****

**OBJECT ORIENTED PROGRAMMING (OOP)**

**By**

**Sir Usman Ahmed**

**ASSIGNMENT 1 (BSCS III)**

**Submitted by**

**NOOR BAHO**

**CSU-F12-116**

**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY**

**THE UNIVERSITY OF LAHORE, ISLAMABAD CAMPUS**

* **Define C++ syntax and one short example of the following.**

1. **Data Types:**
   1. **Integer:**

**Definition:** Variables with integer data type contains the numeric values. Integer variable normally requires 4 bytes of memory space to be stored and ranges from -2147483648 to 2147483647.

**Syntax:** While declaring and initializing the integer data type variable, “int” is used as keyword. Syntax is as followed:

int x=5; //A variable “x” is declared as int data type to store simple numeric (integer) values and is initialized with numeric value 5.

**Example:**

#include<iostream>

#include<conio.h>

//integer data type

using namespace std;

int main()

{

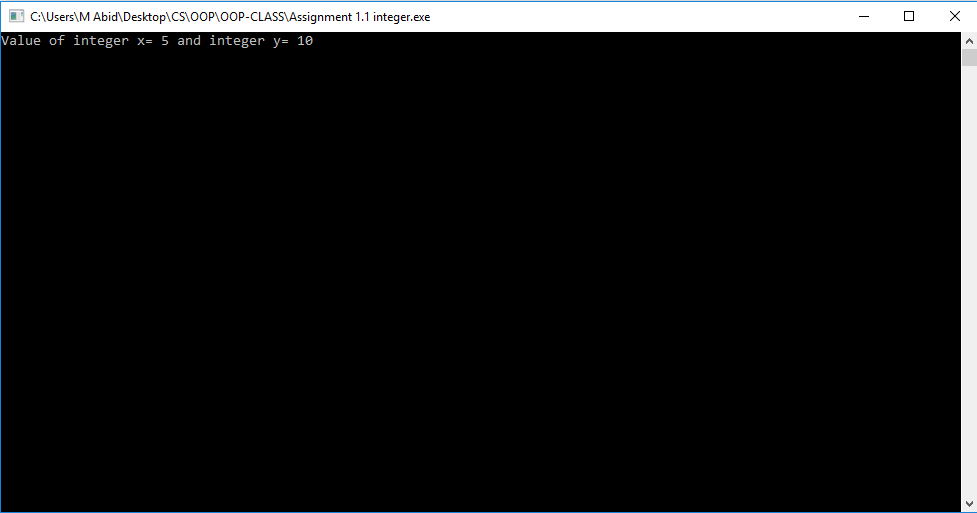
int x=5, y=10;

cout<<"Value of integer x= "<<x<<" and integer y= "<<y;

getch();

return 0;

}



**Figure 1.1**

* 1. **Character:**

**Definition:** To save alphabetic characters in a variable, we use the character data type, which allows us to store a single alphabetic character in a variable. Char type variable requires only 1 byte of memory to store its data and ranges from -128 to 127 for signed and 0 to 255 for unsigned character data type.

**Syntax:** Keyword “char” is used for character data type. Syntax of declaration & initialization is as followed.

char x=’A’; //while declaring character variable “char” is used as data type and while initializing the alphabetic value is written under inverted commas.

**Example:**

#include<iostream>

#include<conio.h>

//Character data type

using namespace std;

int main()

{

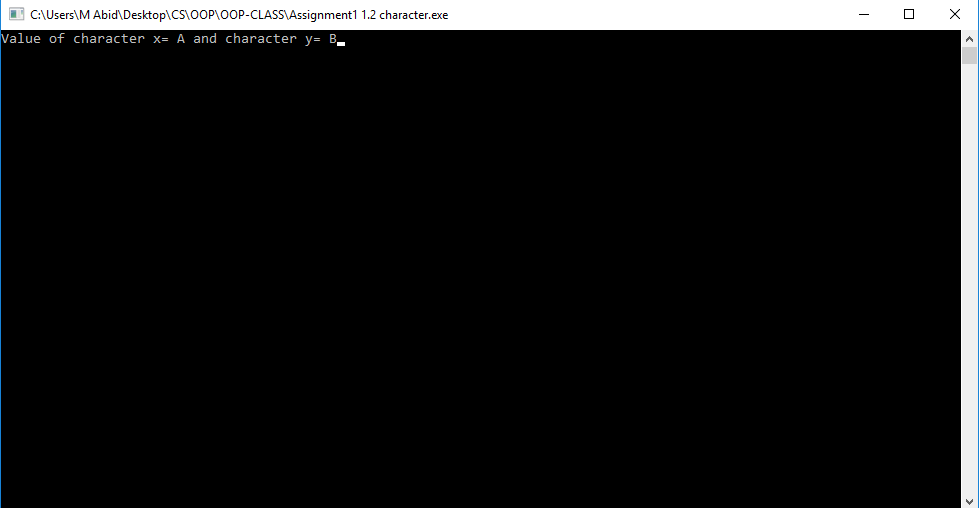
char x='A', y='B';

