SRI INDU INSTITUTE OF ENGINEERING &TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

A.Y:2019-2020(R18)

C++ PROGRAMMING LAB MANUAL

CS309PC: C++ PROGRAMMING LAB

B.Tech. II Year I Sem. L T/P/D C 0 0/2/0 1

Prerequisites: A course on "Programming for Problem Solving".

Course Objectives

- Introduces object-oriented programming concepts using the C++ language.
- Introduces the principles of data abstraction, inheritance and polymorphism;
- Introduces the principles of virtual functions and polymorphism
- Introduces handling formatted I/O and unformatted I/O
- Introduces exception handling
- Course Outcomes Ability to develop applications for a range of problems using object-orientedprogramming Techniques

LIST OF EXPERIMENTS

1. Write a C++ Program to display Names, Roll No., and grades of 3 students who have appeared in the examination. Declare the class of name, Roll No. and grade. Create an array of class objects. Read and display the contents of the array.
2. Write a C++ program to declare Struct. Initialize and display contents of member variables.
3. Write a C++ program to declare a class. Declare pointer to class. Initialize and display the contents of the class member.
4. Given that an EMPLOYEE class contains following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary and print data members.
5. Write a C++ program to read the data of N employee and compute Net salary of each employee (DA=52% of Basic and Income Tax (IT) =30% of the gross salary).
6. Write a C++ to illustrate the concepts of console I/O operations.
7. Write a C++ program to use scope resolution operator. Display the various values of the same variables declared at different scope levels.
8. Write a C++ program to allocate memory using new operator.
9. Write a C++ program to create multilevel inheritance. (Hint: Classes A1, A2, A3)
10. Write a C++ program to create an array of pointers. Invoke functions using array objects.
11. Write a C++ program to use pointer for both base and derived classes and call the member function. Use Virtual keyword

EXPERIMENT- 1

AIM: Write a C++ program to display names, roll no and grades of 3 students whohave appeared in examination. declare the class of name, roll no and grade. create an array of class objects .read and display the contents of array.

PROCEDURE:

Step 1 - Include the required header files (iostream.h and conio.h).
Step 2 - Create a class (StudentInfo) with the following class members as public members.
student_name, roll_number and grade as data members.
read_info() and display_info() as member functions.
Step 3 - Implement the read_info() and display_info() member functions.
Step 4 - Create a main() method.
Step 5 - Create an object of the above class inside the main() method.
Step 6 - Make function calls to read_info() and display_info() using class object.

PROGRAM:

```
#include <iostream>
using namespace std;
class Student_Info{
   int roll_number;
   char student_name[50], grade[2];
   public:
      void read_data(int count){
         cout<<"\n\n--------- Enter student "<<count+1<<" information ---------\n";
         cout<<"Name of the Student (Max. 50 characters only): ";
         cin>>student_name;
         cout<<"Roll Number: ";
         cin>>roll_number;
         cout<<"Grade (O, A+, A, B+, B, C, D, F): ";
         cin>>grade;
         cout<<"\nStudent information with roll number "<<roll_number<<" has saved!";
      }
      void display_data(int count){
         cout<<"\n\n******** Student "<<count+1<<" Information ********";
         cout<<"\nName of the Student: "<<student_name;
         cout<<"\nRoll Number: "<<roll_number;
         cout<<"\nGrade Secured: "<<grade;
         cout<<"\n--------------------------------------\n";
```

```
        }
};
int main(){
    Student_Info stud[3];
    int i;
    for(i=0; i<3; i++)
        stud[i].read_data(i);
    cout<<"\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
    cout<<"The information of 3 students has been saved.";
    cout<<"\n++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
    for(i=0; i< i++)
        stud[i].display_data(i);
    return 0   }
```

OUTPUT:

```
"C:\Users\User\Desktop\New folder\Student_Info\bin\Debug\Student_Info.exe"

--------- Enter student 1 information ---------
Name of the Student (Max. 50 characters only): Rama
Roll Number: 111
Grade (O, A+, A, B+, B, C, D, F): A+

Student information with roll number 111 has saved!

--------- Enter student 2 information ---------
Name of the Student (Max. 50 characters only): Seetha
Roll Number: 222
Grade (O, A+, A, B+, B, C, D, F): O

Student information with roll number 222 has saved!

--------- Enter student 3 information ---------
Name of the Student (Max. 50 characters only): Shyam
Roll Number: 333
Grade (O, A+, A, B+, B, C, D, F): B

Student information with roll number 333 has saved!

++++++++++++++++++++++++++++++++++++++++++++++
The information of 3 students has been saved.
++++++++++++++++++++++++++++++++++++++++++++++


******** Student 1 Information ********
Name of the Student: Rama
Roll Number: 111
Grade Secured: A+İ
--------------------------------------


******** Student 2 Information ********
Name of the Student: Seetha
Roll Number: 222
Grade Secured: O
--------------------------------------


******** Student 3 Information ********
Name of the Student: Shyam
Roll Number: 333
Grade Secured: B
--------------------------------------

Process returned 0 (0x0)   execution time : 35.633 s
Press any key to continue.
```

EXPERIMENT – 2

AIM: Write a c++ program to declare struct, intialize and display contents of member varaiables

PROCEDURE:

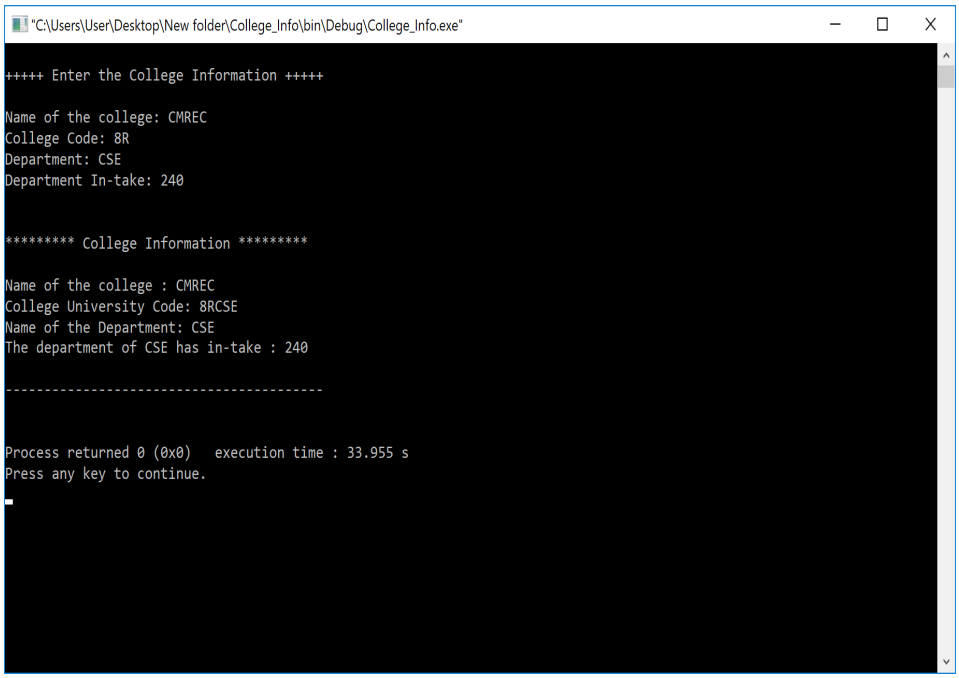
- **Step 1 -** Include the required header files (iostream.h and conio.h).
- **Step 2 -** Create a structure (college_info) with the following member variables. college_name, college_code, dept, and intake as data members.
- **Step 3 -** Create a main() method.
- **Step 4 -** Create a variable of the above structure inside the main() method.
- **Step 5 -** Then, read data into member variables of that structure using structure variable with dot operator.
- **Step 6 -** Finally, display data of member variables of that structure using structure variable with dot operator.

PROGRAM:

```
#include <iostream>
using namespace std;
struct college_info{
   char college_name[15];
   char college_code[2];
   char dept[50];
   int intake;
};
int main()
{
   struct college_info college;
   cout<<"\n+++++ Enter the College Information +++++\n\n";
   cout<<"Name of the college: ";
   cin>>college.college_name;
```

```
    cout<<"College Code: ";
    cin>>college.college_code;
    cout<<"Department: ";
    cin>>college.dept;
    cout<<"Department In-take: ";
    cin>>college.intake;
    cout<<"\n\n********* College Information *********\n\n";
    cout<<"Name of the college : "<<college.college_name;
    cout<<"\nCollege University Code: "<<college.college_code;
    cout<<"\nName of the Department: "<<college.dept;
    cout<<"\nThe department of "<<college.dept<<" has in-take : "<<college.intake;
    cout<<"\n\n----------------------------------------\n\n";
    return 0;
}
```

OUTPUT:



EXPERIMENT - 3

AIM: Write a C++ program to declare a class, declare pointer to class, initailize and display contents of class member.

PROCEDURE:

- **Step 1 -** Include the required header files (iostream.h, conio.h, and windows.h for colors).
- **Step 2 -** Create a class (Rectangle) with the following members as public members. length and breadth as data members. initialize(), getArea() and display() as member functions.
- **Step 3 -** Create a main() method.
- **Step 4 -** Create a variable (**rect**) and a pointer variable (**class_ptr**) of the above class inside the main() method.
- **Step 5 -** Assign the address of class object (**rect**) to class pointer object (**class_ptr**).
- **Step 6 -** Then, call the member functions initialize() and display() using class pointer object (**class_ptr**) to illustrate member functions access using class pointer.
- **Step 7 -** Assign values to data members of the class using class pointer object to illustrate data members access using class pointer.
- **Step 8 -** Finally, call the member functions initialize() and display() using class pointer object (**class_ptr**).

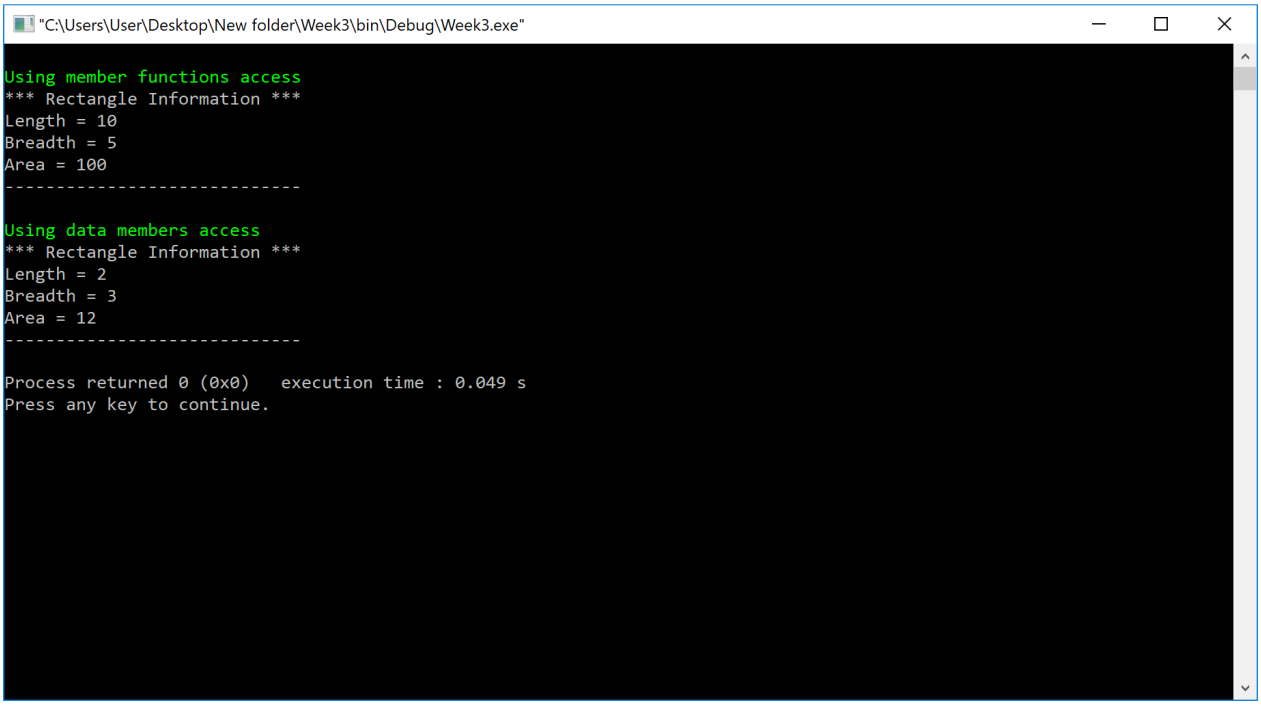
PROGRAM:

```
#include <windows.h>
#include <iostream>
using namespace std;
class RectangleTest{
   public:
      int length, breadth;
   public:
      void initialize(int len, int bre){
         length = len;
         breadth = bre;
      }
      int getArea(){
         return 2*length*breadth;
```

```
    }
    void display(){
        int area = getArea();
        cout<<"\n*** Rectangle Information ***\n";
        cout<<"Length = "<<length;
        cout<<"\nBreadth = "<<breadth;
        cout<<"\nArea = "<<area;
        cout<<"\n----------------------------\n";
    }
};
int main()
{
    RectangleTest rect, *class_ptr;
    HANDLE color=GetStdHandle(STD_OUTPUT_HANDLE);
    class_ptr = &rect;
    //Accessing member functions using class pointer
    SetConsoleTextAttribute(color, 10 );    //Setting color Green
    cout<<"\nUsing member functions access";
    SetConsoleTextAttribute(color, 7 );     //Setting color White
    class_ptr->initialize(10,5);
    class_ptr->display();
    //Accessing data members using class pointer
    SetConsoleTextAttribute(color, 10 );    //Setting color Green
    cout<<"\nUsing data members access";
    SetConsoleTextAttribute(color, 7 );     //Setting color White
    class_ptr->length = 2;
    class_ptr->breadth = 3;
    class_ptr->initialize(class_ptr->length,class_ptr->breadth);
    class_ptr->display();
    return 0;
```

