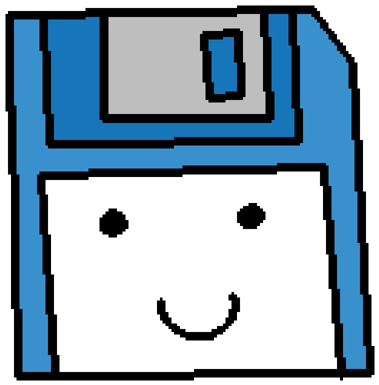
Flow of Control: Branches

Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. http://creativecommons.org/licenses/by-nc-sa/3.0/

Written by Rachel J. Morris, last updated 2015-01-14

Boolean Expressions



Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
License. http://creativecommons.org/licenses/by-nc-sa/3.0/

Written by Rachel J. Morris, last updated 2015-01-14

Boolean Expressions

Before we work with “if, then” statements, we need to look at **Boolean Expressions.**

These are expressions that can either result in **true** or **false**.

Boolean Expressions

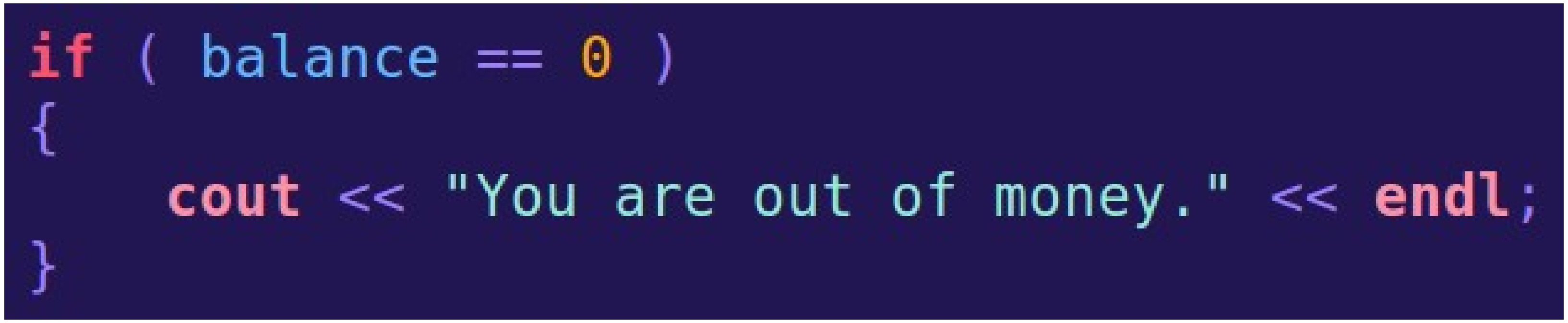
Here is a summary of boolean expressions

Conditional	C++
Equal to	`( var1 == var2 )`
Not equal to	`( var1 != var2 )`
Less than	`( var1 < var2 )`
Less than or equal to	`( var1 <= var2 )`
Greater than	`( var1 > var2 )`
Greater than or equal to	`( var1 >= var2 )`
True	`( var1 == true )`
True, shorthand	`( var1 )`
False	`( var1 == false )`
False, shorthand	`( !var1 )`

Boolean Expressions

Equal To

```
if ( var1 == var2 )
```



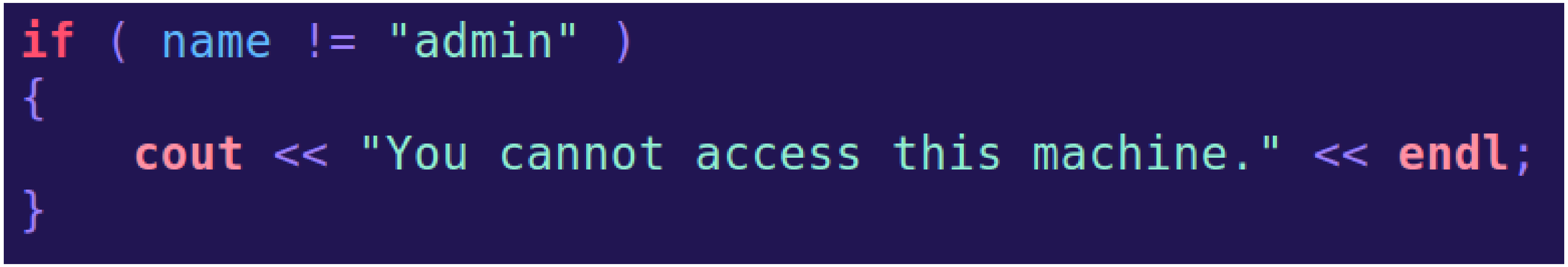
```
if ( balance == 0 )
{
    cout << "You are out of money." << endl;
}
```

Compare whether two variables are equal, or if a variable equals some value,
(or if one value equals another value)
With the equality comparison operator **==**.

Boolean Expressions

Not Equal To

`if ( var1 != var2 )`



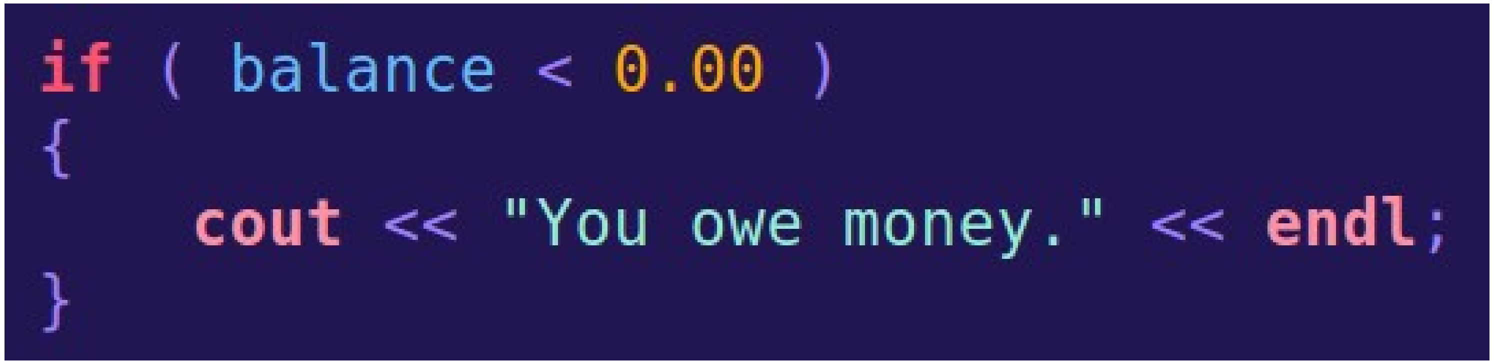
```
if ( name != "admin" )
{
    cout << "You cannot access this machine." << endl;
}
```

Likewise, you can check whether things are ***not*** equivalent with the **`!=`** operator.

Boolean Expressions

Greater than / Less than

```
if ( var1 > var2 )
if ( var1 < var2 )
```



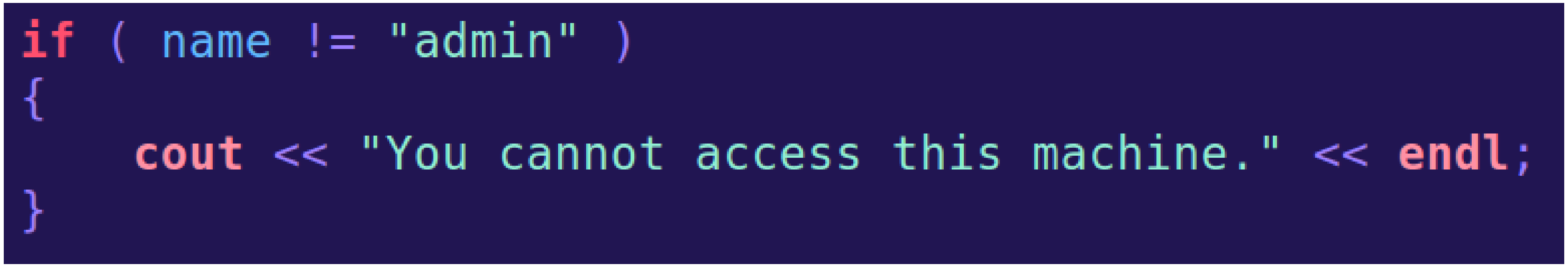
```
if ( balance < 0.00 )
{
    cout << "You owe money." << endl;
}
```

Used to compare numbers.

Boolean Expressions

Not Equal To

`if ( var1 != var2 )`



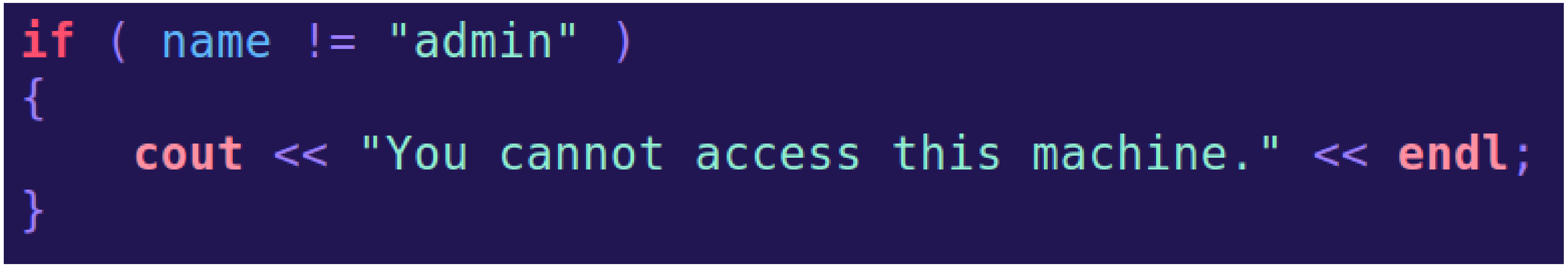
```
if ( name != "admin" )
{
    cout << "You cannot access this machine." << endl;
}
```

Likewise, you can check whether things are ***not*** equivalent with the **`!=`** operator.