cout<<"Value of character x= "<<x<<" and character y= "<<y;

getch();

return 0;

}



**Figure 1.2**

* 1. **Floating Point:**
  2. **Definition:** As integer data type only store simply numeric values not the decimal values, so floating point data type fulfils this need and allows storing such numeric data that contains decimal values including characteristics and mantissa. Float data type needs 4 bytes in memory to store its data.

**Syntax:** “float” is the keyword used for floating point data type. Declaration and initialization is as followed.

float x=5.5; //x variable is declared as float data type to store floating point value 5.5

**Example:**

#include<iostream>

#include<conio.h>

//floating point data type

using namespace std;

int main()

{

float x=5.5, y=9.98;

cout<<"Value of floating point x= "<<x<<" and floating point y= "<<y;

getch();

return 0;

}



**Figure 1.3**

* 1. **Double Floating Point:**

**Definition:** As floating point consists of two parts, the precision value and the decimal value. So it also sometimes needs more space to store its larger value for which simple float data type sometimes overflows so double floating point data type provides double space to save doubled values then floating point data type. Where floating data type requires 4 bytes, double floating point requires 8 bytes to store data.

**Syntax:** keyword “double” is used to declare the data type of variable as double floating point. Explained as followed.

double x=4523.67; //x variable with double data type can now store larger decimal number as it occupies larger space in memory.

**Example:**

#include<iostream>

#include<conio.h>

//double floating point data type

using namespace std;

int main()

{

double x=2564.54, y=9734.98;

cout<<"Value of double floating point x= "<<x<<" and double floating point y= "<<y;

getch();

return 0;

}



**Figure 1.4**

* 1. **Boolean:**

**Definition:** Boolean data type stores the logical values of any operation. It only stores 1 or 0 either yes or no. It acquires 1 bit to store its data.

**Syntax:** Boolean data type is declared by “bool” keyword. Declaration is explained as followed.

bool x=TRUE; //

**Example:**

#include<iostream>

#include<conio.h>

//Boolean data type

using namespace std;

int main()

{

bool x;

cout<<"Enter 1 to continue and zero to terminate";

cin>>x;

if(x==1)

cout<<"You said to continue.";

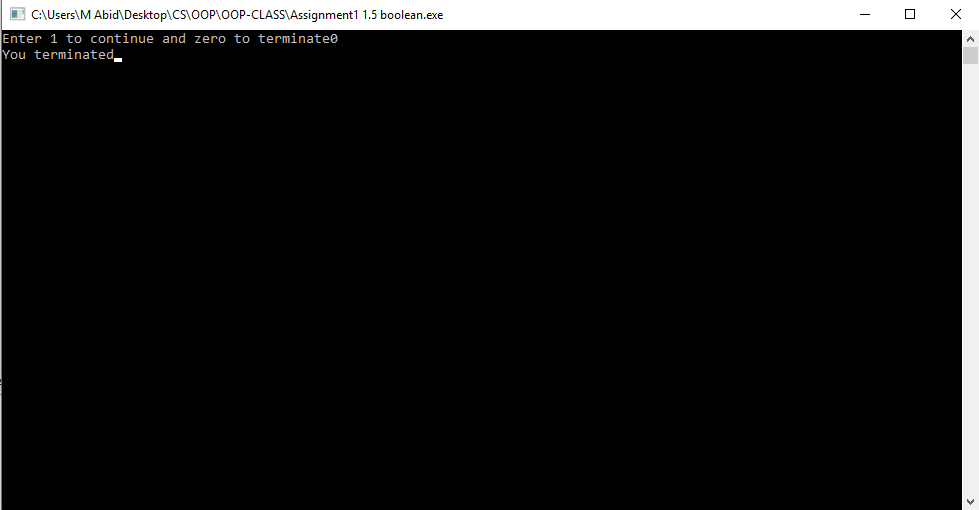
else

cout<<"You terminated";

getch();

return 0;

