Chapter 5: Conditionals

At some point in every program you will need to run different portions of code. For example, in a text adventure game the player will decide which direction to take at a fork in the road. We need some way of telling the program to execute a different portion of code depending on this choice. Unsurprisingly C++ has functionality built in for this type of decision-making.

The following table shows the possible statements, which can be used. These will be used with the operators you looked at in the previous chapter. Following on from Table 5.1: Conditional Statements there will be an example depicting each approach in code.

|  |  |
| --- | --- |
| **Statement** | **Description** |
| if statement | An if statement consists of a Boolean expression followed by one or more statements. |
| if…else statement | An if statement can be followed by an optional else statement, which executes when the Boolean expression is false. |
| nested if statements | You can use one if or else statement inside another if or else statement. |
| switch statement | A switch statement allows a variable to be tested for equality against a list of values. |

Table 5.1: Conditional Statements

5.1 Example: if statement

An if statement is the simplest form of condition check. It literally works, as you would expect – If the condition is true, do the following portion of code. If the condition equates to false then the following portion of code is skipped. The portion of code to be executed is defined by the curly braces.

bool a = true;

if ( a == true )

{

// Do this portion of code if a is true.

}

We can reduce the amount of code in the **if statement** of the previous code snippet by removing the ‘== true’ portion. The reason we can do this is that the **if statement** is implicitly asking if the condition is true.

bool a = true;

if ( a )

{

// Do this portion of code if a is true.

}

We can further reduce the code in the above example by removing the curly braces. This will only work if the code to be executed is a single line of code. If there is more than one line of code, only the first will be executed. The following lines will be executed after the **if statement** has exited. This may cause issues later as a syntax error will not be given. This is a logical error and the compiler can not help you.

bool a = true;

if ( a )

// Single line of code can be executed if a is true.

What follows are code snippets demonstrating how curly braces should be used.

bool a = true;

if ( a )

num1++;

num2 += 5;

num3 \*= 2;

is equivalent to:

bool a = true;

if ( a )

{

num1++;

}

num2 += 5;

num3 \*= 2;

The correct code is:

bool a = true;

if ( a )

{

num1++;

num2 += 5;

}

num3 \*= 2;

5.2 Example: if statement using Operators

Within an **if statement** brackets you can use any of the relational or logical operators as detailed in the previous chapter. Below are some examples.

Relational Greater than

If( num1 > 10 ) {}

If( num1 > num2 ) {}

Relational Equal to

If( num1 == 10 ) {}

If( num1 == num2 ) {}

Relational Not Equal to

If( num1 != 10 ) {}

If( num1 != num2 ) {}

Logical AND

If( a&& b ) {}

Logical OR

If( a|| b ) {}

Logical NOT

If( !a ) {}

5.3 Example: if…else statement

We can extend an **if statement** to include a portion of code to be executed in the event of the condition being false. In English we are saying, if the condition is true do the first portion of code, otherwise do the second portion of code. This extra portion of code will only ever be executed if the condition is false.

bool a = true;

if ( a )

{

// Do this portion of code if a is true.

}

else

{

// Do this portion of code if a is false.

}

Again, given only a single line of code to be executed we can remove the curly braces to reduce the code.

bool a = true;

if ( a )

// Do this portion of code if a is true.

else

// Do this portion of code if a is false.

5.4 Example: if…else if statement

Building on from the if…else statement, we can add another condition to the second portion of code. Here we are saying, if the first condition is true then do the first portion of code, otherwise check if the second condition is true, and if so do the second portion of code, and if neither are true, do the third portion of code.

bool a = false;

bool b = true;

if ( a )

{

// Do this portion of code if a is true.

}

else if( b )

{

// Do this portion of code if b is true.

}

else

{

// Do this portion of code if a & b are both false.

}

Note: If a condition equates to true, the corresponding code portion will be executed. All others will be ignored.

Again, given only a single line of code to be executed we can remove the curly braces to reduce the code.

5.5 Example: nested if statements

Any legal code can go inside the **if statement** portion of code to be executed, therefore it is possible to have any of the above variants of an if statement inside an **if statement** code block.

bool a = true;

bool b = true;

if ( a )

{

// Do this portion of code if a is true.

if( b )

{

// Do this portion of code if b is true.

}

}

else

{

// Do this portion of code if a is false. B will never have been tested.

}

5.6 Example: switch statements

When you have a lot of options multiple if statements can become messy. Take the text based adventure example again. The player could enter numerous responses to the on screen prompt and you as programmer would need to code an if statement for each one. Alternatively you can use a switch statement. A switch statement allows a variable to be tested against a list of constant values.