Boolean Expressions

Not Equal To

`if ( var1 != var2 )`



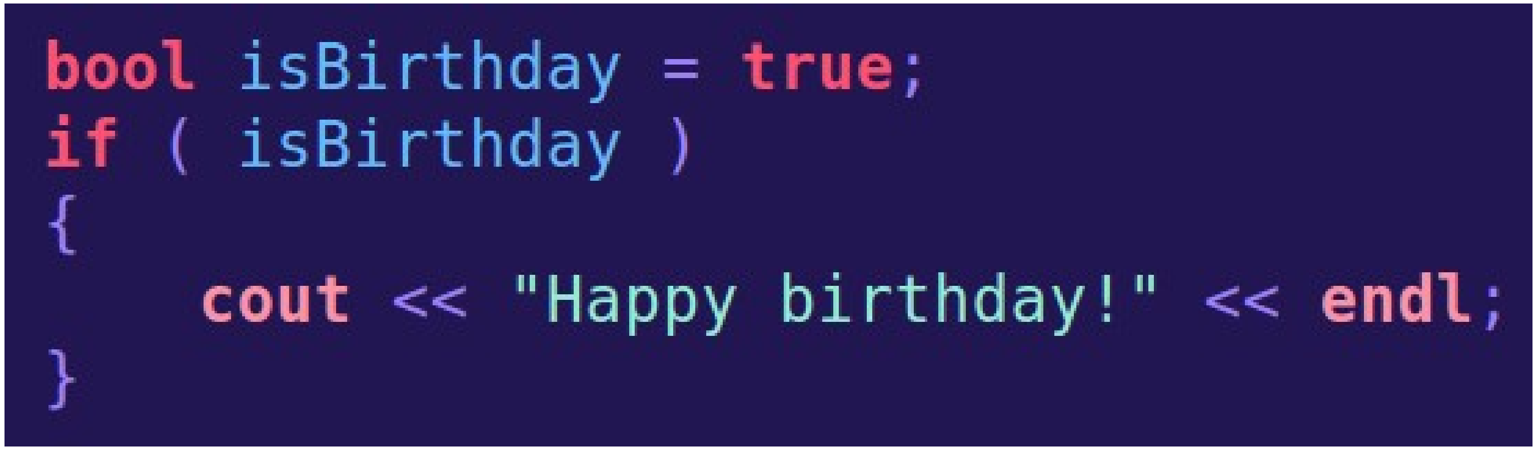
```
if ( name != "admin" )
{
    cout << "You cannot access this machine." << endl;
}
```

Likewise, you can check whether things are ***not*** equivalent with the **`!=`** operator.

Boolean Expressions

If True

```
if ( var1 == true )
if ( var1 )
```



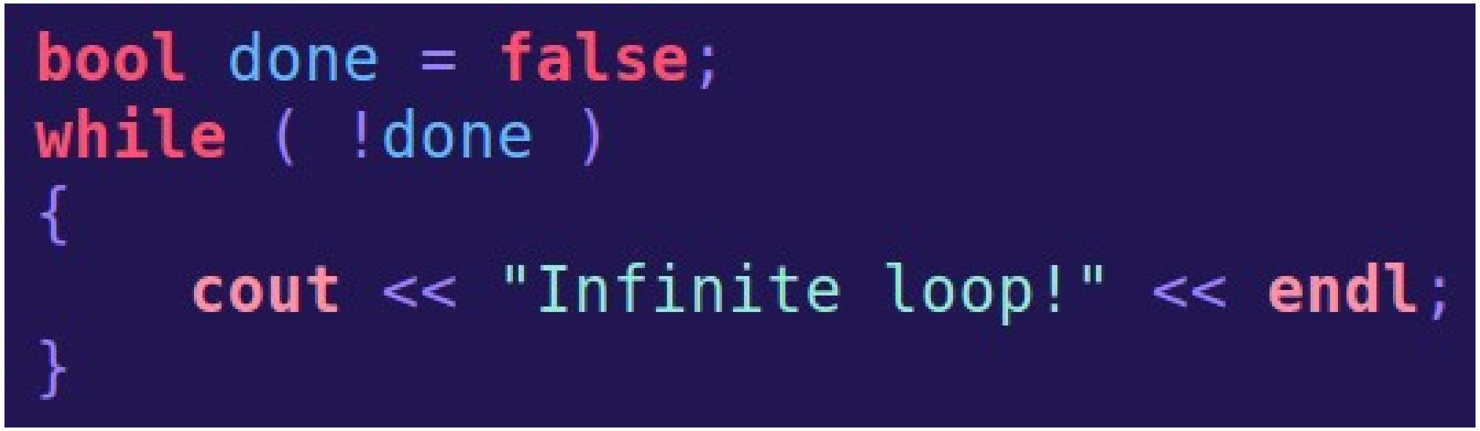
```
bool isBirthday = true;
if ( isBirthday )
{
    cout << "Happy birthday!" << endl;
}
```

Used for booleans, or expressions that result in booleans (such as boolean expressions!)

Boolean Expressions

If False

```
if ( var1 != true )
if ( !var1 )
```

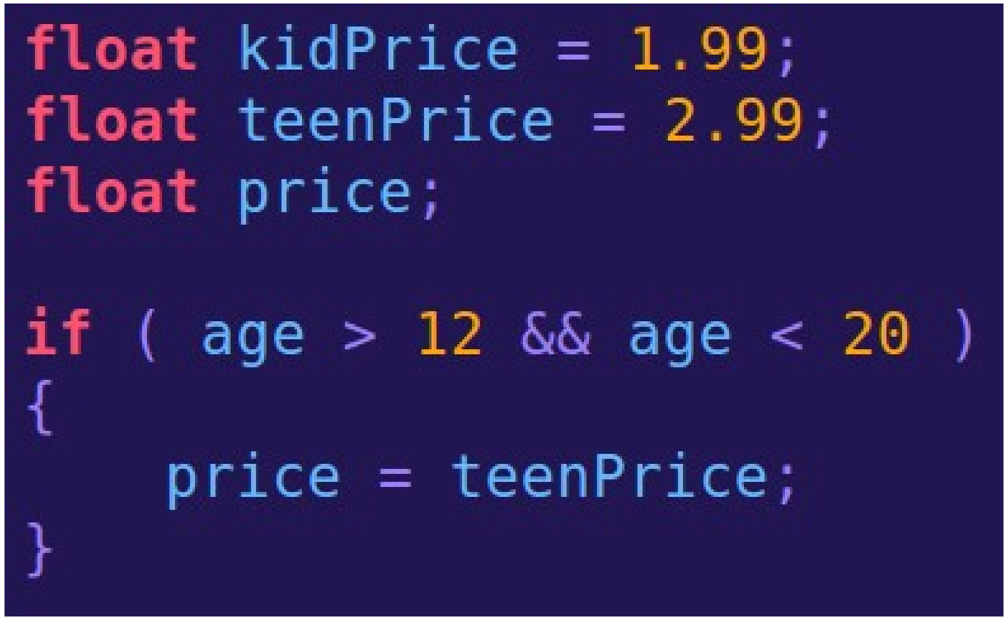


```
bool done = false;
while ( !done )
{
    cout << "Infinite loop!" << endl;
}
```

Used for booleans, or expressions that result in booleans (such as boolean expressions!)

Boolean Expressions

We can also string together any amount of boolean expressions with the **and** **`&&`** and **or** **||** operators.

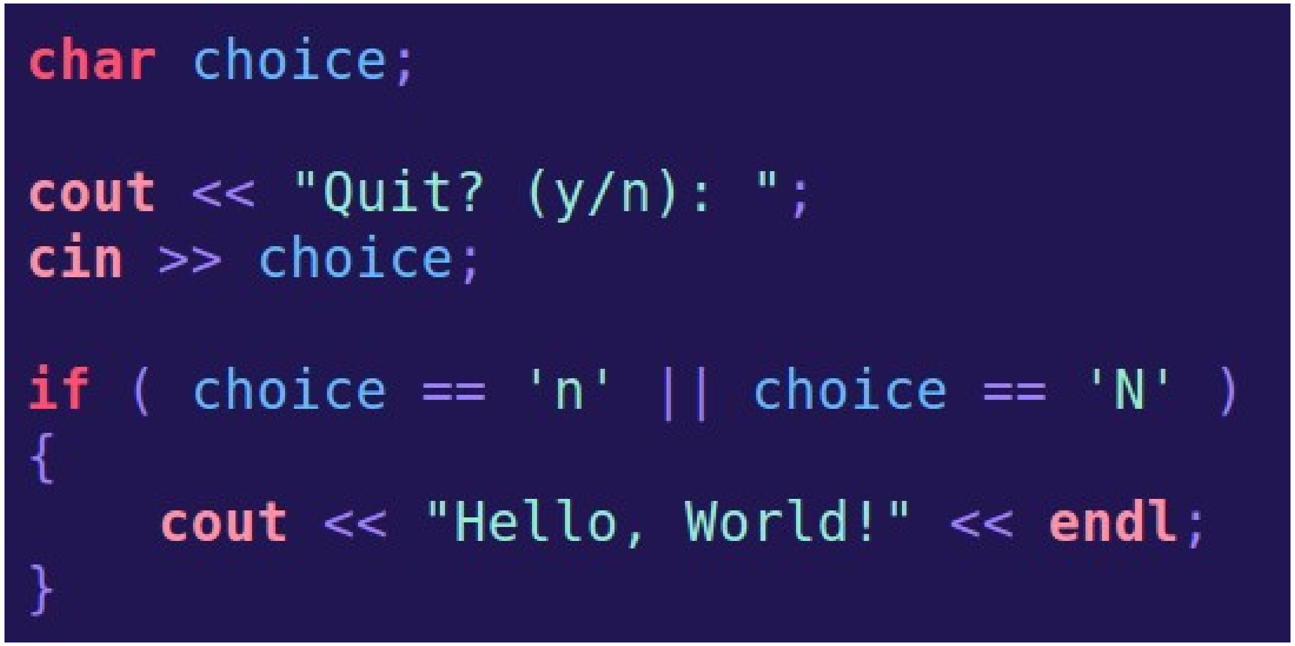


```
float kidPrice = 1.99;
float teenPrice = 2.99;
float price;

if ( age > 12 && age < 20 )
{
    price = teenPrice;
}
```