}



**Figure 1.5**

* 1. **String:**

**Definition:** String data type is to store number of alphabetical characters in a single variable. It stores a complete word, combination of number of characters in a string type variable.

**Syntax:** “string” keyword is used to declare a variable as string data type.

String x=”Welcome”; //complete word can be stored in a single variable with the help of string data type. Value is mentioned in inverted commas.

**Example:**

#include<iostream>

#include<conio.h>

//string data type

using namespace std;

int main()

{

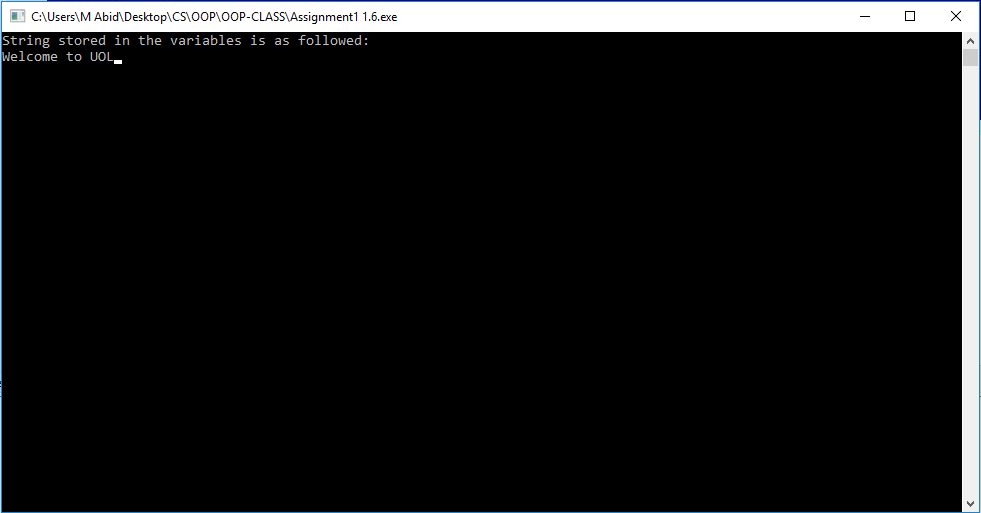
string x="Welcome", y="to", z="UOL";

cout<<"String stored in the variables is as followed: "<<endl<<x<<" "<<y<<" "<<z;

getch();

return 0;

}



**Figure 1.6**

1. **Arrays:**

**Definition:** Array is basically a data structure which provides to arrange data of same type in a particular sequence. In simple words, arrays are number of variables of same type structured to store data. Array consists of consecutive nodes of same type data. Array’s size is defined by the index of array through which the particular node of array is accessed and is being processed where needed.

Arrays are of two types: Single dimension array & multi-dimension array. The single dimension array contains indexes in a row where multi-dimension array contains data in the form of matrix. Multi-dimension array has two indexes where 1st index indicates the size of row and 2nd index indicates size of column of the matrix.

Same as variable, array can also be of any data type; int, char, float etc. Array is not a data type but a structure to manage same type of data.

**Syntax:** Syntax of array contains; array name, array index, data type of array and its initialization value. Array index is written in between square brackets “[ ]” as mentioned below. The index starts from 0 to n-1 number. If array index is defined by 5 that means it contains 5 number of consecutive nodes (spaces) to store data which starts form 0 to 4. In following example; array[2] means: array[0], array[1], array[2].

int array[2]={‘5’, ‘7’, ‘1’};

Where in multi-dimension array there are two indexes involved as already discussed.

int array[2][1]; //It means that there are two rows and one column in the array. We will access the nodes by: array[0][0], array[1][0].