}

OUTPUT:


"C:\Users\User\Desktop\New folder\Week3\bin\Debug\Week3.exe"
Using member functions access
*** Rectangle Information ***
Length = 10
Breadth = 5
Area = 100

Using data members access
*** Rectangle Information ***
Length = 2
Breadth = 3
Area = 12

Process returned 0 (0x0) execution time : 0.049 s
Press any key to continue.

.

EXPERIMENT - 4

AIM: Given that an employee class contains following members, data members, employee number, employee name ,basic, DA, IT, net salary and print data members

PROCEDURE:

- **Step 1 -** Include the required header files (iostream.h, conio.h, and windows.h for colors).
- **Step 2 -** Create a class (employee) with the following members as public members. emp_number, emp_name, emp_basic, emp_da, emp_it, and emp_net_sal as data members. get_emp_details(), find_net_salary() and show_emp_details() as member functions.
- **Step 3 -** Implement all the member functions with their respective code (Here, we have used scope resolution operator ::).
- **Step 3 -** Create a main() method.
- **Step 4 -** Create an object (**emp**) of the above class inside the main() method.
- **Step 5 -** Call the member functions get_emp_details() and show_emp_details().
- **Step 6 -** returnn 0 to exit form the program execution.

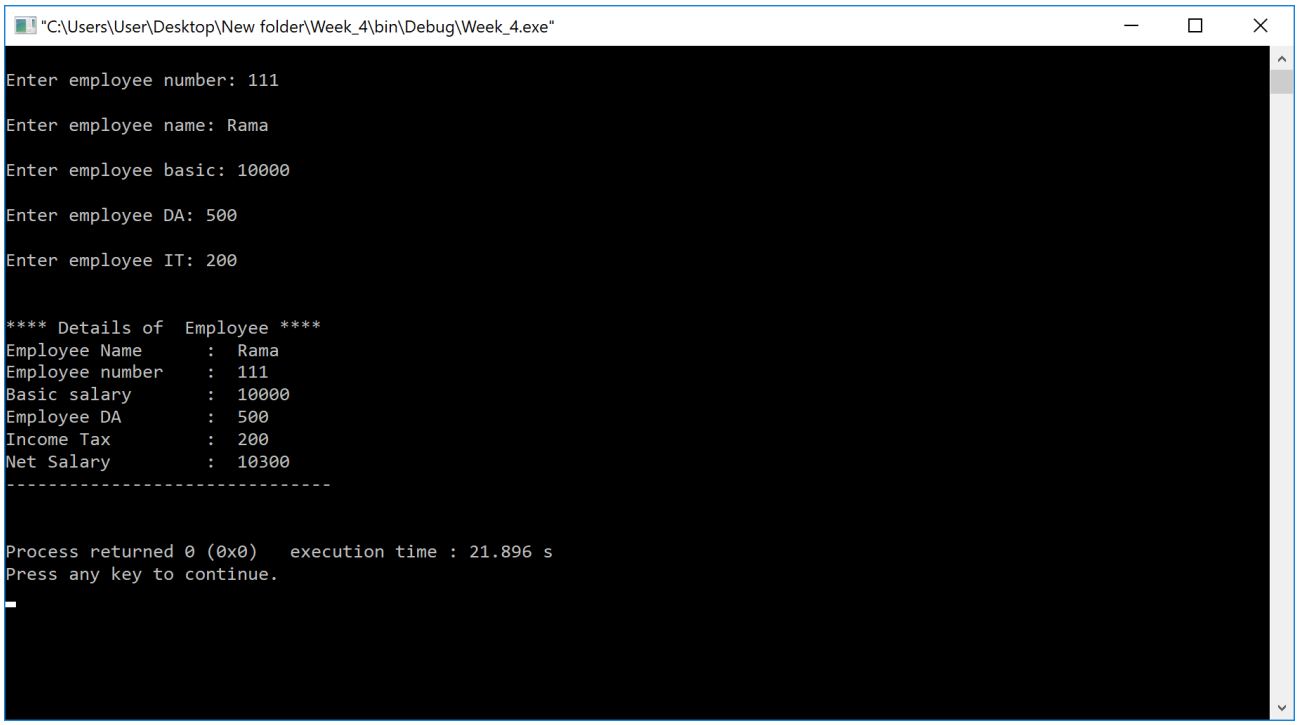
PROGRAM:

```
#include <windows.h>
#include <iostream>
using namespace std;
class employee
{
        int   emp_number;
        char  emp_name[20];
        float emp_basic;
        float emp_da;
        float emp_it;
        float emp_net_sal;
        public:
                void get_emp_details();
                float find_net_salary(float basic, float da, float it);
                void show_emp_details();
```

```
};
void employee :: get_emp_details()
{
        cout<<"\nEnter employee number: ";
        cin>>emp_number;
        cout<<"\nEnter employee name: ";
        cin>>emp_name;
        cout<<"\nEnter employee basic: ";
        cin>>emp_basic;
        cout<<"\nEnter employee DA: ";
        cin>>emp_da;
        cout<<"\nEnter employee IT: ";
        cin>>emp_it;
}
float employee :: find_net_salary(float basic, float da, float it)
{
   return (basic+da)-it;
}
void employee :: show_emp_details()
{
        cout<<"\n\n**** Details of  Employee ****";
        cout<<"\nEmployee Name      :  "<<emp_name;
        cout<<"\nEmployee number    :  "<<emp_number;
        cout<<"\nBasic salary       :  "<<emp_basic;
        cout<<"\nEmployee DA        :  "<<emp_da;
        cout<<"\nIncome Tax         :  "<<emp_it;
        cout<<"\nNet Salary         :  "<<find_net_salary(emp_basic, emp_da, emp_it);
        cout<<"\n-------------------------------\n\n";
}
int main()
```

```
{
    employee emp;
    emp.get_emp_details();
    emp.show_emp_details();
    return 0;
}
```