If age is between 12 and 20.
12 < age < 20

Boolean Expressions

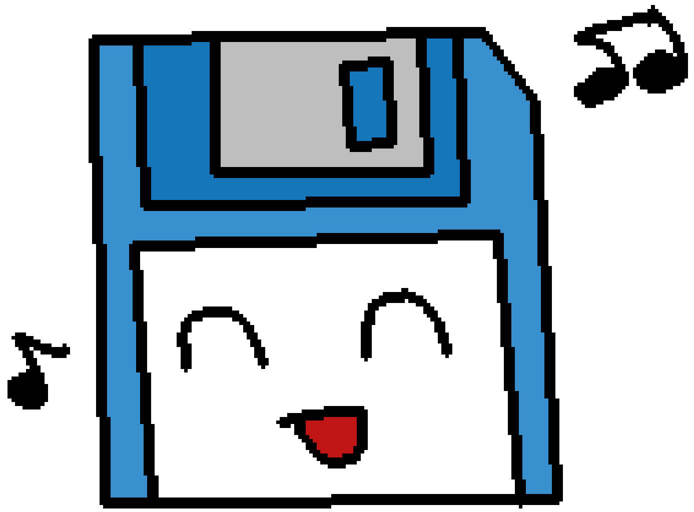
We can also string together any amount of boolean expressions with the **and** **`&&`** and **or** **||** operators.



```
char choice;

cout << "Quit? (y/n): ";
cin >> choice;

if ( choice == 'n' || choice == 'N' )
{
    cout << "Hello, World!" << endl;
}
```

If input is lower-case or upper-case “N”

If Statements



Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
License. http://creativecommons.org/licenses/by-nc-sa/3.0/

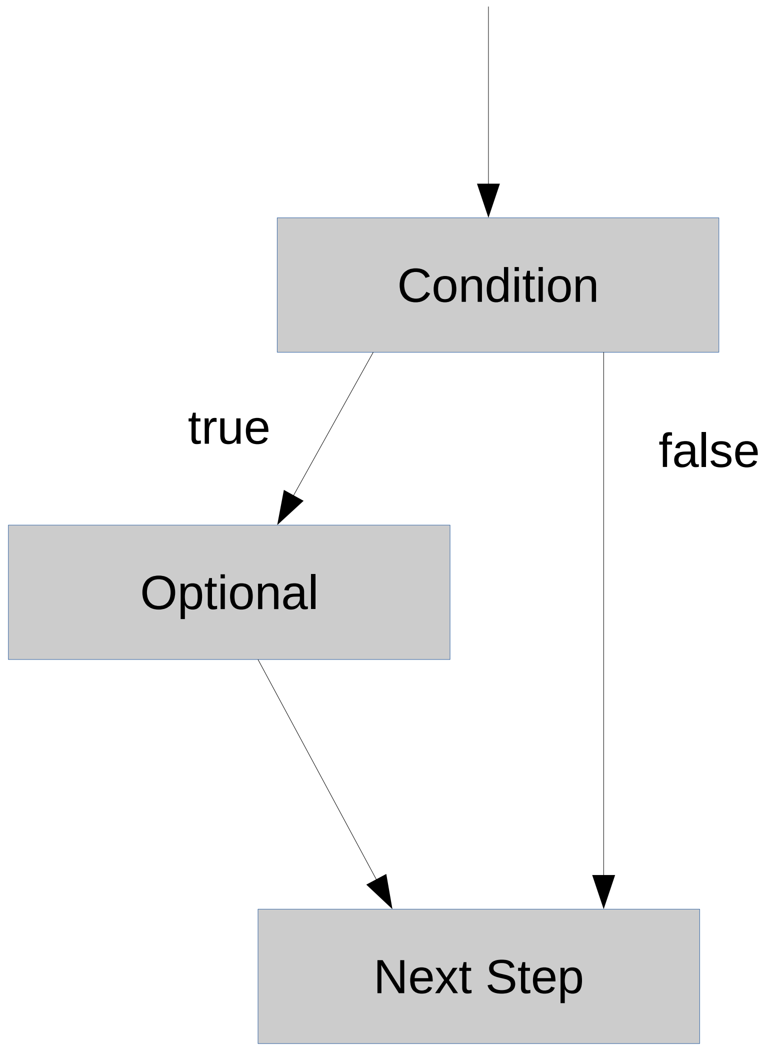
If Statements

One of the main methods of branching logic in your program is through **if statements**.

If Statements

IF

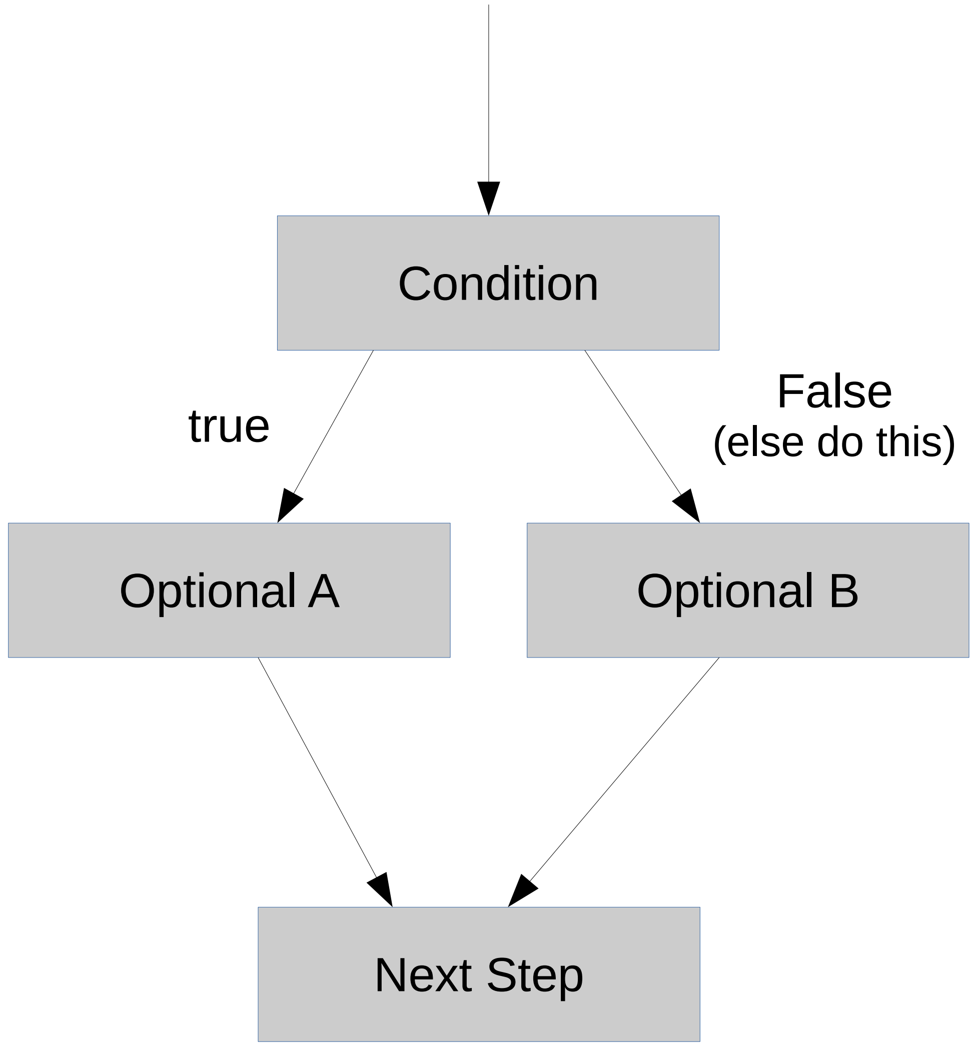
You can use an if statement to check one condition, and skip over a block of code when false