**Example:**

#include<iostream>

#include<conio.h>

//array

using namespace std;

int main()

{

int array1[2]={5, 10};

int array2[2]={0, 1};

int array3[2]={0};

cout<<"Array 1: "<<endl;

for(int x=0; x<2; x++)

{

cout<<array1[x]<<" ";

}

cout<<endl<<"Array 2: "<<endl;

for(int x=0; x<2; x++)

{

cout<<array2[x]<<" ";

}

cout<<endl<<"Sum of two arrays is: "<<endl;

for(int a=0; a<2; a++)

{

array3[a]=array1[a]+array2[a];

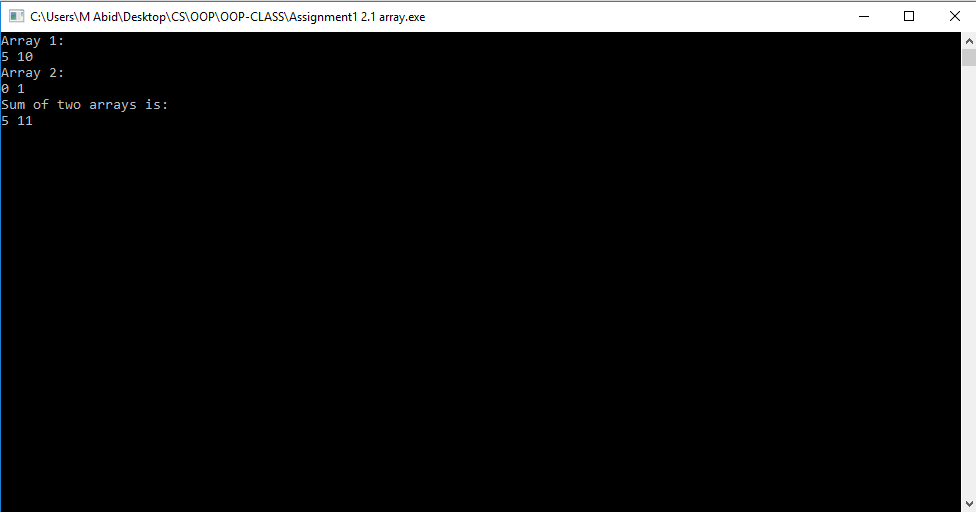
cout<<array3[a]<<" ";

}

getch();

return 0;

}­­­



**Figure 2.1**

1. **Loop Operations (for, while, do-while and switches):**

**Definition:** Loop operations are performed where statements are to be repeated again and again. Basically loop continues to execute a specific piece of code until the given condition goes false so the loop is designed to not perform those statements manually though they are set on a pet logic by which they perform and accumulate it automatically. Loops are of three types according to their syntax and working pattern; for loop, while loop and do-while loop whereas switch cases are type of conditional statements but they also perform in an automated pattern.

**Syntax for loop:** In for loop; a variable declared and initialized with a value, the loop condition and the increment statement above the body of statement. Compiler, first checks the condition, performs block of code if condition is true then increment the variable and again do the same process until the condition goes false and the loop will be terminated and exits from body of loop.

*for(int a=1; a<=5; a++) //for loop declared, variable a initialized with 1. As per condition loop will run for 5 times, each time value of a will be incremented by 1.*

*{ //body of loop starts*

*cout<<a<<” ”; //each time loop will execute this statement and print value of variable a.*

*} //body of loop ends, when loop condition will go false, it will be terminated and continued to the upcoming code.*

**Example:**

#include<iostream>

#include<conio.h>

//for loop

using namespace std;

int main()

{

cout<<"Printing integers from 1 to 100: "<<endl;

for(int a=1; a<=100; a++)

{

cout<<a<<" ";

}

getch();

return 0;

}

**Syntax while loop:** While loop performs same but the procedure differs. Variable for condition statement should be declared and initialized out of the loop and only condition statement is proceeded with in the loop. Let’s find same output by different loops.

int a=1; //variable a initialized out of loop

while(a<=5) //condition applied

{ //body of loop starts

cout<<a; //executable statements

a++; //increment statement

} //body of loop ends

**Example:**

#include<iostream>

#include<conio.h>

//while loop

using namespace std;

int main()

{

int a=1;

cout<<"Printing integers from 1 to 100: "<<endl;

while(a<=100)

{

cout<<a<<" ";