OUTPUT:



EXPERIMENT - 5

AIM: Write a C++ program to read the data of N employee and compute net salary of each employee (DA=52%of basic and income tax(IT)=30% of gross salary.

PROCEDURE:

- **Step 1 -** Include the required header files (iostream.h, conio.h, and windows.h for colors).
- **Step 2 -** Create a class Employee with the following members. emp_number, emp_name, basic, da, it, gross_salary, and net_salary as data members read_emp_details(), find_net_salary(), and display_emp_details() as member functions.
- **Step 3 -** Implement all the member functions with their respective code.
- **Step 4 -** Create a main() method.
- **Step 5 -** Create an array of class object with a specific size and number_of_emp as integer.
- **Step 6 -** Read number_of_emp.
- **Step 7 -** Call the read_emp_details() method through the array of class object from 0 to number_of_emp.
- **Step 8 -** Call the find_net_salary() method through the array of class object from 0 to number_of_emp.
- **Step 9 -** Call the display_emp_details() method through the array of class object from 0 to number_of_emp.
- **Step 10 -** returnn 0 to exit form the program execution.

PROGRAM:

```
#include<iostream.h>
#include<conio.h>
class Employee
{
  char emp_name[30];
  int emp_number;
  float basic, da, it, gross_salary, net_salary;
  public:
  void read_emp_details(int count){
          cout<<"\n\n*** Enter Employee "<<count<<" Details ***";
          cout<<"\nEmployee Number: ";
```

```
            cin>>emp_number;
            cout<<"Employee Name: ";
            cin>>emp_name;
            cout<<"Basic Salary: ";
            cin>>basic;
            cout<<"\n---- Employee "<<count<<" Datails are saved ----\n\n";
  }
  float find_net_salary(){
    da = basic * 0.52;
    gross_salary = basic + da;
    it = gross_salary * 0.30;
    net_salary = (basic + da) - it;
    return net_salary;
  }
  void display_emp_details(int count){
    cout<<"\n\n*** Employee "<<count<<" Details ***\n";
    cout<<"\nEmployee Number        : "<<emp_number;
    cout<<"\nEmployee Name          : "<<emp_name;
    cout<<"\nNet Salary: "<<net_salary;
    cout<<"\n--------------------------\n";
  }
};
int main(){
  Employee emp[100];
  int number_of_emp, count;
  clrscr();
  cout<<"\nPlease enter the number of Employees (Max. 100): ";
  cin>>number_of_emp;
  for(count=0; count< number_of_emp; count++){
    emp[count].read_emp_details(count+1);
```

```
  }
  for(count=0; count < number_of_emp; count++){
    emp[count].find_net_salary();
  }
  for(count=0; count < number_of_emp; count++){
    emp[count].display_emp_details(count+1);
  }
  cout<<"\nPress any key to close!!!";
  getch();
  return 0;
}
```

OUTPUT:

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TC
Please enter the number of Employees (Max. 100): 3


*** Enter Employee 1 Details ***
Employee Number: 111
Employee Name: Raja
Basic Salary: 16000

---- Employee 1 Datails are saved ----



*** Enter Employee 2 Details ***
Employee Number: 222
Employee Name: Shyam
Basic Salary: 12000

---- Employee 2 Datails are saved ----



*** Enter Employee 3 Details ***
Employee Number: 333
Employee Name: Seetha
Basic Salary: 15000_
```

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TC

*** Employee 1 Details ***

Employee Number : 111
Employee Name   : Raja
Net Salary: 17024
-------------------------


*** Employee 2 Details ***

Employee Number : 222
Employee Name   : Shyam
Net Salary: 12768
-------------------------


*** Employee 3 Details ***

Employee Number : 333
Employee Name   : Seetha
Net Salary: 15960
-------------------------

Press any key to close!!!_
```

EXPERIMENT - 6

AIM: Write a C++ to illustrate the concepts of console I/O operations.

There are mainly two types of consol IO operations.

1. **Unformatted consol IO -**
2. **Formatted consol IO**

6.1. Unformatted console IO operations

We use the following built-in functions to perform operations of type unformatted consol IO operations.

