NANYANG POLYTECHNIC
EGR204 Microcontroller Applications

E-Laboratory Session A

Course:	Diploma in Robotics and Mechatronics
Module:	EGR204 Microcontroller Applications
Experiment:	E-Lab A
Title:	Basic C programming I

Objective:

- Create a basic structure for C program using interactive computer programming environment
- Demonstrate understanding of basic operators and data types in developing a C program

Learning Objectives:

- Understand basic structure of a simple C program.
- Understand basic C data types.
- Basic operators.

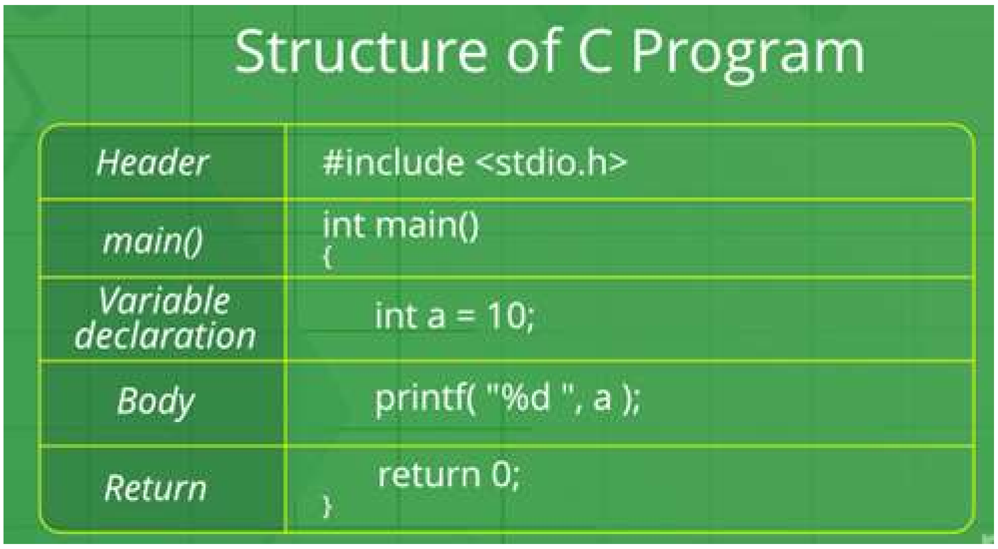
1. Introduction

C is a procedural programming language. It was initially developed in the year 1972. It was mainly developed as a system programming language to write an operating system. The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programming like an operating system or compiler development.

Many later languages have borrowed syntax/features directly or indirectly from C language. Like syntax of Java, PHP, JavaScript, and many other languages are mainly based on C language. C++ is nearly a superset of C language.

2. Structure of a C program

Structure of C program:



Structure of C Program

Header	`#include <stdio.h>`

main()	`int main()`
	`{`

Variable declaration	`int a = 10;`
Body	`printf( "%d ", a );`

Return	`return 0;`
	`}`

The components of the above structure are:

<u>Header Files Inclusion</u>: The first and foremost component is the inclusion of the Header files in a C program. A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

stddef.h – Defines several useful types and macros.
stdint.h – Defines exact width integer types.
stdio.h – Defines core input and output functions
stdlib.h – Defines numeric conversion functions, pseudo-random network generator, memory allocation
string.h – Defines string handling functions
math.h – Defines common mathematical functions

Syntax to include a header file in C:

```
#include
```

<u>Main Method Declaration</u>: The next part of a C program is to declare the main() function. The syntax to declare the main function is:

Syntax to Declare main method:

```
int main()

{

  // your code goes here

}
```

<u>Variable Declaration</u>: The next part of any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in the C program, variables must be declared before it can be used. In a C program, the variables are to be declared before any operation in the function.

Example:

```
int main()

{

    int a;    // integer variable named 'a' is declared
.
.
```

<u>Body</u>: Body of a function in C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

Example:

```
int main()

{

    int a;


    printf("%d", a);   // print value of variable 'a' in terminal
.
.
```

<u>Return Statement</u>: The last part in any C program is the return statement. The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

Example:

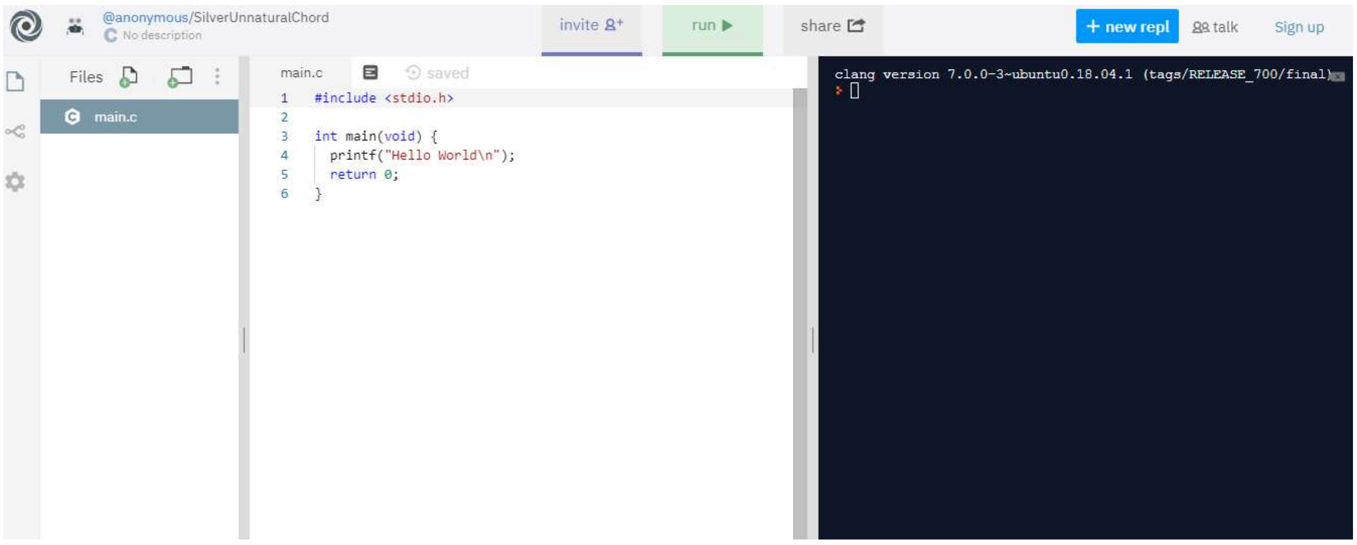
```
int main()
{
    int a;

    printf("%d", a);

    return 0;              // the main() returns 0
}
```