If Statements

IF - ELSE

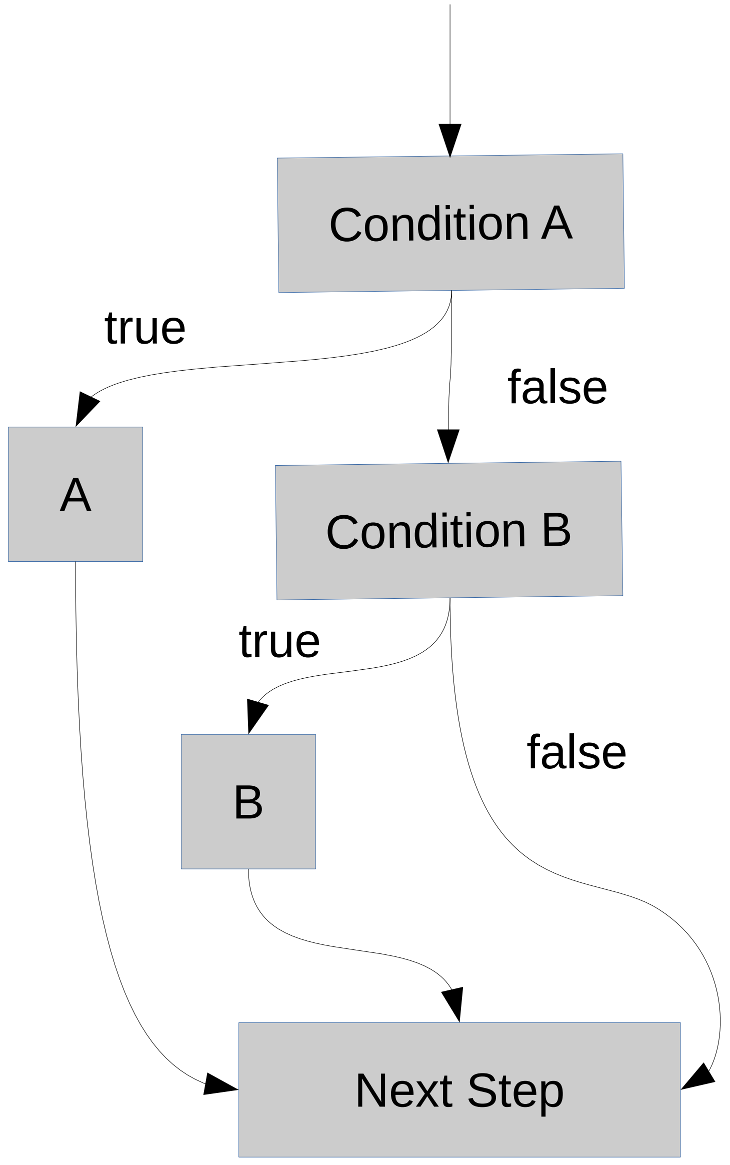
You can test one condition and have two separate outcomes with IF – ELSE.



If Statements

IF – ELSE IF

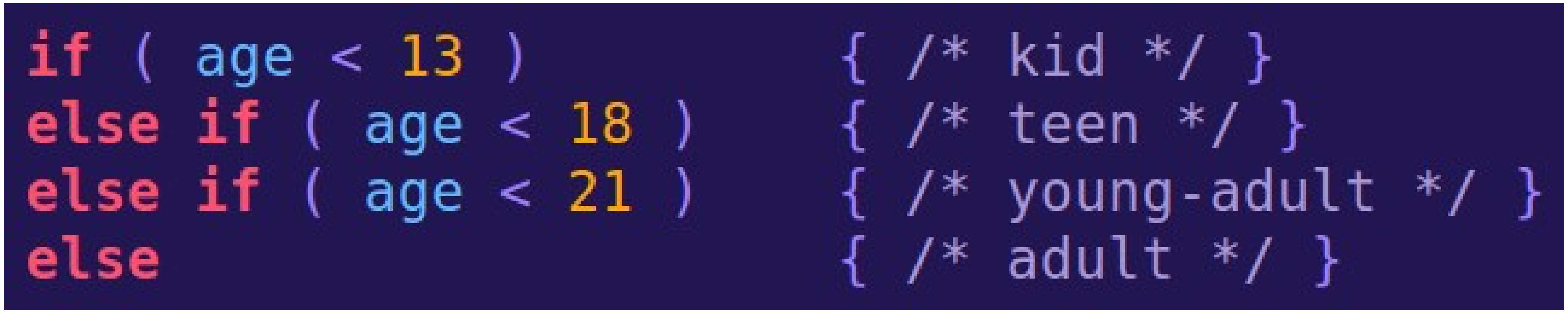
You can test multiple conditions with IF – ELSE IF (and, optionally, ELSE at the end for a “default” result)



If Statements

IF – ELSE IF

You can chain as many **else if** conditionals as you would like, but there is only one **if** and only one (or zero) **else** for an if-else if statement.



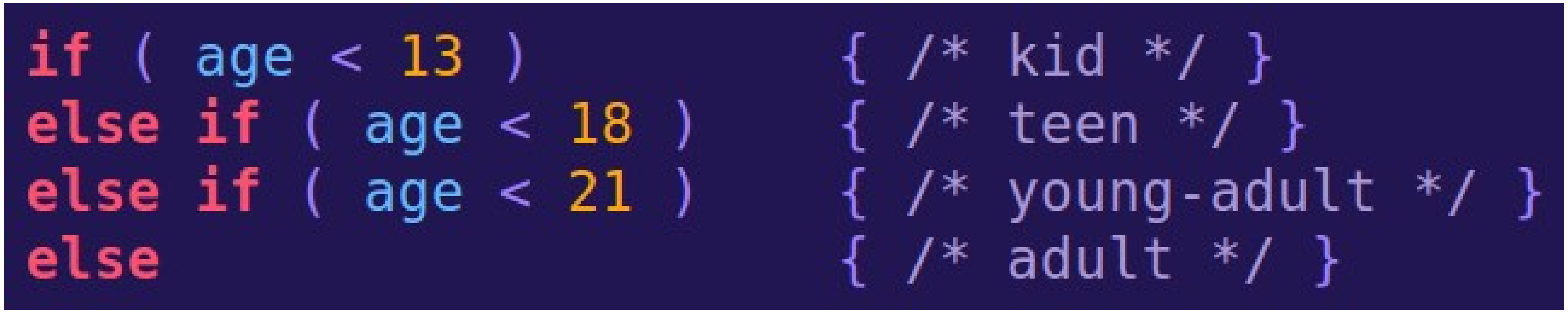
```
if ( age < 13 )           { /* kid */ }
else if ( age < 18 )      { /* teen */ }
else if ( age < 21 )      { /* young-adult */ }
else                      { /* adult */ }
```

If Statements

IF – ELSE IF

With related if statements, whichever one evaluates to **true** first is the one that is executed, even if subsequent **else if** statements would also be true.

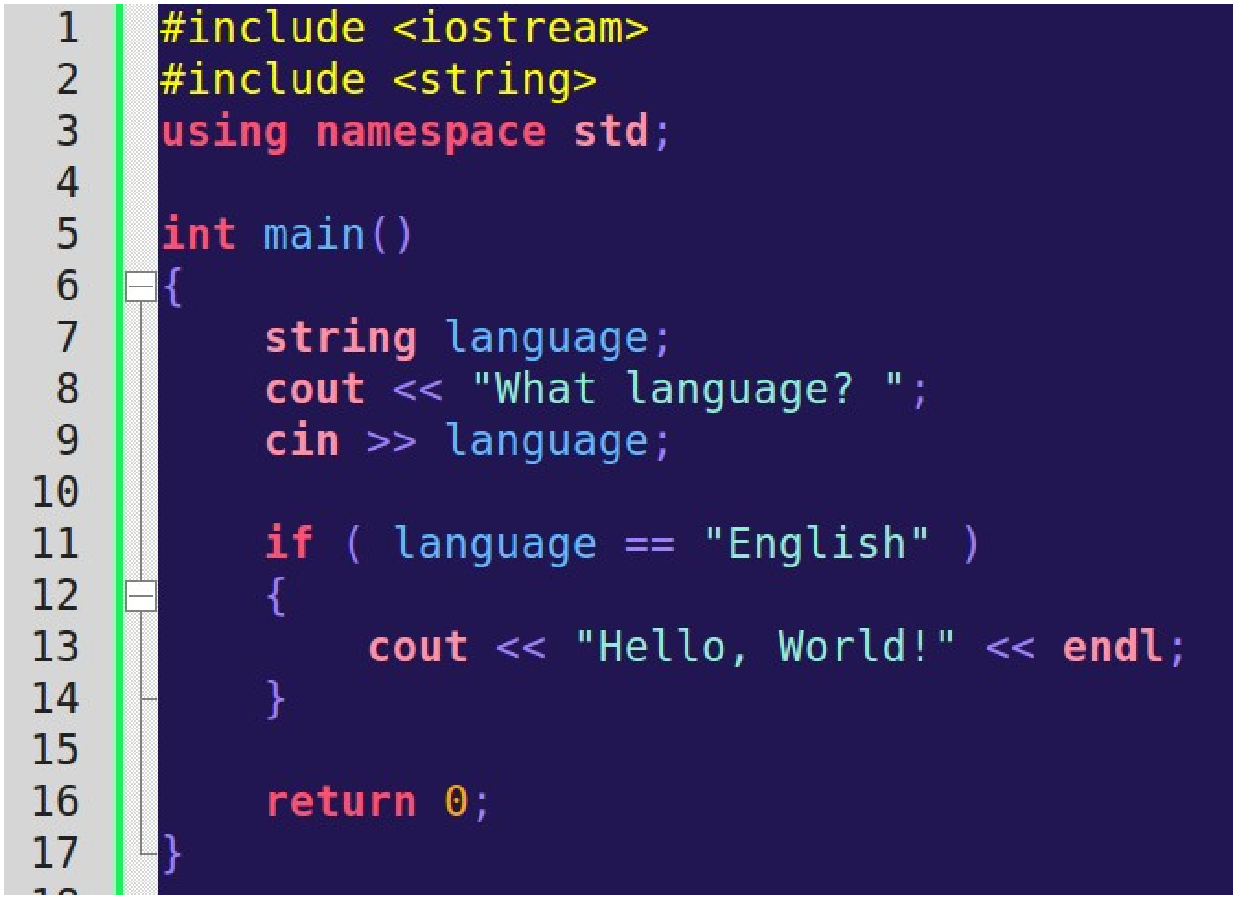
Here, if your age is 12, the first line would trigger, even though line 2 and line 3 are also true.