6.1 A) get( ) and put( )

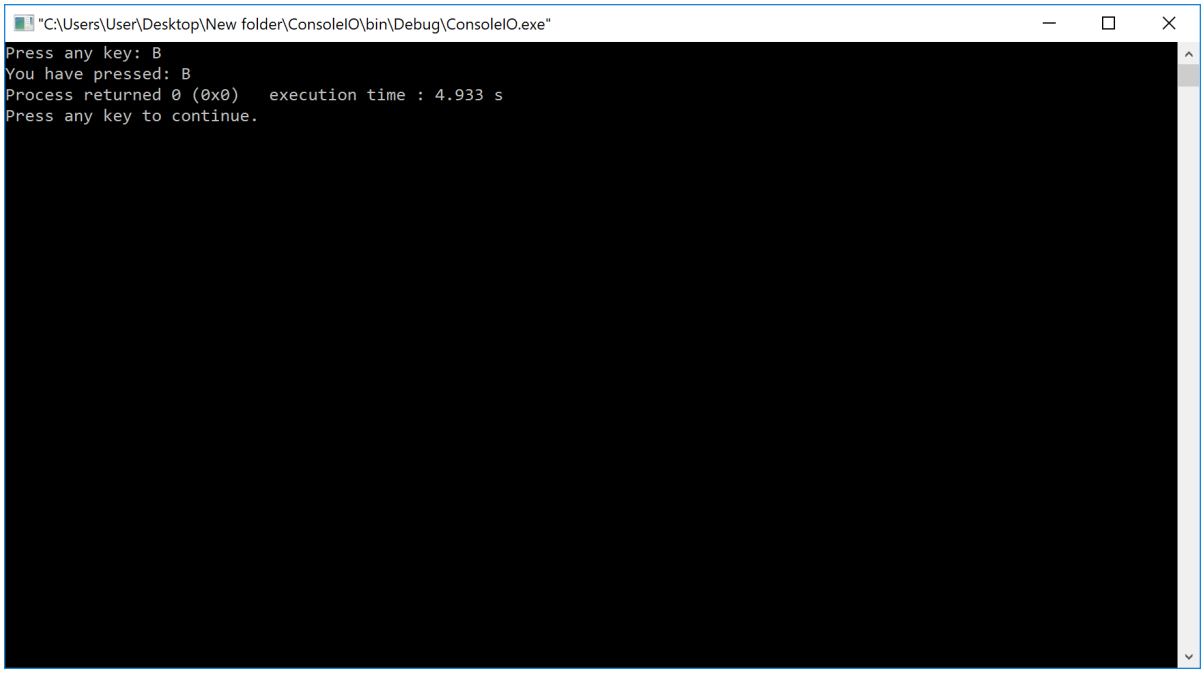
The **get( )** is a method of **cin** object used to input a single character from the standard input device (keyboard). Its main property is that it allows wide spaces and newline character. The get() function does not have any return value (void get( )).

The **put()** is a method of **cout** object and it is used to print the specified character on standard output device (monitor).

PROGRAM:

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   cout<<"Press any key: ";
   ch = cin.get();
   cout << "You have pressed: ";
   cout.put(ch);
   return 0;
}
```

OUTPUT:



6.1 B) getline(char *buffer,int size) and write(char * buffer, int n)

The **getline(char *buffer,int size)** is a method of **cin** object and it is used to input a string with multiple spaces.

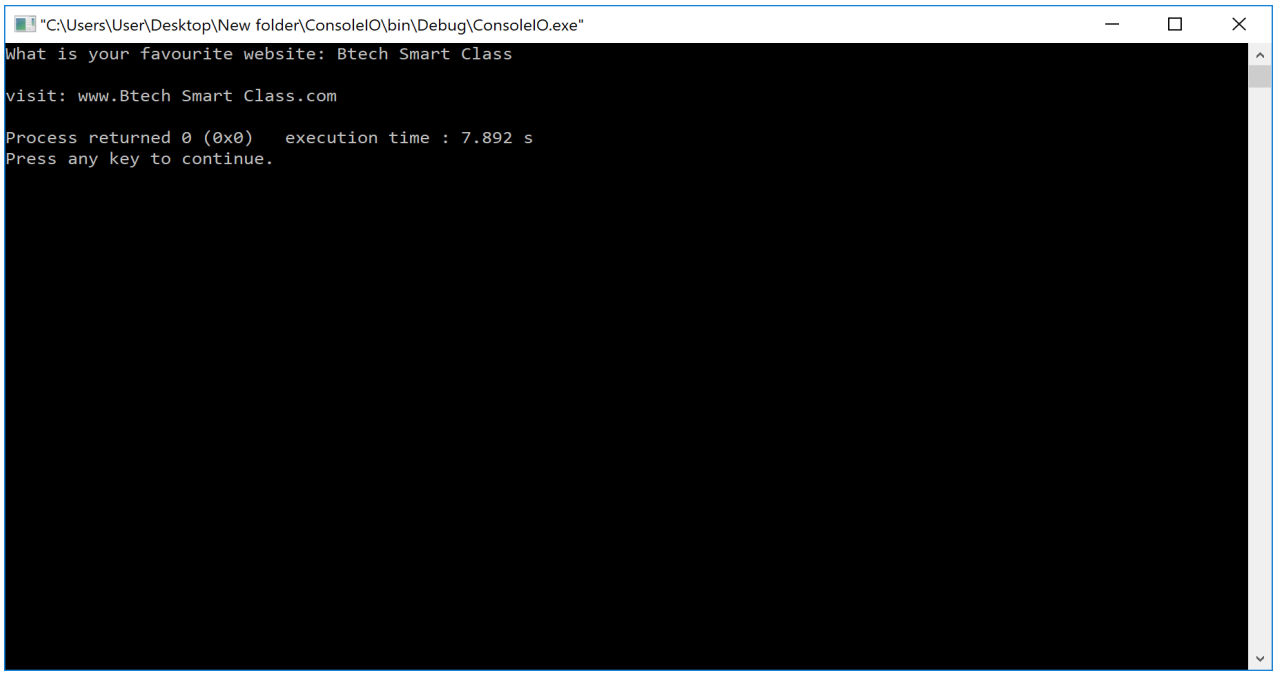
The **write (char * buffer, int n)** is a method of **cout** object and it is used to read n character from buffer variable.

PROGRAM:

```
#include <iostream>

using namespace std;

int main()

{

   char ch[20];

   cout<<"What is your favourite website: ";

   cin.getline(ch, 20);

   cout <<endl<< "visit: www.";

   cout.write(ch, 17);

   cout<<".com"<<endl;