Int playerInput;

// Read in the players response from the keyboard.

cin >> playerInput;

switch( playerInput )

{

case 0:

// Add code to execute if the player entered ‘0’

break;

case 1:

// Add code to execute if the player entered ‘1’

break;

default:

// Add code to execute if non of the above options matched the player input.

break;

}

As you can see from the above code snippet, the variable to be tested against goes into the brackets beside the **switch** keyword, then within the curly braces are a list of values to compare against. We are saying, in the case of playerInput matching with ‘0’ do this portion of code. It is crucial to include the **break** keyword, otherwise code execution will automatically drop through to the next section without checking the condition.

The last part of a **switch statement** to note is the **default** case. This states that if none of the values match then do this portion of code. It acts in the same way as an else portion of an **if…else statement**.

There may be occasions where you wish a portion of code to execute if several conditions are met, in the same manner as an **if statement** which uses the **Logical OR** operator. In this case follow the below format.

Int playerInput;

// Read in the players response from the keyboard.

cin >> playerInput;

switch( playerInput )

{

case 0:

case 1:

// Add code to execute if the player entered ‘0’ or ‘1’

break;

default:

// Add code to execute if non of the above options matched the player input.

break;

}

**Nested switch statements** work in much the same way **as nested if statements** do.

**Program 9: Share of Savings**

This is for you to complete. You need to add the code listing and screen shot for this program here. It is crucial that you do this to keep your portfolio as a useful resource as the end of year test will require a selection of these programs.

Pretend that, for some reason, you are now entitled to half my savings (☹) and we’ve written a program that helps calculate how much of that money is yours. The code might look like:

#include <iostream>

using namespace std;

int main()

{

int mySavings = 2000;

int yourPercentage = 50;

int yourShare= ( mySavings \* (yourPercentage / 100) );

cout << “Your share: “ << yourShare << endl;

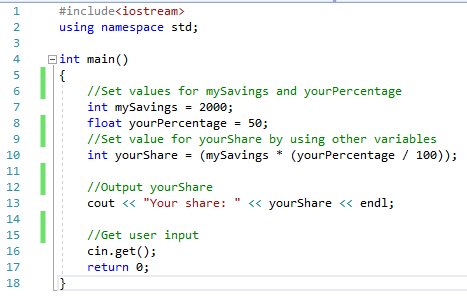
cin.get();

return 0;

}

Run this program in Visual studio and see if you can understand why we get a wrong answer printed out instead of the correct answer of 1000. Then try and fix the error! Note there are multiple ways of fixing this and the quickest solution requires only 2 keystrokes!

Add the fixed code below along with a screenshot of the output.





**Program 10: Share of Haribo**

This is for you to complete. You need to add the code listing and screen shot for this program here. It is crucial that you do this to keep your portfolio as a useful resource as the end of year test will require a selection of these programs.

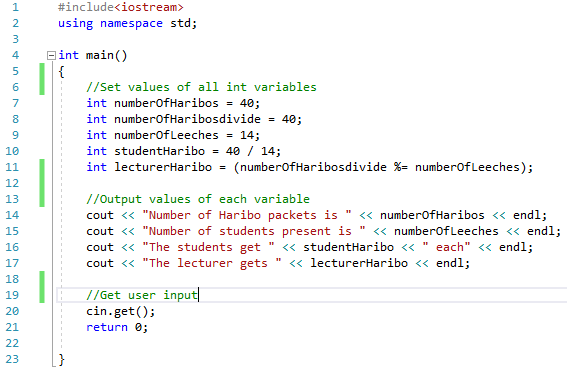
Pretend that, for some reason, you are now entitled to my stash of Haribo (☹). There are **40** packets available and to make things fair, they are to be shared out **equally**, as in you all have to have the same number of packets.

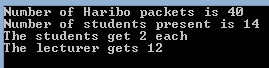
On the day when I decide to hand out the sweet sweet Haribo, only **14** students turned up (pretend that a popular video game had been released that day and many students were suddenly too “ill” to turn up).

Write a program that

1. Calculates how many of the **40** packets of Haribo do each of the students get?
2. Calculates how many I get, which will be the remainder left after sharing the 40 packets as equally as possible

Your final answers should be **2** packets for each student, leaving **12** packets left for me. What we want to see is you coding the solution and getting the computer to do the right calculations that come up with these 2 answers. We’re not bothered about the answers themselves, similar to how many mathematics questions are more concerned with “method marks” than the final answer.





**Program 11: Weather Menu**

This is for you to complete. You need to add the code listing and screen shot for this program here. It is crucial that you do this to keep your portfolio as a useful resource as the end of year test will require a selection of these programs.

Write a program that displays a short menu such as:

* + Please choose an option: 1. Sunny 2. Cloudy 3. Raining 4. Exit

The program should use a **switch statement** to display a short message appropriate to the option chosen (such as "Don't forget your sunscreen").

For an example of reading input from the player, take a look at Game 1: Funny Headlines in the Appendices.

