Hello C...TYI!

Domhnall O'Hanlon

February 14, 2017

Part 2: Arithmetic and Logic

Maths Operators

Also known as binary operators, these mathematical operators require two operands to produce an output. For example, 40 + 2 = 42 requires two inputs to produce an output. As we will see later, programming languages also support unary opeartors.

Operation	Symbol
addition	+
subtraction	-
multiplication	*
division	
tabularnewline modulo	%
equivalence	==

A simple incrementer

One of the most common tasks you will encounter in programming (in any language) is incrementing a value i.e. increasing it by a specific amount. Take for example scoring points in a game. Every time you hit a pig in Angry Birds 5000 points are added to your score.

Try the following code snippet:

```
//initialise a variable to store the number of times prinf is run
int numPrints = 0;

//increment the variable by one
numPrints = numPrints + 1;
  printf("The printf function has run %d times \n", numPrints);
```

```
numPrints = numPrints + 1;
  printf("The printf function has run %d times \n", numPrints);

numPrints = numPrints + 1;
  printf("The printf function has run %d times \n", numPrints);
```

i++

The most common amount to increment a variable by is one, in fact it's so common that there's a shorthand for it. All you have to do is write it in the form `myVariable++`, so the example above becomes:

```
int numPrints = 0;

numPrints++;
  printf("The printf function has run %d times \n", numPrints);

numPrints++;
  printf("The printf function has run %d times \n", numPrints);

numPrints++;
  printf("The printf function has run %d times \n", numPrints);
```

Unary Operators

Unlike binary operators, unary operators only require one operand. They allow you to write your code more concisely. For example, `i++` or `i--` will incrememt or decrement the integer `i`.

There are serveral other ways of concisely expressing arithmetic operations, but they are binary rather than unary. For example, if we want to implement the Angry Birds scoring system we could use `myScore+=5000` to increase the score in steps of 5000.

Table of Unary Operators

In the previous chapter we saw the **Address Of** unary operator when we learned about the `scanf()` function. We were able to scan information from the console and store it in a variable called myInt using the following line of code:

```
scanf("%d", &myInt);
```

The table below sets out the rest of the unary operators.

Operation	Symbol	Also
Increment	x++	++x
Decrement	x–	–x
Address	&x	
Pointer	`*x`	
Positive	`+x`	
Negative	-x	
Ones Complement	~x	
Logical negation	!x	
Variable Size	sizeof x	sizeof(type-name)
Type casting	(type-name)	

Performing Calculations in printf()

You can create functions to perform maths operations almost anywhere in your program. It is also worth knowing that you can perform calculations within other functions such as `printf()`. Try the following example:

```
printf("Pi is approximately %f", 22/7);
```

Working with Modulo

You should be familiar with the first four mathematical operations, but modulo may be new to you. Put simply, modulo tells you the remainder of dividing one number by another. For example `7%2` would return `1`, since 2 goes into 7 three time with a remainder of 1. Similarly `6%3` would return `0` since 3 divides evenly into six. In general terms, the modulo operator will always return a number between 0 and d-1, where d is the divisor (or denominator, if you prefer to think in fractions).

Order of Operations

Multiplication is just a concise way of saying that you want to repeatedly add a number to itself, and similarly division is just a simpler way of saying you

want to repeatedly subtract a number. Then when multiplication and division become too cumbersome to work with you can use exponents and radicals (roots) to indicate repeated multiplication and division, respectively. This heirarchy is the reason you had to memorise some acronym for the order of mathematical operations.

PEMDAS	**BOMDAS**	**BIMDAS**
Parentheses	Brackets	Brackets
Exponents	Orders	Indices
Multiplication	Multiplication	Multiplication
Division	Division	Division
Addition	Addition	Addition
Subtraction	Subtraction	Subtraction

Order of Operations

Unlike many calculators, C is intelligent enough to understand this order and will apply it when making calculations. This means that if you want some particular step of your calculation to happen in a certain order then you will have to make careful use of brackets.

Try the following snippet as an example and see if you can come up with some others.

```
printf(3 + 5 * 7); //returns 38
printf((3+5) * 7); //returns 56
```

Programming Challenges

Here are a few simple scenarios to challenge your understanding of the programming concepts we've covered so far.