   return 0;
```

}

OUTPUT:



6.1 C) cin and cout objects

The **cin** is the object used to take input value of any variable of any type. It must be used with an overloaded operator **"<<"**.

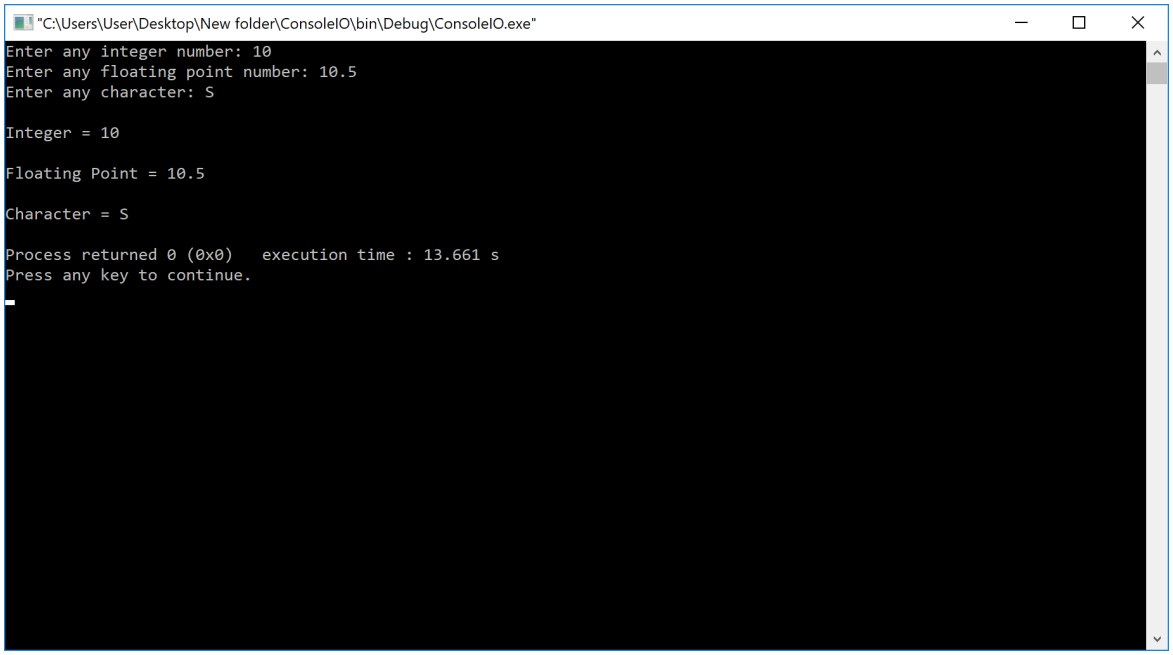
The **cout** is an object used to print string and variable values. It must be used with an overloaded operator **"<<"**.

Example

```
#include <iostream>

using namespace std;

int main()

{

   int variable_int;

   float variable_float;

   char variable_char;


   cout<<"Enter any integer number: ";

   cin>>variable_int;
```

```
    cout<<"Enter any floating point number: ";
    cin>>variable_float;
    cout<<"Enter any character: ";
    cin>>variable_char;
    cout <<endl<< "Integer = "<<variable_int<<endl;
    cout <<endl<< "Floating Point = "<<variable_float<<endl;
    cout <<endl<< "Character = "<<variable_char<<endl;
    return 0;
}
```

OUTPUT:



6.2) Formatted console IO operations

The C++ programming language provides the following built-in functions to display the output in formatted form. These built-in functions are available in the header file **iomanip.h**.

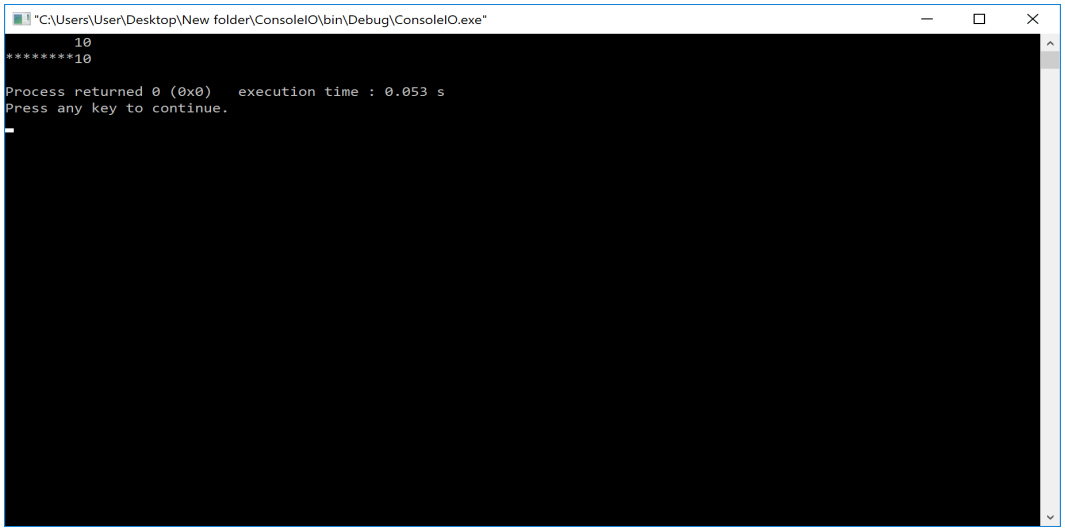
6.2 A) setw(int) and setfill(char)

The **setw(int)** is a method used to set width of the output.
The **setfill(char)** is a method used to fill specified character at unused space.

PROGRAM:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   int x=10;
   cout<<setw(10);
   cout<<x<<endl;
   cout<<setw(10)<<setfill('*')<<x<<endl;
   return 0;
}
```

OUTPUT:



EXPERIMENT - 7

AIM: Write a C++ program to use scope resolution operator. Display the various values of the same variables declared at different scope levels.

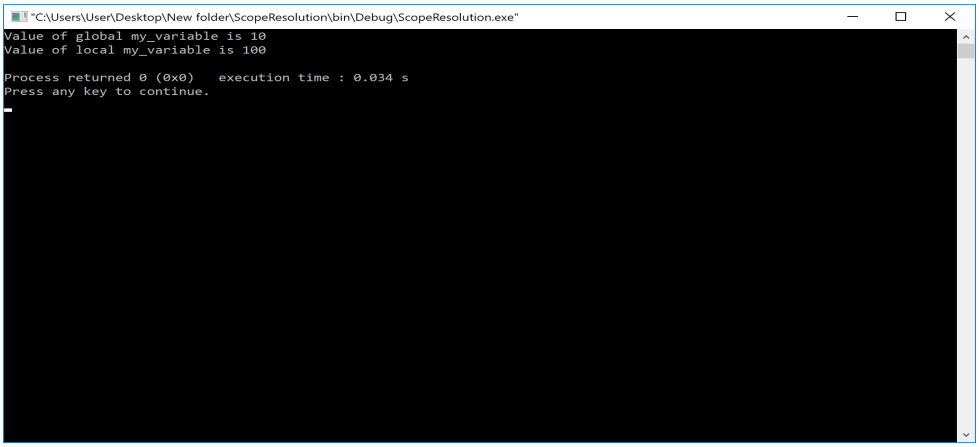
7 A) Accessing Global variable

When both global and local variables have the same name then global variable in hidden inside the local scope. Here, we use the scope resolution operator to refer to the global variable.

