Week 4 – 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

**Storing a memory address**

We’ve seen that every variable is stored somewhere in memory. We can find out the location / address of any variable using the **&** known as the **address of** operator. To find out the number of bytes used at this location we use the **sizeof()** operator.

We’ve seen that a variable location (address) looks something like this: 0x 00 28 FF 1C. If we want a variable to remember this – what size would/ how much space would this variable need?

To store the address itself in memory it will use 4 bytes of memory (this may vary depending on the system you are running). 4 bytes (32 bits) may be used to identify any byte in a 4 GB chunk of memory.

C supports a data type that stores addresses:

* int\* is used as the type to store an address of an integer
* char\* is used to store the address of a char

Here is the code to store the address of num1 and then to print out that address.

**int\*** pAddress1= NULL;

pAddress1= &num1;

printf("\n The location of num1 is at 0x%p\n", pAddress1);

Create another address variable pAddress2 to store the address of num2 and print this out.

Create an address variable pAddress3 to hold the address of one of a char variable that you initialise and print this out.

char\* pAddress3 = NULL;

What do you think is the different between a pointer to a char and a pointer to an int?

# Exercise 2

**Using pointers to access and print information in memory**

Our intention is to store an address of an integer in memory and use this address later to access the integer at that address. This way we can easily use 'follow ' or dereference the address/pointer to print the number stored in num1. Because it is an address of an integer when we dereference it the compiler knows that it should read 4 bytes (all ints use 4 bytes). If it was a char\* address then it would only read 1 byte.

Note to follow an address/dereference an address we use the \* symbol immediately in front of the address name.

* Print the address of num1 and the value of num1 using the pointer.

printf("\n\nInt Value at address %p is %d", pAddress1, **\***pAddress1);

* Print the address of num2 and the value of num2 using the pointer pAddress2.
* Print the address of the char and the value of the char using the pointer pAddress3.

# Exercise 3

**User input using scanf()**

In C programming, scanf() is one of the commonly used function to take input from the user. The scanf() function reads formatted input from the standard input such as keystrokes. The value that is read is saved at a memory address designated by the user.

Example:

#include <stdio.h>

int main()

{

int testInteger;

printf("Enter an integer: ");

scanf("%d", &testInteger);

printf("Number = %d",testInteger);

return 0;

}

Write a program using scanf() to read a user’s phone number (you can make up a fake number). What advantage do you see for using scanf() over getchar()?

Write a program using scanf() to read a user’s name.