

CC-112L

Programming Fundamentals

Laboratory 04

Modular Program Development – I

Version: 1.0.0

Release Date: 08-08-2022

Department of Information Technology

University of the Punjab

Lahore, Pakistan

Contents:

- Learning Objectives
- Required Resources
- General Instructions
- Background and Overview
 - **break** Statement
 - **continue** Statement
 - Logical Operators
 - Assignment Operators
 - Functions in C
 - Microsoft ® Visual Studio
- Activities
 - Pre-Lab Activity
 - **break** Statement
 - **continue** Statement
 - Task 01: Even Numbers
 - In-Lab Activity
 - Logical Operators in C
 - Assignment Operators
 - **if** Statement
 - Functions in C
 - Defining a function
 - Function Declaration/Prototype
 - Calling a Function
 - Math Library Functions
 - Task 01: Find Second Largest Number
 - Task 02: Square Asterisks
 - Task 03: Fill in the blank area
 - Task 04: Dry Run Program
 - Task 05: Find and Resolve Errors
 - Post-Lab Activity
 - Task 01: Reverse Digits
 - Task 02: Fill in the blank area
- Submissions
- Evaluations Metric
- References and Additional Material
- Lab Time and Activity Simulation Log

Learning Objectives:

- Using **break** Statement
- Using **continue** Statement
- Using Logical Operators
- Using Assignment Operators
- Functions in C
- Using Math Library Functions
- Defining a Function in C
- Function Prototypes in C

Resources Required:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022

General Instructions:

- In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how is s/he doing, this may result in negative marking. You can **ONLY** discuss with your Teaching Assistants (TAs) or Lab Instructor.
- Your TAs will be available in the Lab for your help. Alternatively, you can send your queries via email to one of the followings.

Course Instructors:		
Teacher	Prof. Dr. Syed Waqar ul Qounain	swjaffry@pucit.edu.pk
Teacher Assistants	Usman Ali	bitf19m007@pucit.edu.pk
	Saad Rahman	bsef19m021@pucit.edu.pk

Background and Overview:

break Statement: break Statement terminates the loop execution. When it is encountered in a loop, the loop immediately terminates. It is also used to terminate a switch case statement.

continue Statement: The continue Statement works like the break Statement but instead of terminating the loop it only skips one iteration of a loop e.g., it forces the next iteration by skipping any code in between.

Logical Operators: Operators which are used to perform logical operations of given expressions or variables.

Assignment Operators: Assignment Operators are used to assign a value to a variable. The left-side operand of the assignment operator is a variable and right-side operand of the assignment operator is a value. The value on the right side must be of the same data-type of the variable on the left side.

Function in C: A named block of code which performs a specific task. Every C program has at least one function of **main()**. You can pass data (parameters) to a function. A function is useful in reusing a code, which means to write a code once and use it multiple times.

Microsoft ® Visual Studio: It is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Microsoft ® Visual Studio supports 36 different programming languages and allows the code editor and debugger to support nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic, .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past.


© https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Activities:

Pre-Lab Activities:

break Statement:

The **break** statement terminates a loop at a certain point. Its execution is explained by the following code.



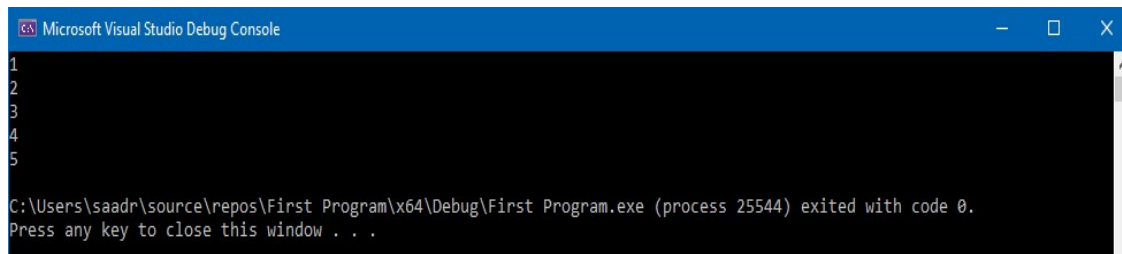
```
1  #include <stdio.h>
2  int main()
3  {
4      for (int i=1; i<=10; i++)
5      {
6          if (i == 6)
7          {
8              break;
9          }
10         printf("%d\n", i);
11     }
12 }
13
14 }
```