PROGRAM:

```
#include <iostream>

using namespace std;

int my_variable = 10;  // Global variable my_variable

int main()

{

  int my_variable = 100; // Local variable my_variable

  cout << "Value of global my_variable is " << ::my_variable << endl;

  cout << "Value of local my_variable is " << my_variable << endl;

  return 0;

}
```

OUTPUT:



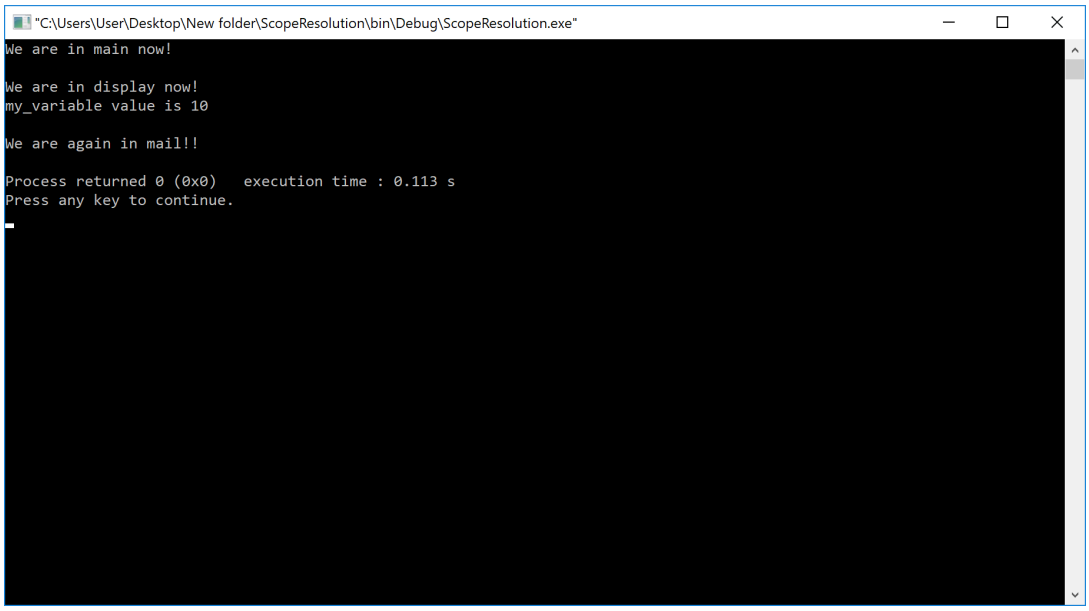
7 B) Defining a function out of class

When a class has a function prototype inside but the definition is outside of the class. Here, to define the function outside the class we use the scope resolution operator.

PROGRAM:

```
#include <iostream>
using namespace std;
class My_Class
{
public:
    int my_variable = 10;
    void display(); //Prototype of display function
};
void My_Class :: display(){ //Defining function using Scope Resolution Operator
   cout << endl << "We are in display now!" << endl;
   cout << "my_variable value is " << my_variable << endl << endl;
}
int main(){
    My_Class obj;
    cout << "We are in main now << endl;
    obj.display();
    cout << "We are again in mail!!" << endl;
    return 0;
}
```

OUTPUT:



7 C) Accessing static members of a class

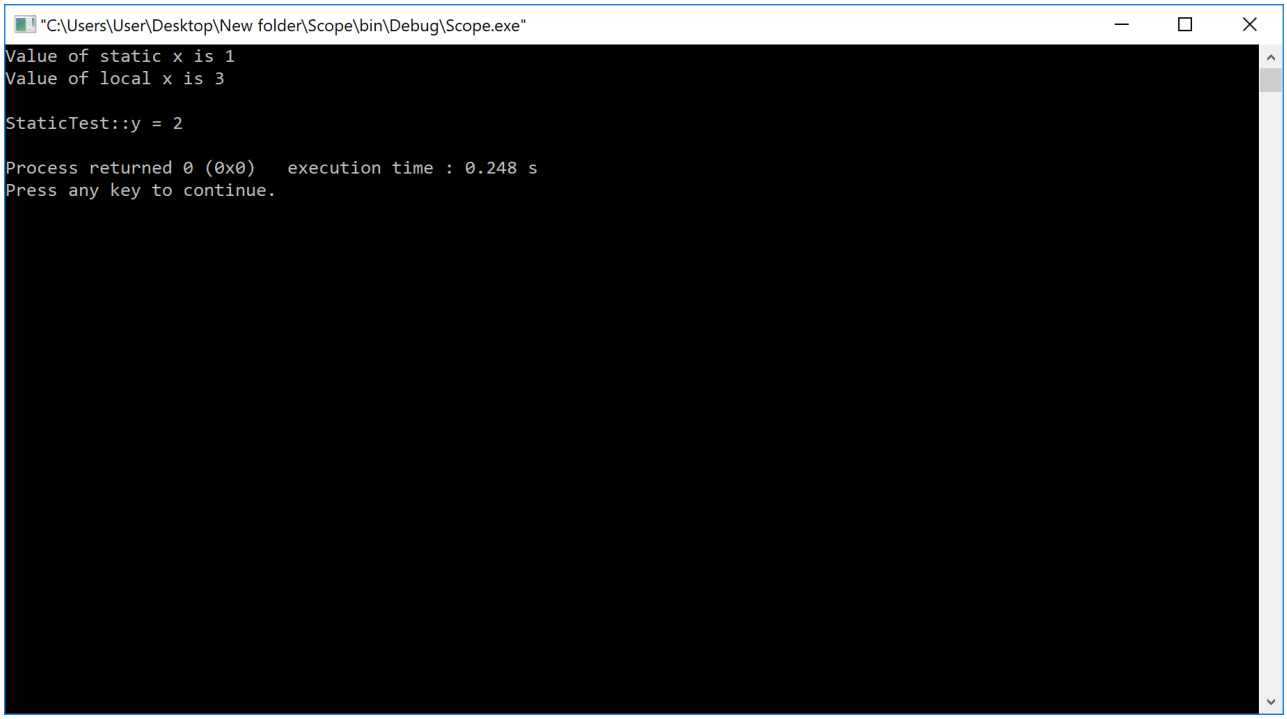
The scope resolution operator can be used to access static members of a class when there is a local variable with the same name.

