x << " and " << y << " are not equal\n";

    return 0;
}
```

Two runs of this program:

```
Enter an integer: 5
Enter another integer: 5
5 and 5 are equal
```

```
Enter an integer: 6
Enter another integer: 4
6 and 4 are not equal
```

In this case, our conditional expression is simply a function call to function isEqual, which returns a Boolean value.

Non-Boolean conditionals

In all of the examples above, our conditionals have been either Boolean values (true or false), Boolean variables, or functions that return a Boolean value. What happens if your conditional is an expression that does not evaluate to a Boolean value?

In such a case, the conditional expression is converted to a Boolean value: non-zero values get converted to Boolean true, and zero-values get converted to Boolean false.

Therefore, if we do something like this:

```
#include <iostream>

int main()
{
    if (4) // nonsensical, but for the sake of example...
        std::cout << "hi";
    else
        std::cout << "bye";

    return 0;
}
```

This will print "hi", since 4 is a non-zero value that gets converted to Boolean true, causing the statement attached to the if to execute.

We'll continue our exploration of if statements in future lesson 7.2 -- If statements and blocks (https://www.learncpp.com/cpp-tutorial/if-statements-and-blocks/).

Quiz time

Question #1

A prime number is a whole number greater than 1 that can only be divided evenly by 1 and itself. Write a program that asks the user to enter a number 0 through 9 (inclusive). If the user enters a number within this range that is prime (2, 3, 5, or 7), print "The digit is prime". Otherwise, print "The digit is not prime".

Show Hint (javascript:void(0))

Show Solution (javascript:void(0))

Question #2

How can the length of the following code be reduced (without changing the formatting)?

```
#include <iostream>

bool isAllowedToTakeFunRide()
{
  std::cout << "How tall are you? (cm)\n";

  double height{};
  std::cin >> height;

  if (height > 140.0)
    return true;
  else
    return false;
}

int main()
{
  if (isAllowedToTakeFunRide())
    std::cout << "Have fun!\n";
  else
    std::cout << "Sorry, you're too short.\n";

  return 0;
}
```

Show Solution (javascript:void(0))

Next lesson
4.11 Chars

Back to table of contents

Previous lesson
4.9 Boolean values

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Find a mistake? Leave a comment!

Avatars from https://gravatar.com/ are connected to your provided email address.

Notify me about replies: POST COMMENT

217 COMMENTS Newest