Fig. 01 (break Statement)

If we ignore the **break** Statement in the above code then it displays number from 1 to 10 on the console. The execution with the **break** statement is as follows:

- Loop executes each iteration and check if variable “i” is equal to value “6”
- Then the program displays the value of “i” on Console
- On sixth iteration, when the **if** condition becomes true, the program moves to the line 7
- As the **break** statement on line 8 is executed the program terminates the loop and shifts to the line 12

Following is the output of the above program



```
1
2
3
4
5
C:\Users\saadr\source\repos\First Program\x64\Debug\First Program.exe (process 25544) exited with code 0.
Press any key to close this window . . .
```

Fig. 02 (break Statement)

continue Statement:

The **continue** statement skips one iteration of a loop at a certain point. Its execution is explained by the following code.

```

1  #include <stdio.h>
2  int main()
3  {
4      for (int i=1; i<=10; i++)
5      {
6          if (i == 6)
7          {
8              continue;
9          }
10         printf("%d\n", i);
11     }
12 }

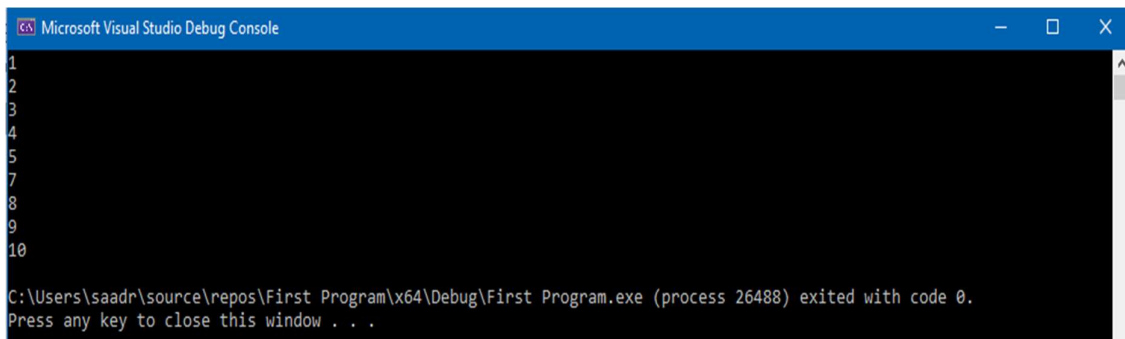
```

Fig. 03 (continue Statement)

If we ignore the **continue** statement in the above code then it displays number from 1 to 10 on the console. The execution with the break Statement is as follows:

- Loop executes each iteration and check if variable “i” is equal to value “6”
- Then the program displays the value of “i” on Console
- On sixth iteration, when the **if** condition becomes true, the program moves to the line 7
- As the **continue** statement on line 8 is executed the program skips an iteration and shifts to the line 4 i.e., “6” is not displayed on the Console

Following is the output of the above program



```

1
2
3
4
5
7
8
9
10
C:\Users\saadr\source\repos\First Program\x64\Debug\First Program.exe (process 26488) exited with code 0.
Press any key to close this window . . .

```

Fig. 04 (continue Statement)

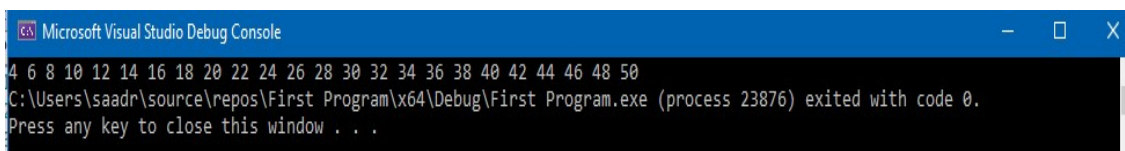
Task 01: Even Numbers

[Estimated 20 minutes / 20 marks]

Create a C program which:

- Iterates a loop from 0 to 100
- Displays Even Numbers between 0 to 50 using **break** statement
- Skips the second and third iteration using **continue** statement

The output should be as follows:



```

4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
C:\Users\saadr\source\repos\First Program\x64\Debug\First Program.exe (process 23876) exited with code 0.
Press any key to close this window . . .