Grade Point Average

Write a simple command line application that accepts 6 integer grades and returns the average of these numbers.

Tip Calculator

Write a command line application that accepts a bill amount and return to the user both the value of a 10% tip and the combined amount of initial bill and tip.

Logic

This section will help you add some "intelligence" or decision making abilities your programs. By the end of this section you will be able to write programs that respond to a variety of different input conditions, and we will conclude this chapter by writing a very simple game.

If Statements

Sometimes refered to as branches, if statements contain code that only executes if a certain condition is met. A nice example of an app that makes decisions based on specific events happening is called "IFTTT" which stands for **If This Then That**. As a more everyday example you can imagine the following scenario: "If it's raining outside then bring an umbrella"

Syntax

A typical if statment will look like so:

```
  if(test if true){
    code to run if test is true;
}
```

Try the following out:

```
int input;

  printf("What is the meaning of life, the universe and everything? \n")?
  scanf("%d", &input);

  if (input == 42){
    prinf("Such learning. Many wisdom. Wow");
}
```

What ELSE can we do?

In the previous example there was only one "correct" answer. When you run the program you only ever get a response if the number '42' is entered. For every other input the program remains silent.

By adding an else condition we can catch all the other alternatives that our if test misses.

```
 if(input == 42){
    printf("Such Learning Many Wisdom. Wow");
```

```
} else{
    printf("Try again");
}
```

Else If

Finally, we can run more than two tests by adding in one (or more) `else if` cases. For example:

```
 if(input == 42){
    printf("Such Learning Many Wisdom. Wow");
} else if(input == 43){
    printf("Close, but no cigar");
} else if(input == 41){
    printf("Close, but no cigar");
} else{
    printf("Try again");
}
```

Ternary Operator

As with most things in programming, there's a more concise way to write your if statements. We've already seen the a unary operator take one operand, a binary operator takes two and with a ternary operator we can use three operands to handle the "If-Then-Else" elements of an if statement.

```
(test) ? trueCode : falseCode;
```

The ternary operator, just like unary and binary operators, can be used as an argument in other functions such as `printf()`. Try this nice way to print plurals correctly!

```
int numFriends = 2;

printf("You have %d friend%s", numFriends, (numFriends!=1) ? "s." : "." );
```

Coding Challenge!

Sorting...Sort of.

Early entrance is changing - students will now be divided by surname, write a program that checks the first character of a lastName string, A-N, M-Z

Truth Tables

Hopefully you will remember truth tables from your electronics studies in EM113. Just in case you've forgotten, here's what the `OR` and `AND` truth tables look like for two inputs.

A OR B

A	B	A+B
0	0	0
1	0	1
0	1	1
1	1	1

A AND B

A	B	A.B
0	0	0
1	0	0
0	1	0
1	1	1

Logic Operators

In your logic tests you can test if more than one condition is met using a logic `AND` operator or you can test to see if either condition is true using the logic `OR` operator.

Here's some pseudo-code to illustrate:

```
  //logic and example
  if (test 1 && test 2){
    code if both are true;
}
```

Logic and is denoted by `&&` which is Shift + 7 on a UK keyboard. Logic or is denoted by `||` which is Shift + \ on a standard keyboard.

Coding Challenges

Try these two challenges to test your knowledge of what we covered in this chapter:

CAO Points Calculator

Write an application that asks for an integer input (between 0 and 100) and converts it the corresponding Leaving Cert grade.

Fizz Buzz

need to introduce for loops first

Print all the numbers from 0 to 30 inclusive. If the number is evenly divisibe by 3 print "FIZZ" instead of the number If it's evenly divisible by 5 then print "BUZZ" instead of that number. If the number is divisible by both 3 && 5 the print FIZZBUZZ

Summary

After this second chapter you should now be able to:

- Understand binary operators and use them in your programs,
- Recognise the unary operators, know where to use them and be able to use the Address Of unary operator,
- Write If-ElseIf-Else statements,
- Use ternary operators instead of IF/Else statements,
- Use logic AND in your programs,
- Use logic OR in your programs.

This page is intentionally left blank