DT228-1 Micros Lab 1.

Key topics : Edit, compile, link, run, arithmetic operators, logical operators.

Introduction

The aim of this lab is to re-acquaint you with the edit/compile/link process and also to introduce you to a range of C operators. You will also investigate numerical ranges of different variable types and investigate number bases. Write the answers to the questions posed on a sheet of paper.

Part 1. Edit, compile, link: Hello World.

The aim of this exercise is to make sure you an operate the editor and the compiler

A detailed breakdown of this program can be found here:
http://eleceng.dit.ie/frank/IntroToC/Comp_Link.html

1) Open Notepad++
2) Type this in:

```
#include <stdio.h>
void main()
{
    printf("Hello World!");
}
```

3) Save this to your home drive or a thumb drive as **hello.c**
4) Go to the start menu, click Run, type in **cmd** and press enter.
5) Navigate to the directory where you saved hello.c
for example, lets say you saved hello.c in e:\myprog
To get there in a cmd window do this:

```
e:
cd \myprog
```

The first line changes you to the **e:** drive. The second line moves you into the **myprog** directory (folder).
6) Compile (and link) the program as follows:

```
gcc hello.c -o hello.exe
```

This creates a program called **hello.exe** (that's what the **-o** command line argument is all about).
7) If there is an error then go back to the editor, fix it, and re-run the compiler (the up-arrow will scroll through your command history)
8) Run the program by typing this in:

```
hello.exe
```

9) You should see “Hello World!”

Part 2. Variables and ranges.

The aim of this exercise is to make sure you understand the differences between variable types and their numeric ranges.

1) Type (or copy/paste) the following in Notepad++

```
#include <stdio.h>
void main()
{
      char MyVariable;  // Line 3
      MyVariable = 29;  // Line 4
      MyVariable = MyVariable+1;
      printf("The decimal value of MyVariable is : %u\n",MyVariable);
      printf("The hexadecimal value of MyVariable is : %X\n",MyVariable);
}
```

2) Save this to the same location as before but call it **var.c**
3) Compile and link as follows:

```
gcc var.c -o var.exe
```

4) Run the program and figure out why the lines displayed are different.
You should see this:

```
The decimal value of MyVariable is : 30
The hexadecimal value of MyVariable is : 1E
```

5) Change Line 4 so that it now reads:

```
MyVariable = 255
```

6) Compile and run the program again. What's happened?

7) Repeat the above but change the data type of your variable on Line 3 to **short**, **int** and **long** in turn. What range of values can these numbers types represent? How many bits are in each type?

8) The **printf** statements make use of formatting characters (%X,%u etc). What other formatting types are there for the printf function?

Part 3 Arithmetic and logical operators.

The aim of this exercise is to make sure you understand how C performs arithmetic and bitwise operations.

You have probably come across the C language set of arithmetic operators (*,/,+,-) before. You may not have come across the more unusual arithmetic and logical operators however.

Procedure
1) Enter the following program into Notepad++, save, compile and execute. Note the output.

```
int main(void) {
        printf("5 + 4 = %d\n", 5 + 4);
        printf("5 - 4 = %d\n", 5 - 4);
        printf("5 * 4 = %d\n", 5 * 4);
        printf("(float)5 / (float)4 = %f\n",(float)5 / (float)4);
        printf("5 / 4 = %d\n",5 / 4);
        printf("5 % 4 = %d\n",5 % 4);
        printf("5 & 4 = %d\n",5 & 4);
        printf("5 | 4 = %d\n",5 | 4);
        printf("5 ^ 4 = %d\n",5 ^ 4);
        printf("~5 = (hex) %x\n", ~5);
}
```

2) Name each of the operators
3) Experiment with values other than 4 and 5 to ensure that you understand exactly how each of the operators behave.

4) What is the difference between the following statements:

```
      if (a & b)
```
AND
```
      if (a && B)
```