Conditional statements

Topic tagline (add picture if you want, but don’t move the text!)

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Contents

- Logic
  Relational operators, truth tables, logical operators
- If statements
- Switch statements
- Ternary operators
- Common errors
- Nested conditionals

Introduction

- So far the programs you've written have been quite linear
- Games are all about the choices a player makes – that's what creates the interactivity.
- This lecture will cover methods for creating branches within a program depending on some conditions.

Relational operators

These are operators that will compare data and return a true or false value

Operator	Function	Example
==	Equal to	lhs == rhs
!=	NOT equal to	lhs != rhs
<	Less Than	lhs < rhs
<=	Less than or equal to	lhs <= rhs
>	Greater Than	lhs > rhs
>=	Greater Than or equal to	lhs >= rhs

```
bool value = 5 < 7;
int a = 10;
value = a == 10;
```

Logical operators

Operator	Function	Example
!	Logical NOT	!conditional
&&	Logical AND	condition1 && condition2
||	Logical OR	condition1 || condition2

- The NOT operator negates the value of a boolean expression.
- The AND and OR operators combine relational expressions

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Truth tables

- So how does AND, OR and NOT work?

a	b	a && b	a || b	!(a || b)
false	false	false	false	True
false	true	false	true	false
true	false	false	true	false
true	true	true	true	false

Logical operators

AND means that both sides of the expression must be true for the whole expression to be true

```
bool isRaining = true, haveUmbrella = true;
bool canPutUpUmbrella = isRaining &&
haveUmbrella;
```

OR means that at least one side of the expression must be true

```
bool zombieApocalypse = true, fridayAfternoon = false;
bool goingToTheWinchester = zombieApocalypse ||
fridayAfternoon;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

If Statements

An if statement conditionally executes another block of code depending on the validity of that statement

```
bool condition = true;

if(condition)   //Line does not end with ;
{
    //Execute the code in here if the condition is true
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

More if

If the condition is not true, then the block of code within the scope of the if statement is not executed.

```
bool condition = false;

if(condition)   //Not true
{
    //This code will not execute
}
//The next line to execute continues from here
```

If continued

Any conditional test can be placed within the parentheses, provided it resolves to true or false.

Braces are optional when only a single line statement is executed.

- Though it is good practice to always have the braces there, especially while you are learning

What if it’s not true?

The *if* statement can be extended with the use of *else.* Any code contained within the *else* code block will be executed if the condition evaluates to false.

```
if( condition )
{
    //if condition is true!
}
else
{
    //if condition is not true!
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

What if it’s not true?!

- You can also extend an if/else block with else if statements. This allows you to check another condition if the first one is false.
- You can have as many else if statements following an if statement as you like.

```
if(condition1)
{
    //Execute the code in here if the condition is true
}
else if(condition2)
{
    //Execute the code in here if the first condition
    //is false and the second condition is true
}
else
{
    //Finally, execute this code if none
    //of the above conditions were true
}
```

Switch it up!

- Large sets of if/else statements can be confusing.
- C++ provides an alternate way to using if statements, called switch statements.

```
switch( value )
{
case 0:
    //statement One
    break;
case 1:
    //statement Two
    break;
//and so on... until
default:
    break;
}
```

Comparison with *if* statement
(for comparisons sake)

```
if( value == 0)
{
    //statement one
}
else if(value == 1)
{
    //statement two
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Introducing the Switch

- A switch statement compares the value in the () braces to each of the cases. If the value matches the value of a case then it will enter the corresponding code block.

- break; is used to exit a code block.

- Without a break statement, one case will carry into another

Defaults and you

- The default case is much the same as an else statement in an if-else block.
- It is executed if all other cases do not execute.
- While not mandatory, a default case is a good idea and good coding practice to boot.

```
switch( value )
{
case 0:
    //statement One
    break;
case 1:
    //statement Two
    break;
default:
    //if no cases executed
    break;
}
```

Ternary Operators?:?

Sometimes C++ and other programmers in general don't like white space in their code.

For easy, or very simple *if* statements we can use the *ternary* operator.

```
int result = 0;
//regular if/else evaluation
if( value1 > value2 )
{
   result = 24;
}
else
{
   result = 16;
}
```

```
//Ternary evaluation
//result = (condition)  ? True : False;
result = (value1 < value2) ? 24 : 16;
```

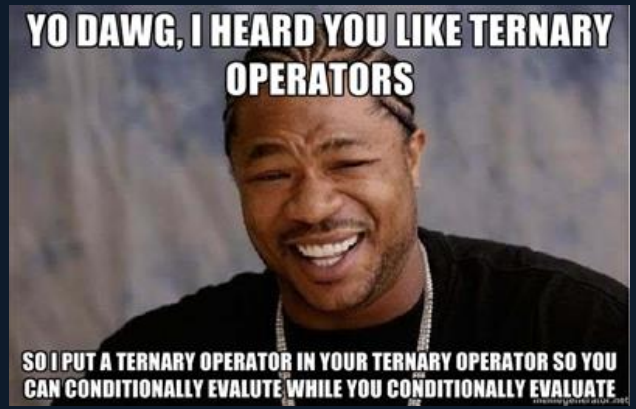
Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Nesting conditionals

Nesting is the process of placing conditionals inside one another

- For example, placing an if block inside another if block




```
if (4 < 5)
{
      if (1 == 0)
      {

      }
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Spot the errors

```
int age;
std::cin >> age;
if (age = 18)
{
     std::cout << "Hi adult";
}
```

```
int height;
std::cin >> height;
if (height < 120);
{
     std::cout << "You cannot ride this rollercoaster";
}
```

```
int red, blue;
std::cin >> red >> blue;
if (red && blue == 255)
{
     std::cout << "Colour is magenta";
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- Conditional statements are a way to alter the flow of your program and create branches and different options

- We can use if statements, switch statements or the ternary operator to do this. If statements are by far the most common.

- By cleverly using logical expressions we can cover some very complex circumstances, but this does take practice!

References

- Gaddis, T, 2012, *Starting Out with C++: From Control Structures Through Objects*, 7th edition, Pearson Education

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX