Birla Institute of Technology & Science, Pilani, Hyderabad Campus
First Semester 2020-2021

Computer Programming [CS F111] Lab 5

I. Local vs global variable.

Variables that are declared inside a function or block are called local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. The following example shows how local variables are used. Here all the variables a, b, and c are local to main() function.

```
#include <stdio.h>
int main () {
  /* local variable declaration */
  int a, b;
  int c;
   /* actual initialization */
  a = 10;
  b = 20;
  c = a + b;
   printf ("value of a = %d, b = %d and c = %d\n", a, b, c);
   return 0;
}
```

Output: value of a = 10, b = 20 and c = 30

Global variables are defined outside a function, usually on top of the program. Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. The following program show how global variables are used in a program.

```
#include <stdio.h>

/* global variable declaration */

int g;

int main () {

  /* local variable declaration */
```

```
  int a, b;
  /* actual initialization */
  a = 10;
  b = 20;
  g = a + b;
  printf ("value of a = %d, b = %d and g = %d\n", a, b, g);
  return 0;
}
```

Output: value of a = 10, b = 20 and g = 30

A program can have same name for local and global variables but the value of local variable inside a function will take preference. Here is an example −

```
#include <stdio.h>
 /* global variable declaration */
int g = 20;
int main () {
  /* local variable declaration */
  int g = 10;
 printf ("value of g = %d\n",  g);
  return 0;
}
```

Output: value of g = 10

II. Enum data-type

Enumeration is a user-defined datatype in C language. It is used to assign names to the integral constants which make a program easy to read and maintain. The keyword “enum” is used to declare an enumeration.

Here is the syntax of enum in C language,

```
enum enum_name{const1, const2, ..... };
```

Eg.: enum flag {const1, const2, ..., constN};

By default, const1 is 0, const2 is 1 and so on. You can change default values of enum elements during declaration (if necessary).

```
// Changing default values of enum constants
enum suit {
   club = 0,
   diamonds = 10,
   hearts = 20,
   spades = 3,
}
```

There are two ways to define the variables of enum type as follows.

```
enum week{sunday, monday, tuesday, wednesday, thursday, friday, saturday};
enum week day;
```

Here is an example of enum in C language.

```
#include<stdio.h>

enum week{Mon=10, Tue, Wed, Thur, Fri=10, Sat=16, Sun};

enum day{Mond, Tues, Wedn, Thurs, Frid=18, Satu=11, Sund};

int main() {

   printf("The value of enum week: %d\t%d\t%d\t%d\t%d\t%d\t%d\n\n",Mon , Tue, Wed, Thur,

Fri, Sat, Sun);

   printf("The default value of enum day: %d\t%d\t%d\t%d\t%d\t%d\t%d",Mond , Tues, Wedn,

Thurs, Frid, Satu, Sund);

   return 0;

}
```

Output: The value of enum week: 10 11 12 13 10 16 17
The default value of enum day: 0 1 2 3 18 11 12

III. For loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax
The syntax of a for loop in C programming language is −

```
for ( init; condition; increment ) {
   statement(s);
}
```

Here is the flow of control in a 'for' loop −

- The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

```
#include <stdio.h>

int main () {

   int a;

   /* for loop execution */
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }

   return 0;
}
```

Output: value of a: 10
value of a: 11
value of a: 12

value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19

Exercises:

1. Write a program to print prime numbers between 2 and n using *for loop*.

2. Write a program to print the following pyramid for a user given positive N. All the below sample outputs are for N=4

a)
```
1
1 2
1 2 3
1 2 3 4
```

b)
```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
  1 2 3 2 1
    1 2 1
      1
```

c)
```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

3. Write a program to find whether a positive integer entered by the user is a palindrome or not. E.g. 12321 is a palindrome whereas 112233 is not.

NOTE: Upload the screenshots of the **Exercise programs** along with the displayed results into your corresponding Google Classroom.

PATH to Submit the Screenshots:
Google Classroom --> Classwork --> View Assignment --> Create/Upload