3 “Hello World” C program

Next, we are ready to write a simple C program. Go to www.repl.it, click “start coding” button and select C program. Click “Create Repl”, a web-based C editor and compiler will be shown. Notice that the cod editing window, output window and file window.



```
#include <stdio.h>

int main(void) {
  printf("Hello World\n");
  return 0;
}
```

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
> 
```

Type in the following code and click run. Write down the output of the C program from the output terminal window (black window on the right)

```
#include <stdio.h>

int main(void) {
  printf("Hello World\n");
  return 0;
}
```

Fill in the output here: ____

Task A: Modify the C program such that the program gives the following output:

```
Welcome to EGR204
This is 2020 S1
Lab session 1: Basic C
My admin number is: "fill in your admin number here"
```

4. Basic C data types

The following table provides the details of standard integer types with their storage sizes and value ranges:

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

Type in the following code and observe the output, is there any difference from section 3 Task A?

```c
#include <stdio.h>

int main(void) {
  int year;
  year = 2020;
  printf("Welcome to EGR204\n");
  printf("This is %d S1\n", year);
  printf("Lab session 1: Basic C\n");
  printf("My admin number is: 190000B\n");
  return 0;
}
```

Task B: change this line:

`int year;`
to
`unsigned char year;`
What is the new output, and why?

____

____

5. Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators −

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Arithmetic Operators

The following table shows all the arithmetic operators supported by the C language. Assume variable **A** holds 10 and variable **B** holds 20 then −

Show Examples

Operator	Description	Example

+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Type the following C code in repl editor and click run, write down the outputs:

```
#include <stdio.h>

int main(void) {
  int x;
  x = 1;
  x++;
  printf("1) x = %d\n", x);
  x--;
  printf("2) x = %d\n", x);
  x= x+99;
  printf("3) x = %d\n", x);
  x=x/10;
  printf("4) x = %d\n", x);
  x=x%3;
  printf("5) x = %d\n", x);

  return 0;
}
```

Output is:

____

____

____

____

____

Bitwise Operators

Assume variable 'A' holds 60 and variable 'B' holds 13, A = 0011 1100, B = 0000 1101

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

Write down the output of following operation, then type the following C code in repl editor
To verify.

A = 1101 0011;
B = 0101 1010;

A&B = ________________

A|B = ________________

A^B = ________________

~A = _________________

A << 3 = ______________

B >> 5 = ______________

```c
#include <stdio.h>

int main(void) {
  unsigned int A, B;
  A = 0b11010011;
  B = 0b01011010;
  printf("1) A&B = %d\n", A&B);
  printf("2) A|B = %d\n", A|B);
  printf("3) A^B = %d\n", A^B);
  printf("4) ~A = %d\n", (unsigned char)~A);
  printf("5) A<<3 = %d\n", A<<3);
  printf("6) B>>5 = %d\n", B>>5);
  return 0;
}
```

Relational Operators

The following table shows all the relational operators supported by C. Assume variable **A** holds 10 and variable **B** holds 20:

True: 1

False: 0

Show Examples

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.

>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Type in the following code and observe the output:

```
#include <stdio.h>

int main(void) {
  unsigned int A, B;
  A = 10;
  B = 20;
  printf("1) A==B is %d\n", A==B);
  printf("2) A!=B is %d\n", A!=B);
  printf("3) A>B is %d\n", A>B);
  printf("4) A<B is %d\n", A<B);
  printf("5) A>=B is %d\n", A>=B);
  printf("6) A<=B is %d\n", A<=B);
  return 0;
}
```

Output is:

____

____

____

____

____

____

6. Assignment submission

Submission details: submit your source file as "E_Lab_A_assignment_(your admin number).c" at Blackboard Assignment "E-Lab A Assignment" Link.

```
#include <stdio.h>

int main(void) {
  _______________________    // declare an unsigned int variable named X

  _______________________    // initialize X to 20

  _______________________    // increment x by 10

  _______________________    // print new value of x

  _______________________    // shift x to left by 2 bits

  _______________________    // print new value of x

  _______________________    // print if x is equal to 20 as true(1) or false(0)

  _______________________    // print if x is greater than 20 as true or false

  return 0;
}
```