Week 2 – 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

**Getting characters from the keyboard and putting chars to the screen**

1. To get a single keystroke from the user we use getchar (the return is of type of getchar is int. ). In reality it returns an unsigned char cast to an int or EOF if there is an error. (See man pages for C library on linux <https://linux.die.net/man/3/getchar> )

Try out the following code snippets:

int keyinput;

keyinput = getchar();

getchar(); // enter key read from input stream

To run this code ask the user to enter a character using the keyboard and then press *Enter*. The second getchar()gets the *Enter* keystroke so we don't need to save the return value from this call to getchar(). We can print out the letter enter in ASCII by printing letter as using the formatting character %d or as a letter using %c

printf("Letter entered %d\n", keyinput);

printf("Letter entered %c\n", keyinput);

More appropriately can store the return value of getchar to a variable of type char.

char letter = (char) getchar();

getchar();

We can print the value stored in letter either as a ASCII number or a letter using the correct format specifier.

printf("Letter entered %d\n", letter);

printf("Letter entered %c\n", letter);

1. To print a single character to the screen use putchar

(See man pages for C library on linux <https://linux.die.net/man/3/putchar> )

char letterX = ‘X’;

putchar(letter);

1. Write a program that gets two initials from the user and print out their initials”

printf("\nEnter your first initial followed by enter key\n");

// Add code to store this letter in firstLetter and read enter key

// add another printf to ask for the second initial

// Add code to store this letter in secondLetter and read enter key

// Store the dot symbol in a char variable called dot

// Use 4 calls to putchar to print the user’s initials separated //by dots Add code to print out

**Sample Output**

**Please Enter your first initial:**

H

**Please Enter your second initial:**

F

**H.F.**

1. Increment both letters and call printf again. We can do this because characters are stored as numbers. (1 byte is used to store the ascii code for the character)

firstLetter = firstLetter +1;

secondletter = secondLetter +1;

The output should have changed by one letter:

**I.G.**

This serves to remind us that letters are being stored as numbers!

1. How would you get the same output using the printf function with four variables following the first parameter which is the formatting string,

**Hint:**

printf(“Initials are %c %c “ , firstletter, dot, ….);

# Exercise 2

**Null Terminated Character Arrays- Strings in C**

Most of the time when writing a program, we want to get strings and print strings on the screen and not single characters. To store the initials "R.D." as a character array we would need an array to hold 5 elements.

One for each of the characters 'R', '.', 'D' and '.' and a 0 to terminate. We could construct this character array like this:

1. Copy the following code to create a character array.

char initials[5]; // allocates a series of 5 bytes

initials[0] = ’R’;

initials[1] = '.';

initials[2] = ‘D’;

initials[3] = '.';

initials[4] = '\0';

1. To print the character array initials you must use printf with the formatting character %s.

printf("\nInitials are %s \n", initials );

1. Let’s add another character array string called word. This time we'll forget to put the \0 at the end.

char word[4];

word[0] = 'c';

word[1] = 'a';

word[2] = 't'

word[3] = 's';

Add a printf to print word. What happens when you execute this code? How should you fix it?

The correct code would allocate another byte when declaring word so we can include a null terminator:

char word[5];

word[4] = '\0';

1. Now change word so that it is only an array of 2 characters:

char word[2];

word[0] = 'c';

word[1] = 'a';

word[2] = 't'

word[3] = 's';

Recompile and run. It may work fine even though there is a bug in the code. Whats the bug? Does the compiler find this bug?

1. We can create and intialise string/ character arrays on one line. For the following array 6 bytes will be allocated

char name[] = “Robin”;

printf(“Bytes allocated %d”, sizeof(name));

Create a character array/string for your name and for your surname. Print them out on one line using printf using the %s conversion specifier for each character array/string variable.

# Appendix

The ASCII character encoding table.

