ENGG1003 - PASS Session 3

Mitchell Deltoer

Monday	**14-15**	**ES238**
Wednesday	**12-13**	**ES238**
Thursday	**10-11**	**MCLG42**

FOR Loops

The general structure of a `for` loop in C is:

```
for( initialisation ; condition ; increment ){
   // some code a set amount of times
}
```

The `initialisation` expression is only done once before the first loop, the `condition` is tested before each loop iteration and the `increment` expression is executed after each loop iteration. The code block associated with the `for` loop will continually execute as long as the `condition` is true before each iteration.

1. Examine the following code listings and determine the output to the user. How many iterations are performed?

   (a)

   ```
   for(int i = 0; i < 10; i++){
      printf("%d\n",i);
   }
   ```

   (b)

   ```
   for(int i = 1; i <= 10; i++){
      printf("%d\n",2*i);
   }
   ```

BREAK and CONTINUE

Two more tools to include in flow control are the `break` and `continue` statements. When a `break` statement is encountered inside a loop, the loop is immediately terminated. In contrast, a `continue` statement forces the next iteration of the loop to take place immediately.

It may not be all that often that it is absolutely necessary that you need to include either one of these, but in some cases it can make your code simpler.

DO-WHILE Loops - Optional

A `do-while` loop in C is a subtle variation on the `while` loop structure. The general structure of a `do-while` loop in C is:

```
do{
   // some code here at least once
} while( condition );
```

The code block is always unconditionally executed first, and then before each subsequent loop iteration the `condition` is tested. Then the code block continually executes as long as the `condition` is true before each iteration.

It is basically the same as a `while` loop except it executes at least once. A `do-while` is never absolutely necessary, and you can code the same behaviour with just a `while` loop, but it can make the intent of your code easier to see if you understand the structure.

Practice Programming

A summary of some quick tips:

- `if ... else if ... else` statements are used to choose between several different options or cases.
- A `for` loop will execute a block of code a set number of times.
- A `while` loop can be used to execute a block of code either indefinitely, or it can be used like a `for` loop.

Task 1: Sum of consecutive numbers

Write a C program that will display the integers from 1 to `n` and their total sum, where `n` is the user's input.

Task 2: Display integers in reverse

Write a C program that will display all of the integers in reverse from `n` to zero, where `n` is the user's input.

Task 3: Add numbers indefinitely

Write a C program that will continuously add numbers that the user enters. When the user enters 0, the final sum and the average should be displayed.

Task 4: Prime numbers

A prime number is a whole number greater than 1 that is only divisible by 1 and itself.

1. Write a C program that will determine whether a given number is a prime number or not.

2. Write a C program that will print all prime numbers between 1 and `n`, where `n` is the user's input.

Task 5: Challenge: Exponential series expansion

The exponential function can be evaluated using the following series expansion:

$$\exp(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

where x is the input, k is the current term of the series being evaluated and ! denotes the *factorial* function.

1. Write a C program that will compute the *factorial* of the user's input.

2. Write a C program that will compute the series for any value x which the user inputs. The series should terminate and print the sum to the user when either:
   - The absolute value of the current term is less than 10^{-6}.
   - 50 iterations have been computed.