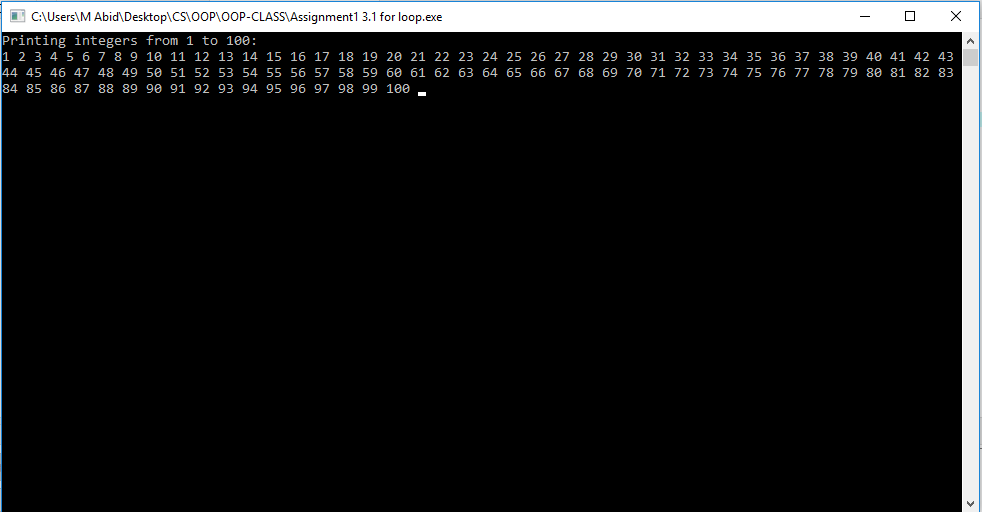
a++;

}

getch();

return 0;

}



**Figure 3.1**

**Syntax do-while loop:** Do-while loop is also a loop to perform block of code repeatedly but there is a bit difference among them. For loop and while loop will be terminated if the condition of loop goes false on first check but in do-while loop, it will do execute the code once then checks the condition, if condition goes wrong at first check, then it will terminate until then loop has been executed for once.

int a=1; //variable a initialized

do{ //body of loop starts and execute minimum for one time

cout<<a; //body of loop

a++; //incremental statement

}while(a<=100); //after execution of loop one time, condition checked and if true then continues, if false then terminates and continues to next block.

**Example:**

#include<iostream>

#include<conio.h>

//while loop

using namespace std;

int main()

{

int a=1;

cout<<"Body of loop executed even condition is false: "<<endl;

do

{

cout<<a<<" ";

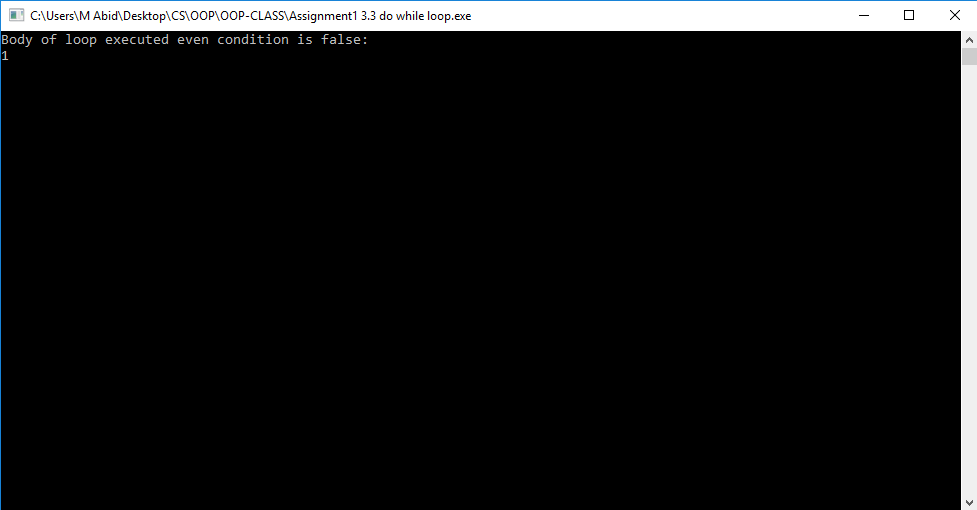
a++;

}while(a==10);

getch();

return 0;

}



**Figure 3.2**

**Syntax switches:** Switches are used as conditional statement, on the basis of specific input, it is decided which case should be executed and which should not be. Switch statement passes control to different cases accordingly. Break; and continue; statements are also used in switches to break the current block and to continue the current block respectively. Also a default case exists which is like else statement, if no case can be switched to be executed, the default will be activated and executed.

int x;

switch(x)

{

case 1:

cout<<”case 1 executed.”;

break;

case2:

cout<<”case 2 executed.”;

break;

.