```

Fig. 05 (Pre-Lab Task)

- Submit “.c” file named your “Roll No” on Google Classroom

In-Lab Activities:**Logical Operators in C:**

Operation	Logical Operator	Description
AND	&&	It returns true when both conditions are true. (x > 1 && y > 1)
OR		It returns true when at-least one condition is true. (x > 1 y > 1)
NOT	!	It reverses the state of the operand. e.g., if (y > 1) is true, then it returns false

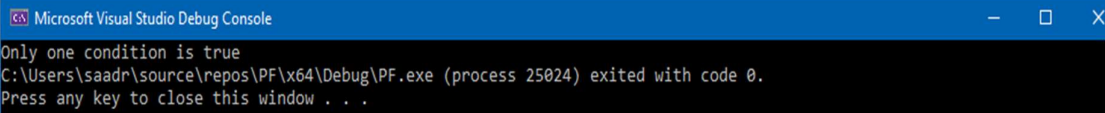
Following is an example of Logical Operators

```

1  #include <stdio.h>
2  int main()
3  {
4      int a = 20;
5      int b = 10;
6
7      if (a > b && b == 0){
8          printf("Both conditions are true");
9      }
10     else if (a > b || b == 0){
11         printf("Only one condition is true");
12     }
13 }
```

Fig. 06 (Logical Operators)

- On line 7, the program checks if both conditions are true. As b is not equal to zero, the program shifts to line 10
- On line 10, the program checks if any condition is true. As a is greater than b i.e., a condition becomes true. The program will display **“Only one condition is true”**



```

Microsoft Visual Studio Debug Console
Only one condition is true
C:\Users\saadr\source\repos\PF\x64\Debug\PF.exe (process 25024) exited with code 0.
Press any key to close this window . . .
```

Fig. 07 (Logical Operators)

Now, we add a **“NOT(!)”** operator with second condition.

```

1  #include <stdio.h>
2  int main()
3  {
4      int a = 20;
5      int b = 10;
6
7      if (a > b && !(b == 0)){
8          printf("Both conditions are true");
9      }
10     else if (a > b || !(b == 0)){
11         printf("Only one condition is true");
12     }
13 }
```

Fig. 08 (Logical Operators)

Now on line 7 the program checks if both conditions are true. As b is not equal to zero, second condition is false but the “!” operator makes it true. So, the program will display “**Both conditions are true**”

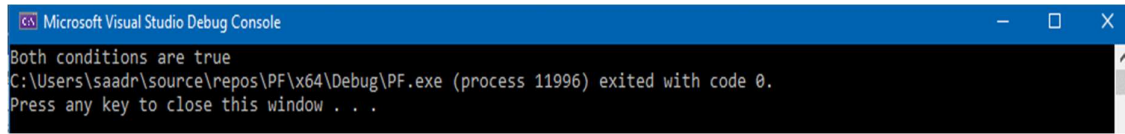


Fig. 09 (Logical Operators)

Assignment Operators:

Assignment Operators assign value to a variable of same type. Assume int a = 1, b = 2, c = 3, d = 4, e = 5.

Assignment Operator	Expression	Explanation	Assigned value
+=	a += 5	a = a + 5	Assigns 6 to a
-=	b -= 1	b = b - 1	Assigns 1 to b
*=	c *= 3	c = c * 3	Assigns 9 to c
/=	d /= 2	d = d / 2	Assigns 2 to d
%=	e %= 3	e = e % 3	Assigns 2 to e

Functions in C:

The function in C language is a named group of statements that together perform a task. Every C program has a function named **main()** and a programmer can define additional functions to reuse a group of statements.

Defining a Function:

A function in C programming is generally defined as follows:

return-type function-name (parameter list)

```
{
    body of function
}
```

- **Return type:** A function may return a value. The return-type is the data type of the value which the function returns. Some functions perform a specific task without returning a value. In that case, the return-type is the keyword “**void**”
- **Function Name:** This is the actual name of the function. The function name and the parameter list together constitute the “**function signature**”
- **Parameters:** A parameter is like a placeholder. When a function is called/invoked, you pass a value to the parameter. This value is referred to as parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are not compulsory, a function may contain no parameters
- **Function Body:** Function body contains the set of statements which defines what the function will do

