Week 1 – SOFT7019 lab session

This week we will utilise an online C IDE called online gdb, please access it at <https://www.onlinegdb.com/>

In the top right corner, you will have the option to select the programming language, please select C.



# Exercise 1

*C library function utilised:* int printf(char\* , …....)

1. Delete the default text in the main.c file. In the main function declare two variables that are initialized and print them. The default program prints out Hello World using a function called printf . This function is part of the stdio library, so there is #include line at the top of the file to #include <stdio.h> file. This is system file. This contains a declaration of the function printf and will make sure our program links to stdio.lib/ stdio.so depending on which OS we are running.

We use ‘<’ ‘>’ around the name of the .h file as this is a system .h file.

int number = 31;

char letter = 'A';

In order to print these parameters we will use printf

printf("Printing Variables");

printf("\n number is\t %d : %x\n", number, number);

printf("\n letter is\t %c : %x\n", letter, letter);

The first parameter is a string containing **conversion specifiers** characters. ie %d or %c or %x

For each conversion specifier a variable name is included as a parameter. We just happen to be printing the same variable twice in different ways.

Mostly we will use %d for int types and %c for char types . It is also useful to print the values in hexadecimal(base 16) using %x. This is how they would appear in a memory dump in the debugger.

In this example we print the value of the variable twice but using two different conversion specifiers.

Build and run the program. The output should look like this:

**num is 31 : 1f**

**letter is A : 41**

**To do:** Modify the second printf so that we can also see the number (ascii code) used to represent the letter 'A' . We use the %d formatting character for this.

printf("\n letter is\t %c : %d: %x\n", letter, letter, letter);

The output is now:

**num is 31 : 1f**

**letter is A : 65 : 41**

# Exercise 2

*C library function utilised:* int sizeof(variable)

Add to main.c

Every time we declare a variable we are allocating some memory. The amount of memory allocated depends on the type of the variable declared. The code below shows the amount of memory used by the variable number which is of type int. This uses the special operator sizeof.

**To do:** Add a line to print out the amount of space used by letter which is of type char.

printf("Amount of memory used");

printf("\n No. of bytes for a int is\t %d : \n", sizeof(number));

# Exercise 3

1. Print all the numbers from 1 to 16 in a row using the following loop.

int start = 0;

int i = 0;

for (i = start; i <= 16 ; i++ )

{

printf("%d %x", i, i );

}

1. Change the target in the for loop from 16 to 255 which is the biggest number 1 byte can hold or FF in hex. Here you can see that the value held in one byte can be represented by two symbols in hex going from 0 to FF.

Modify this to get a 16 \* 16 square of numbers .

1 2 3 4 5 6 7 8 9 A B C D E F

10 ………………………………………………………………1F

….

….

…..

F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

We will see that if we are looking at what is contained in memory hex is used to show the contents of one byte.

# Appendix

The ASCII character encoding table.