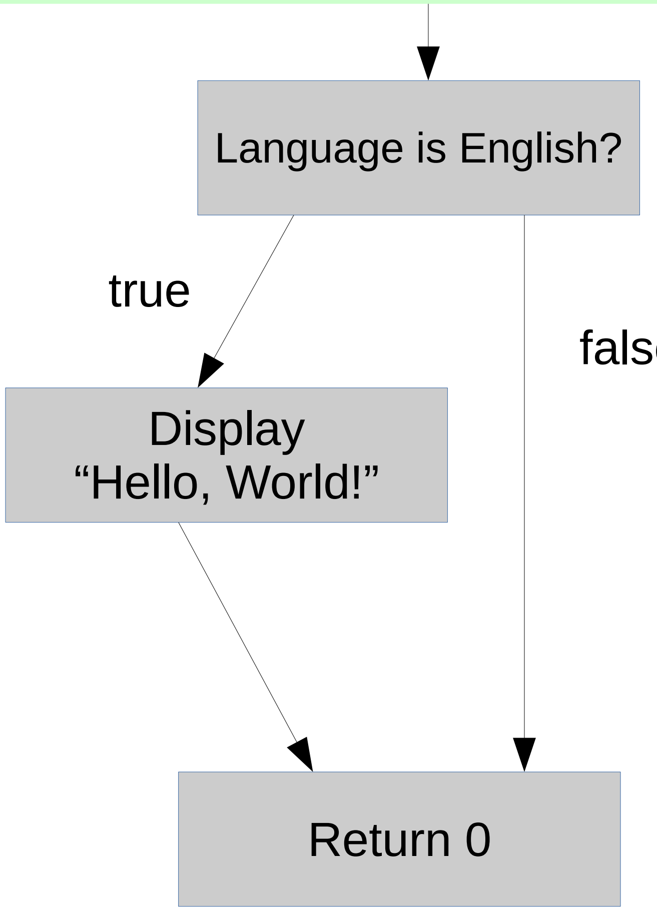
```
if ( age < 13 )            { /* kid */ }
else if ( age < 18 )       { /* teen */ }
else if ( age < 21 )       { /* young-adult */ }
else                       { /* adult */ }
```

If Statements

If Statement



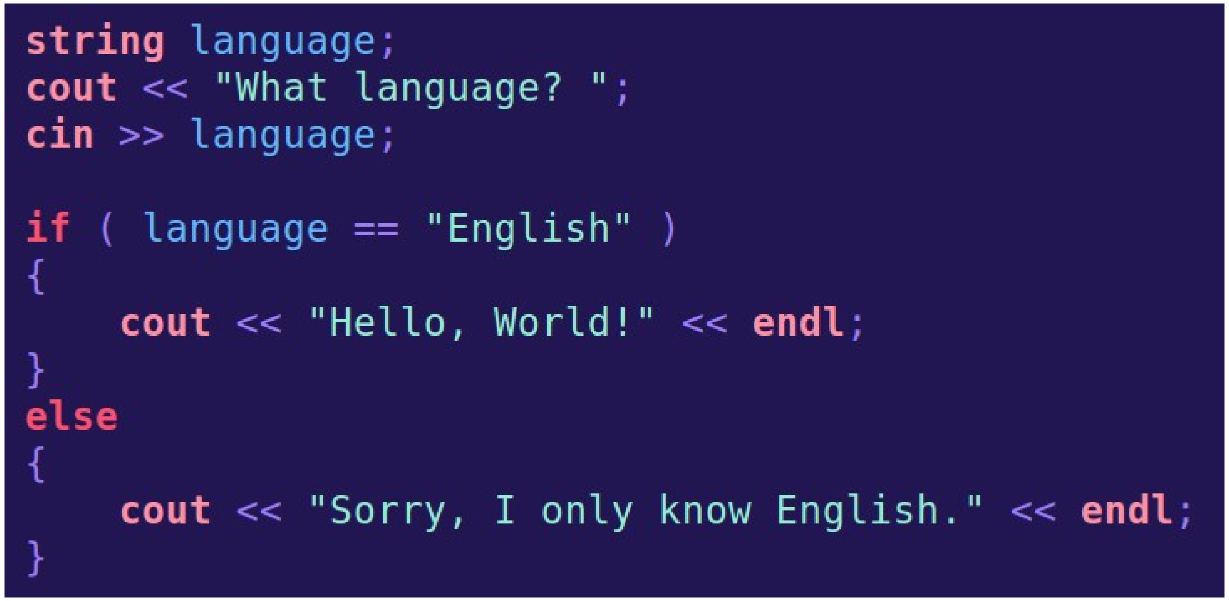
```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string language;
    cout << "What language? ";
    cin >> language;

    if ( language == "English" )
    {
        cout << "Hello, World!" << endl;
    }

    return 0;
}
```



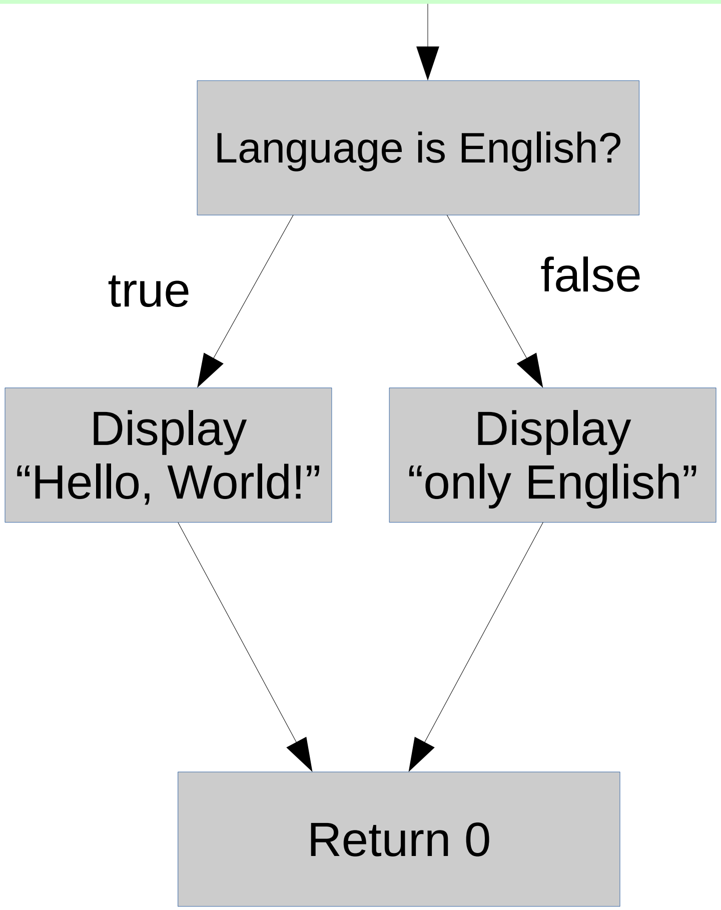
Here, the user enters a language. They only get a Hello, World message if they enter “English”.

If Statements

If – Else Statement



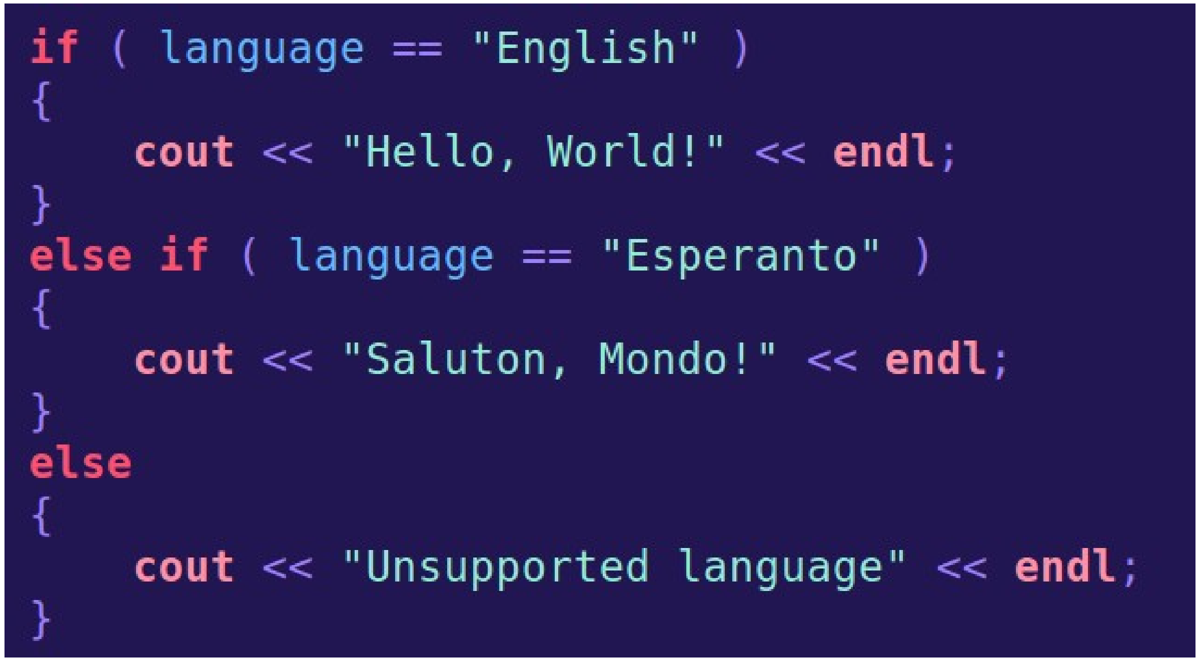
```
string language;
cout << "What language? ";
cin >> language;

if ( language == "English" )
{
    cout << "Hello, World!" << endl;
}
else
{
    cout << "Sorry, I only know English." << endl;
}
```