.

.

case n:

cout<<”case n executed”;

break;

default:

cout<<”no case executed but default does”;

}

**Example:**

#include<iostream>

#include<conio.h>

//while loop

using namespace std;

int main()

{

int a,b, opt;

cout<<"Enter a number: ";

cin>>a;

cout<<"Enter a number: ";

cin>>b;

cout<<"Enter 1 to add them, 2 to subtract";

cin>>opt;

switch(opt)

{

case 1:

cout<<"Sum of two number is: "<<a+b;

break;

case2:

cout<<"Difference of two number is: "<<a-b;

break;

default:

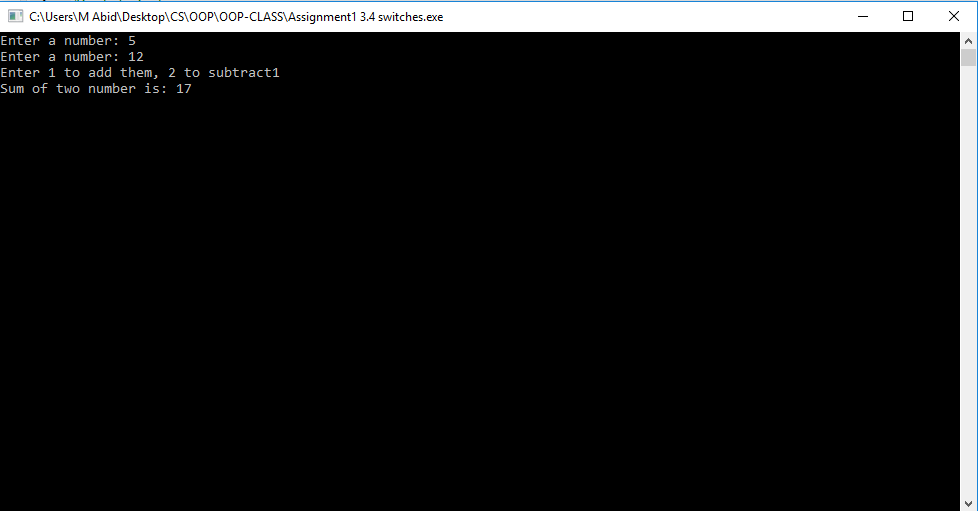
cout<<"no operation selected.";

}

getch();

return 0;

}



**Figure 3.3**

1. **Functions:**

**Definition:** A function is defined just to optimize the code, if same line of code has to be written again and again, it is defined in a function and the function could be call at any stage with its name in just one statement, it saves time and resources as well.

Functions are of two types; Built-in functions and User-defined functions. Built-in functions are built-in the library of the compiler those can be used by adding relevant header file. i.e: square root and power functions are in cmath.h library. User defined functions are such functions that user write by themselves according to his requirements.

From scope aspect, same like variables, functions are also of two types; global scope and local scope. Scope of functions depends on its declaration statement that where in the whole program, functions’ declaration exist that decides its scope.

From structure aspect functions are of four types depends upon its return data type and arguments. Function that returns some value is known as returning function else non-returning function. Function which take arguments (values) as input are known as with argument function else without arguments function.

1. Returning function with arguments. int add(int a, int b);
2. Non-returning function with arguments. void add(int a, int b);
3. Returning function without arguments. int add();
4. Non-returning function without arguments. void add();

**Syntax:** function should be declared out of main function. Declaration tells function return value data type, function’s name, function taking arguments and closed with semicolon; also known as function prototype.

int add (int a, int b);

After then function is defined within the main function, it can be defined with the function declaration.