Following is the source code for a function called “**minimum**”. The function takes two parameters num1 and num2 and returns the minimum between the two.

```

1
2 int minimum(int num1, int num2)
3 {
4     int min;
5     if (num1 < num2)
6     {
7         min = num1;
8     }
9     else
10    {
11        min = num2;
12    }
13    return min;
14 }

```

Fig. 10 (Function in C)

Function Declaration/Prototype:

Function prototype is widely known as Function declaration. Function declaration tells the compiler about the function name and how to call a function. The body of the function can be defined separately. A function declaration has following parts:

return-type function-name (parameter list);

For the function above in Fig. 10, the function declaration is: **int minimum(int num1, int num2);**

Parameter names are not important in function declaration, only their type is required. So, the above function declaration can be written as: **int minimum(int, int);**

Calling a Function:

While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function. When a program calls a function, program control is transferred to the called function. A called function performs defined task, and when its return statement is executed or when the function ends it returns program control back to the main program.

```

1 #include<stdio.h>
2 //Function Declaration
3 int minimum(int num1, int num2);
4 int main()
5 {
6     int a = 100;
7     int b = 50;
8     int result;
9     //Calling a function
10    result = minimum(a, b);
11    printf("Minimum number is: %d", result);
12 }
13 //Function definition
14 int minimum(int num1, int num2)
15 {
16     int min;
17     if (num1 < num2)
18     {
19         min = num1;
20     }
21     else
22     {
23         min = num2;
24     }
25     return min;
26 }

```

Fig. 11 (Function in C)

The execution of above program is as follows:

- Execution starts from the main()
- At line 6 & 7, two integers are defined
- At line 8 an integer is declared
- At line 10, function is called and its value is assigned to an integer
- As the function is called program control shifts to line 14
- Specified task is performed from line 14 to 21
- After execution of line 21, program control shifts to line 11
- At line 11, message is displayed on the console

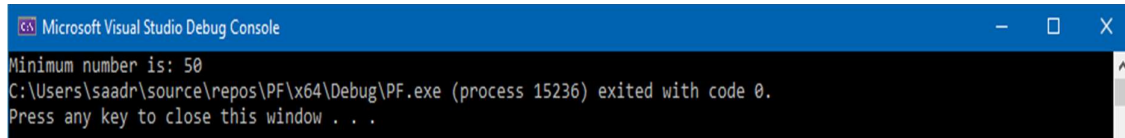


Fig. 12 (Function in C)

Math Library Functions:

C's math library functions allow you to perform common mathematical calculations. To use math library function you have to include a header "**math.h**" as follows:

```
#include <math.h>
```

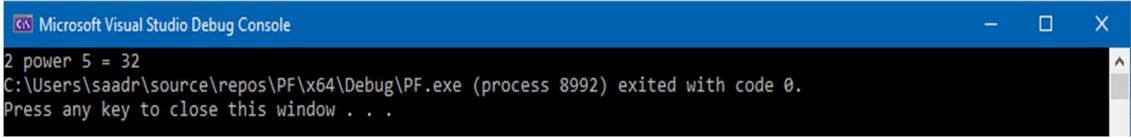
Function	Description
sqrt (x)	Square root of x
cbrt (x)	Cube root of x
exp (x)	Exponential function e^x
log (x)	Natural logarithm of x
log10 (x)	Logarithm of x (base 10)
fabs (x)	Absolute value of x
ceil (x)	Rounds x to smallest integer not less than x
floor (x)	Rounds x to largest integer not greater than x
pow (x, y)	x raised to the power y (x^y)
fmod (x, y)	Remainder of x/y
sin (x)	Sine of x
cos (x)	Cosine of x
tan (x)	Tangent of x

Use of power function is shown below:



Fig. 13 (Math Library Function)

The power function returns the result of 2^5 .

A screenshot of the Microsoft Visual Studio Debug Console. The window has a blue title bar with the text "Microsoft Visual Studio Debug Console" and standard window controls (minimize, maximize, close). The console area is black with white text. The output shows: "2 power 5 = 32", "C:\Users\saadr\source\repos\PF\x64\Debug\PF.exe (process 8992) exited with code 0.", and "Press any key to close this window . . .".

```
Microsoft Visual Studio Debug Console
2 power 5 = 32
C:\Users\saadr\source\repos\PF\x64\Debug\PF.exe (process 8992) exited with code 0.
Press any key to close this window . . .
```

Fig. 14 (Math Library Function)