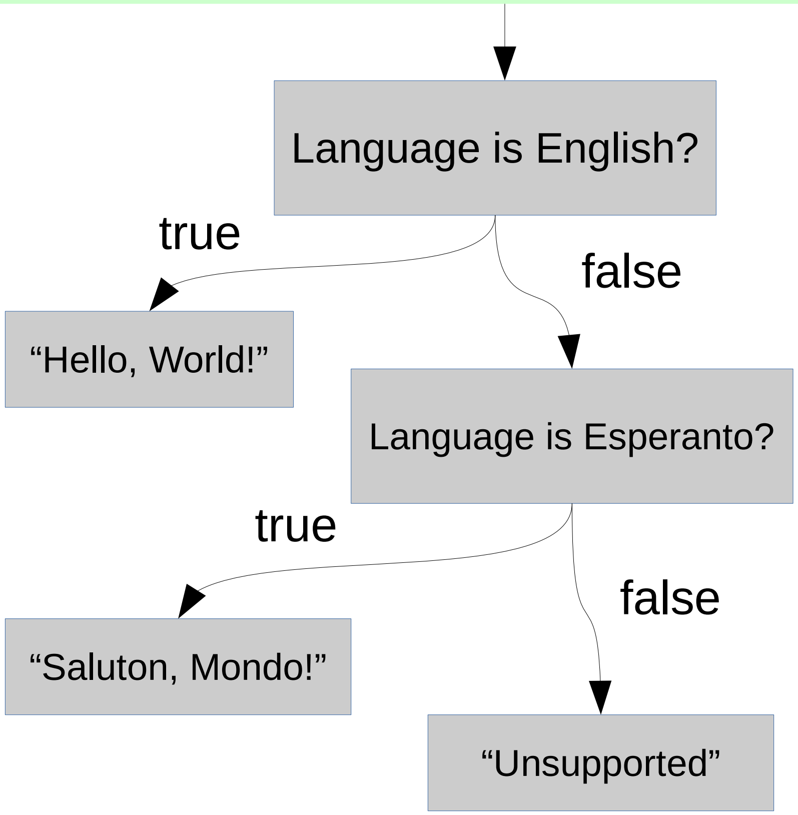
We can add an **else** statement, to give some default message.

If Statements

If – Else If – Else Statement



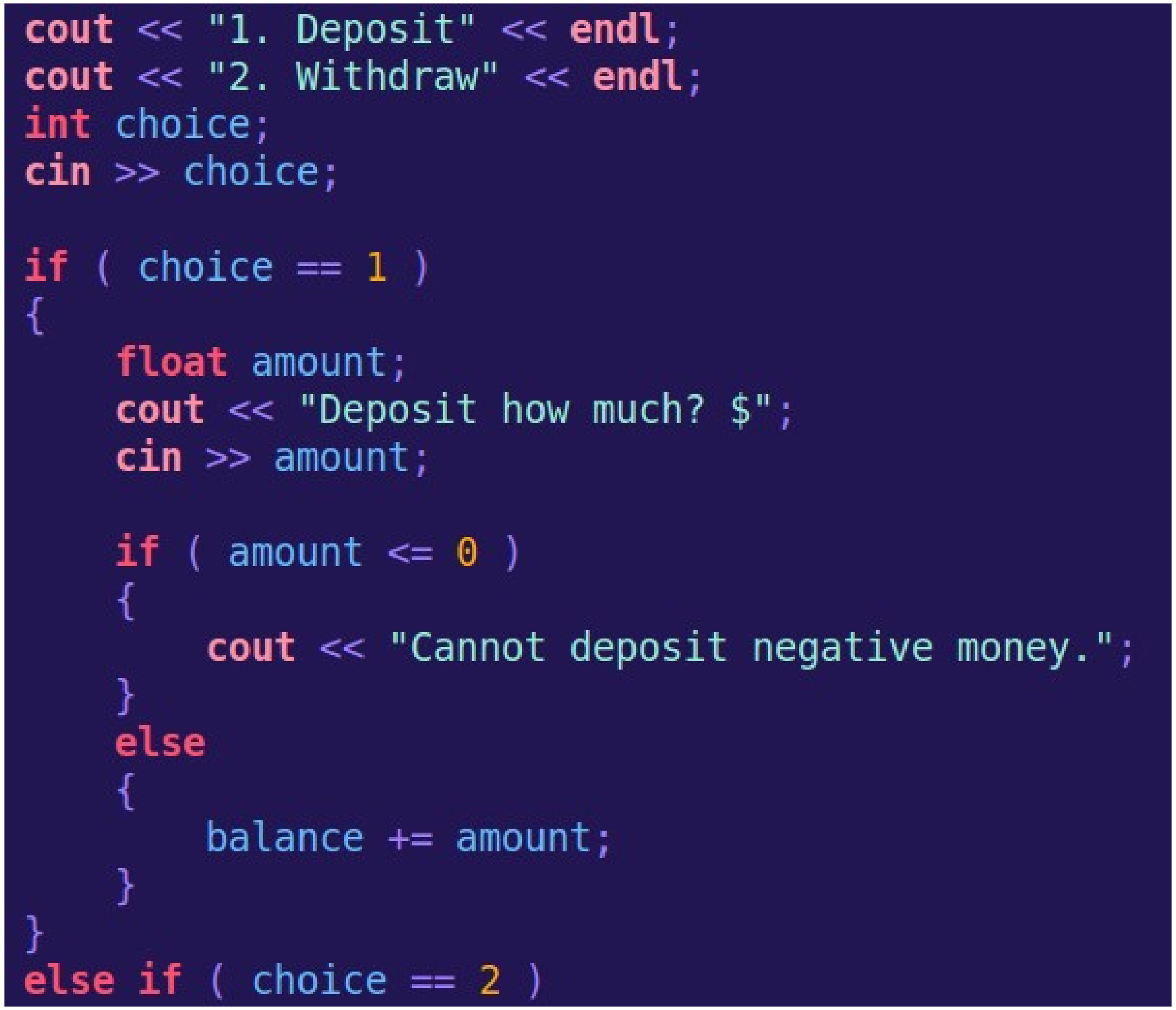
```
if ( language == "English" )
{
    cout << "Hello, World!" << endl;
}
else if ( language == "Esperanto" )
{
    cout << "Saluton, Mondo!" << endl;
}
else
{
    cout << "Unsupported language" << endl;
}
```



Now we have two language options and the default.
We can keep adding options with more **else if** statements.

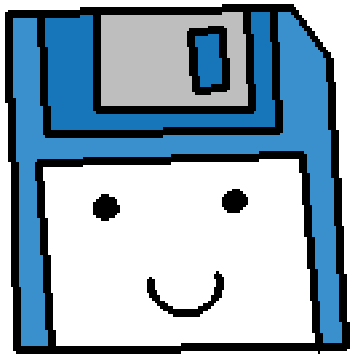
If Statements

Nesting If Statements



```
cout << "1. Deposit" << endl;
cout << "2. Withdraw" << endl;
int choice;
cin >> choice;

if ( choice == 1 )
{
    float amount;
    cout << "Deposit how much? $";
    cin >> amount;

    if ( amount <= 0 )
    {
        cout << "Cannot deposit negative money.";
    }
    else
    {
        balance += amount;
    }
}
else if ( choice == 2 )
```

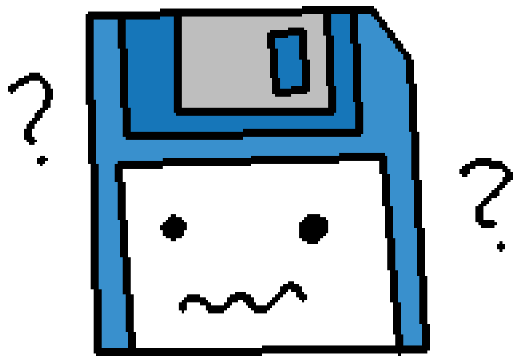
You can also contain if statements within other if statements. This allows additional branches in your program logic.



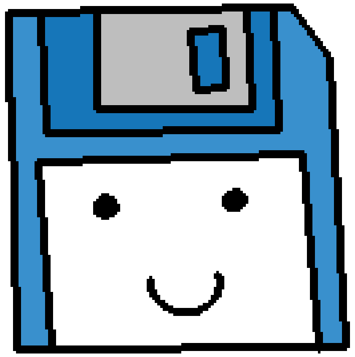
If Statements

Nesting If Statements

Just be careful to not over-clutter your program.
Later on, we will write functions to delegate tasks to other areas,
instead of crowding **`main()`** with all the program logic.



Switches



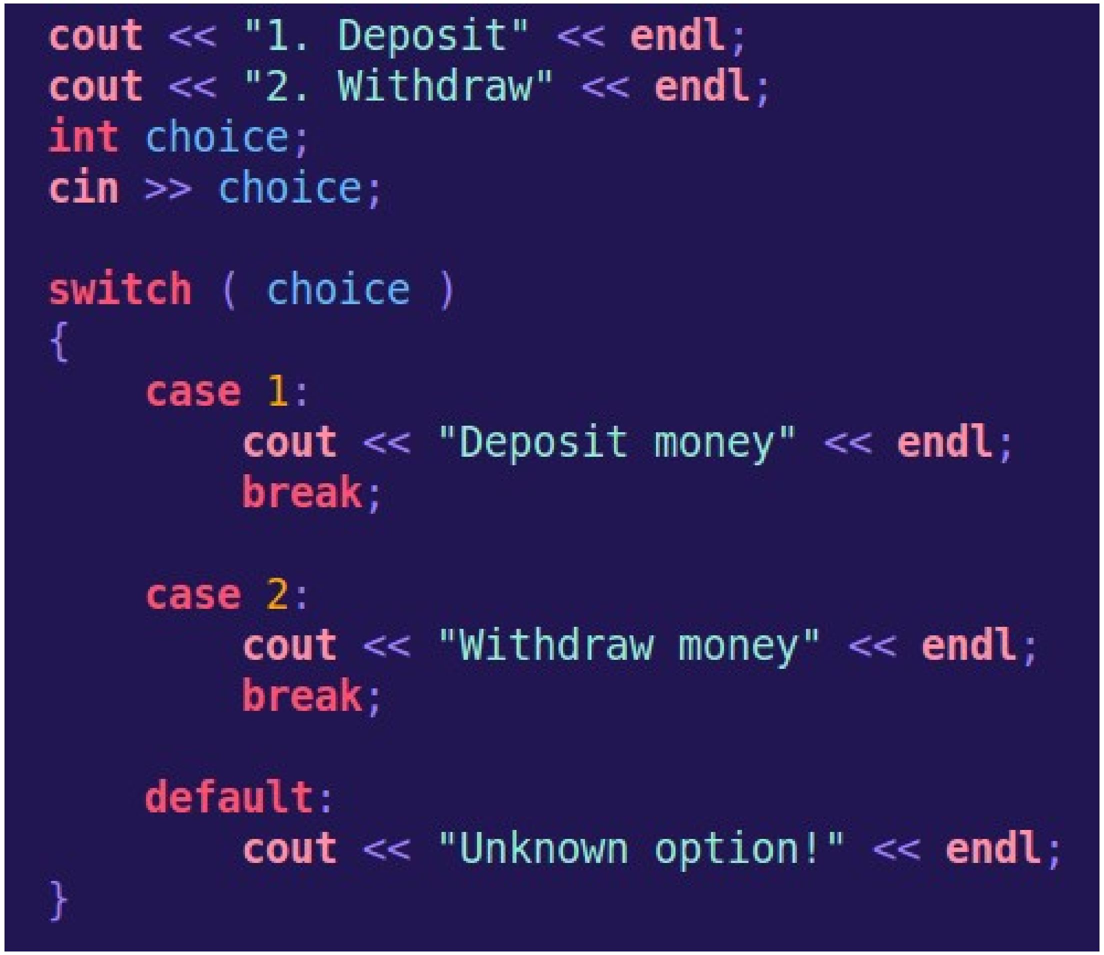
Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. http://creativecommons.org/licenses/by-nc-sa/3.0/