**Example:**

#include<iostream>

#include<conio.h>

//functions

using namespace std;

int add(int x, int y)

{

return x+y;

}

int main()

{

int a,b;

cout<<"Enter a number: ";

cin>>a;

cout<<"Enter a number: ";

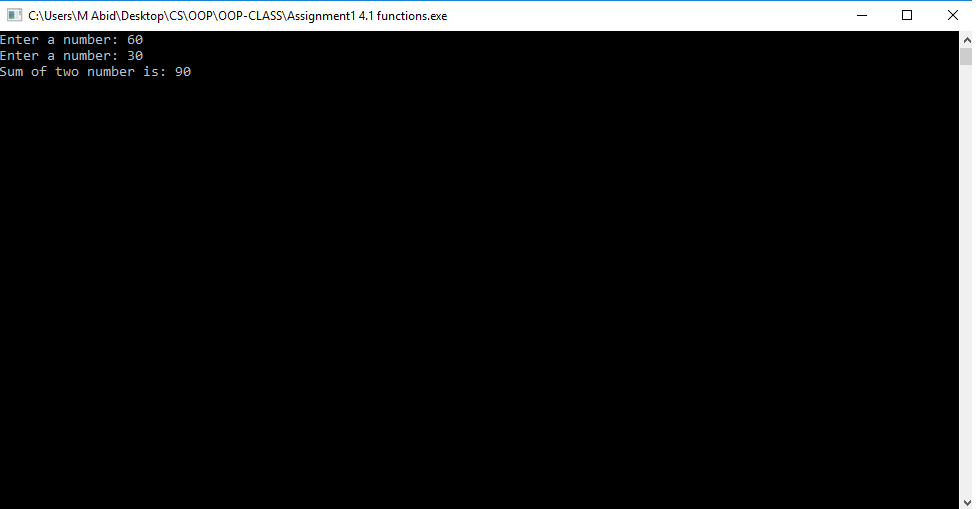
cin>>b;

cout<<"Sum of two number is: "<<add(a,b);

getch();

return 0;

}



**Figure 4.1**

1. **Pointers:**

**Definition:** Whenever a variable is declared and initialized, its value is saved somewhere in the memory but the address of memory where that particular data is saved, is maintained by the pointer. Pointer is an object that defines the addresses of values saved by variables etc.

**Syntax:** If a variable int xyz is declared, the pointer for this variable xyz will be \*xyz; the value stored at address of variable xyz can be accessible through pointer of the variable that is \*xyz.

**Example:**

#include<iostream>

#include<conio.h>

//functions

using namespace std;

int add(int x, int y)

{

return x+y;

}

int main()

{

int a=5,b;

int \*p=&a;

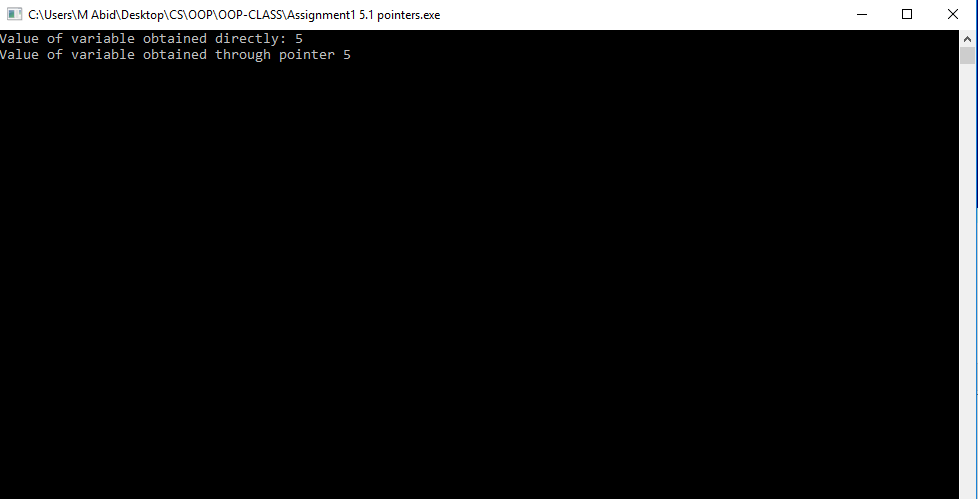
cout<<"Value of variable obtained directly: "<<a<<endl;

cout<<"Value of variable obtained through pointer "<<\*p;

getch();

return 0;

}



**Figure: 5.1**