Switches

Another way to check conditions and branch for different values are with **switch statements**.

Switch statements can test boolean expressions, enums, integers, or characters.

They look a little odd compared to if statements.

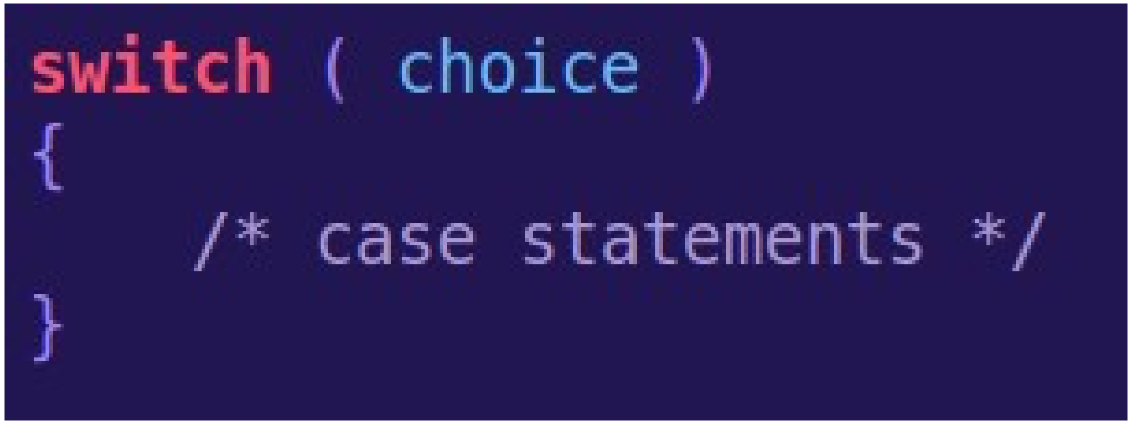


```
cout << "1. Deposit" << endl;
cout << "2. Withdraw" << endl;
int choice;
cin >> choice;

switch ( choice )
{
    case 1:
        cout << "Deposit money" << endl;
        break;

    case 2:
        cout << "Withdraw money" << endl;
        break;

    default:
        cout << "Unknown option!" << endl;
}
```

Switches

Switch statements start with the **`switch`** keyword.

The **controlling expression** is within parenthesis here.

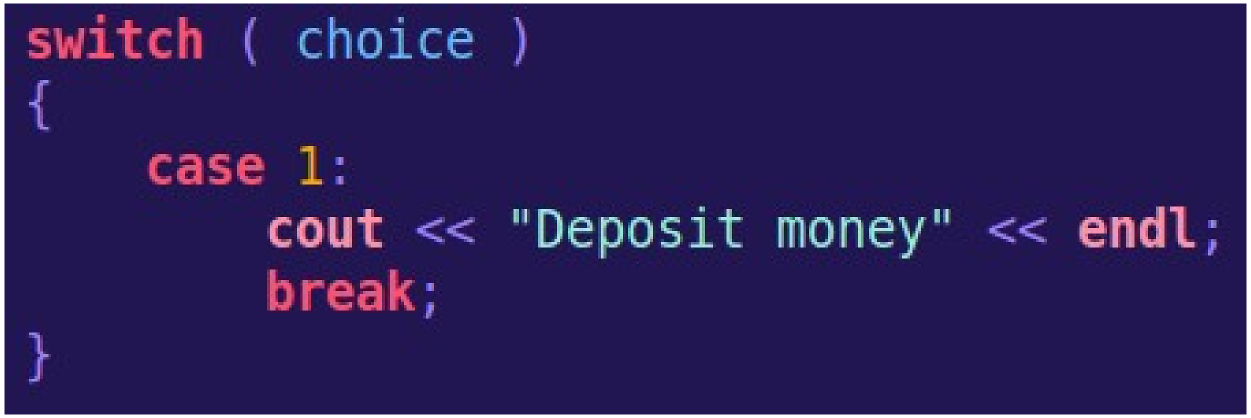


```
switch ( choice )
{
    /* case statements */
}
```

The case statements that we write will be based on the value of the **controlling expression**. In this case, the expression is what the value of the integer **`choice`** is.

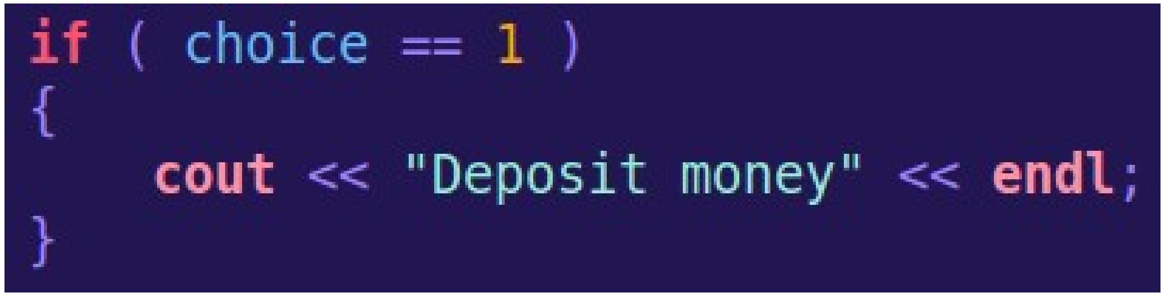
Switches

We can add **case** statements. After the keyword **case**, you specify the value we're testing for.



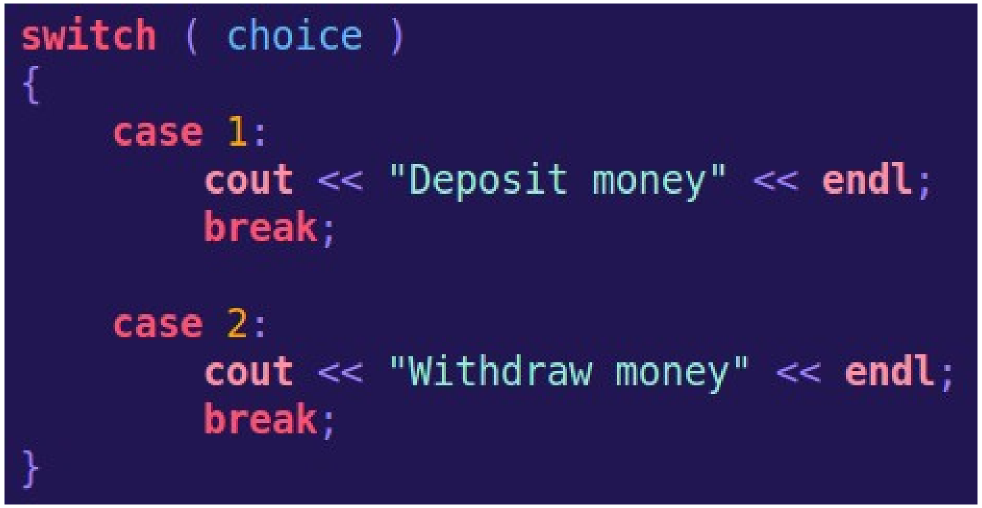
```
switch ( choice )
{
    case 1:
        cout << "Deposit money" << endl;
        break;
}
```

The **switch** statement and this if statement do the same thing.



```
if ( choice == 1 )
{
    cout << "Deposit money" << endl;
}
```

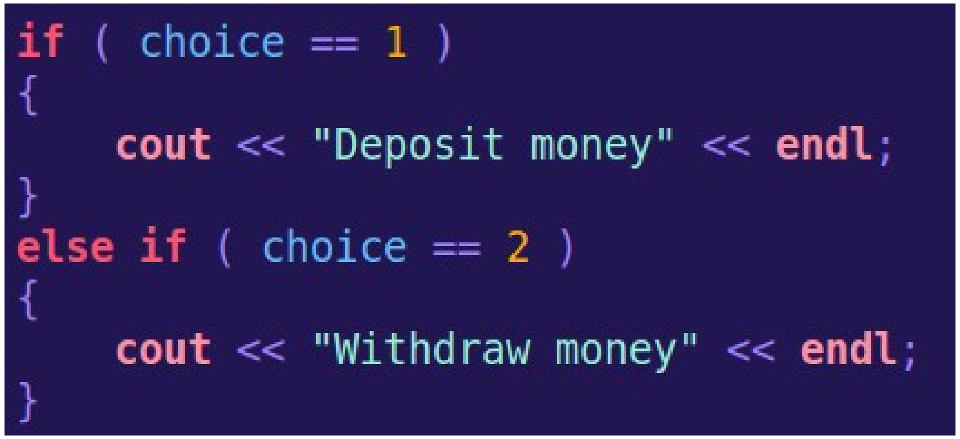
Switches



```
switch ( choice )
{
    case 1:
        cout << "Deposit money" << endl;
        break;

    case 2:
        cout << "Withdraw money" << endl;
        break;
}
```

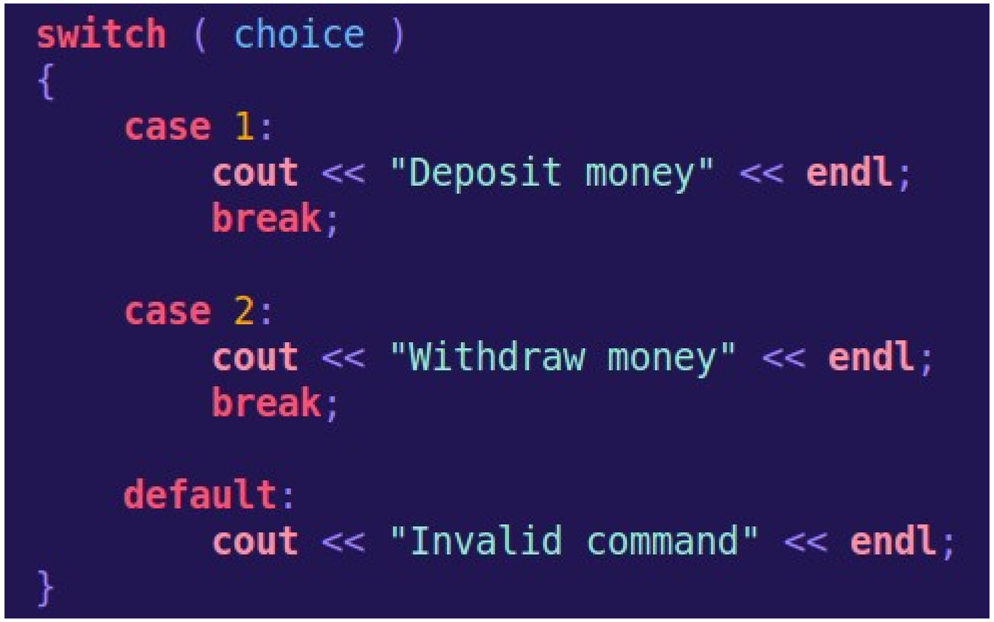
Notice that the case statements have a **`break;`** command before the next case statement.

When the **`break;`** is hit, it exits the **`switch`** statement. That way, it doesn't execute additional code under another **`case`**.



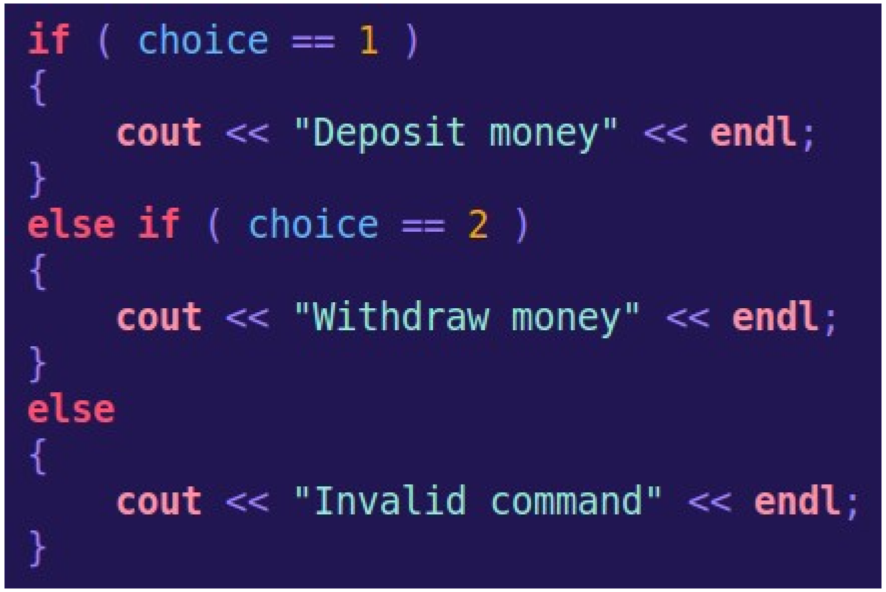
```
if ( choice == 1 )
{
    cout << "Deposit money" << endl;
}
else if ( choice == 2 )
{
    cout << "Withdraw money" << endl;
}
```

Switches



```
switch ( choice )
{
    case 1:
        cout << "Deposit money" << endl;
        break;

    case 2:
        cout << "Withdraw money" << endl;
        break;

    default:
        cout << "Invalid command" << endl;
}
```

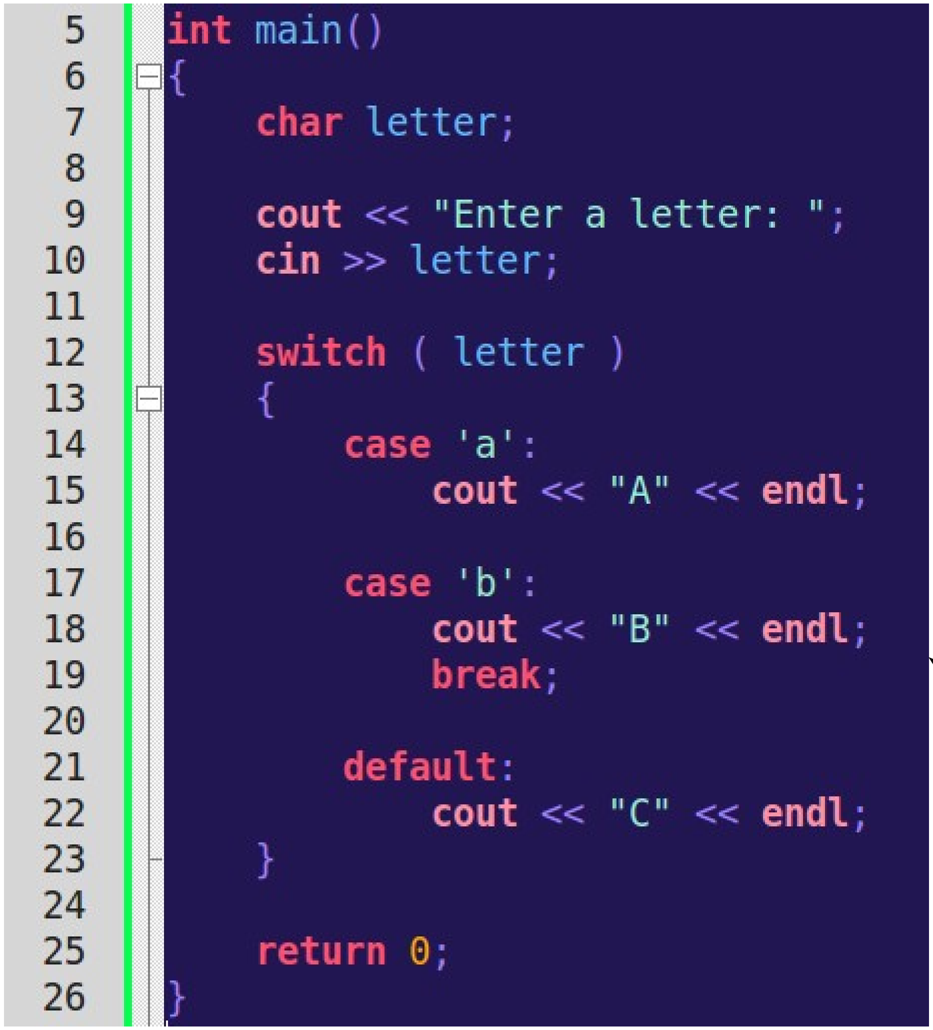
The **`default`** option is what is executed if none of the **`case`** statements match.



```
if ( choice == 1 )
{
    cout << "Deposit money" << endl;
}
else if ( choice == 2 )
{
    cout << "Withdraw money" << endl;
}
else
{
    cout << "Invalid command" << endl;
}
```

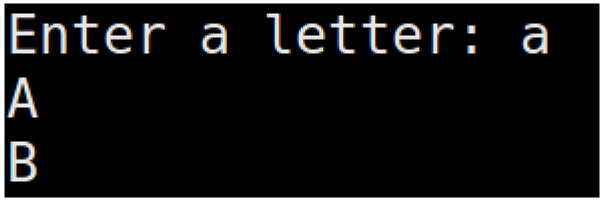
This would be similar to our **`else`** block.

Switches

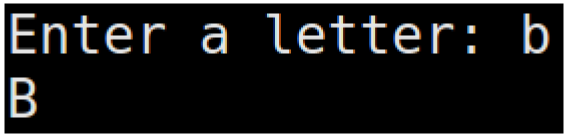


```cpp
int main()
{
    char letter;

    cout << "Enter a letter: ";
    cin >> letter;

    switch ( letter )
    {
        case 'a':
            cout << "A" << endl;

        case 'b':
            cout << "B" << endl;
            break;

        default:
            cout << "C" << endl;
    }

    return 0;
}
```

If we leave off the **`break;`** within a **`case`**, the code will “bleed through” into the next **`case`** block.



```
Enter a letter: a
A
B
```



```
Enter a letter: b
B
```

Time to code?