PROGRAM:

```
#include <iostream>

using namespace std;

class StaticTest

{

   static int x;

public:

   static int y;

   // Local parameter 'x' hides class member

   // 'x', but we can access it using ::

   void my_function(int x)

   {

      // We can access class's static variable

      // even if there is a local variable
```

```
        cout << "Value of static x is " << StaticTest::x;
        cout << "\nValue of local x is " << x;
    }
};
// In C++, static members must be explicitly defined like this
int StaticTest::x = 1;
int StaticTest::y = 2;
int main()
{
    StaticTest obj;
    int x = 3 ;
    obj.my_function(x);
    cout << "\n\nStaticTest::y = " << StaticTest::y << endl;
    return 0;
}
```

OUTPUT:

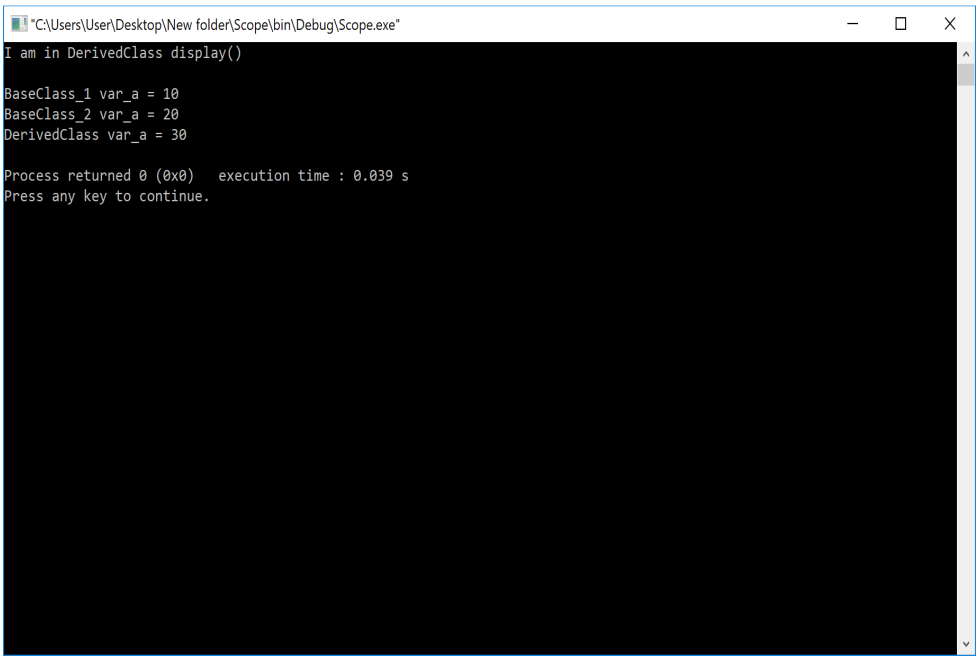


7 D) Scope Resolution with Multiple Inheritance

```
#include <iostream>

using namespace std;

class BaseClass_1{
   public:
      int var_a = 10;
      void display(){
         cout<<"I am in BaseClass display()"<<endl;
         cout<<"var_a = "<<var_a;
      }
};

class BaseClass_2{
   public:
      int var_a = 20;
      void display(){
         cout<<"I am in BaseClass_2 display()"<<endl;
         cout<<"var_a = "<<var_a;
      }
};
class DerivedClass:public BaseClass_1, public BaseClass_2{
   public:
      int var_a = 30;
      void display(){
         cout<<"I am in DerivedClass display()"<<endl<<endl;
         cout<<"BaseClass_1 var_a = "<<BaseClass_1::var_a<<endl;
         cout<<"BaseClass_2 var_a = "<<BaseClass_2::var_a<<endl;
         cout<<"DerivedClass var_a = "<<var_a<<endl;
```

```
    }
};
int main()
{
    DerivedClass obj;
    obj.display();
    return 0;
}
```

OUTPUT:



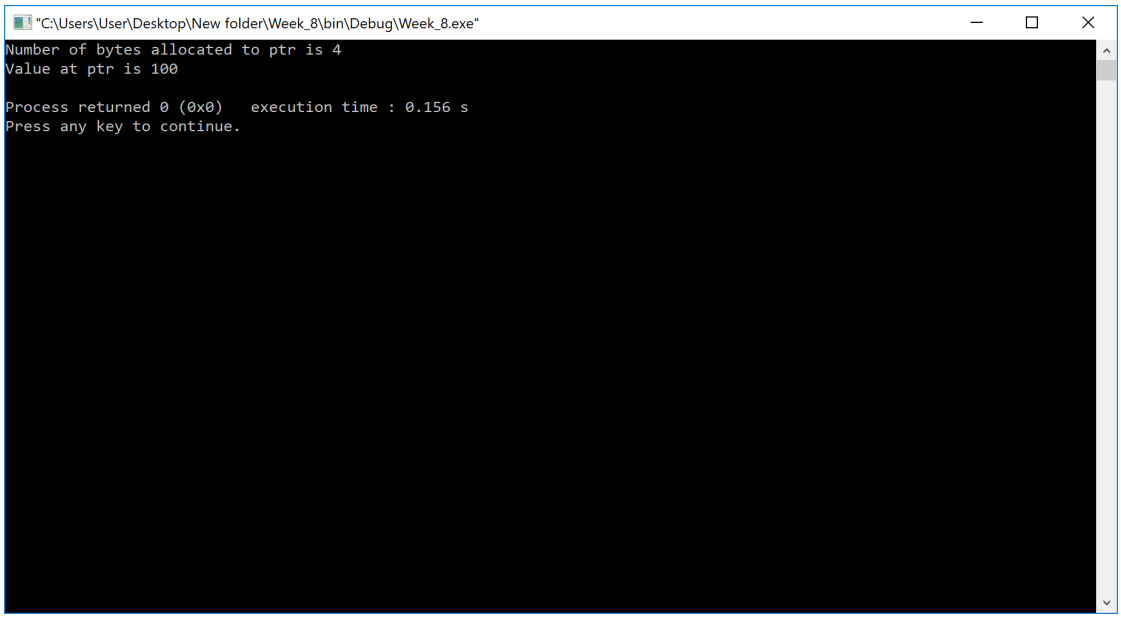
EXPERIMENT-8

AIM: Write a C++ program to allocate memory using new operator

PROGRAM:

```
#include <iostream>
using namespace std;
int main()
{    int *ptr;
   ptr = new int;  // Dynamic memory allocation
   cout<< "Number of bytes allocated to ptr is " << sizeof(ptr) << endl;
   *ptr = 100;
   cout << "Value at ptr is " << *ptr << endl;
   return 0;
}
```

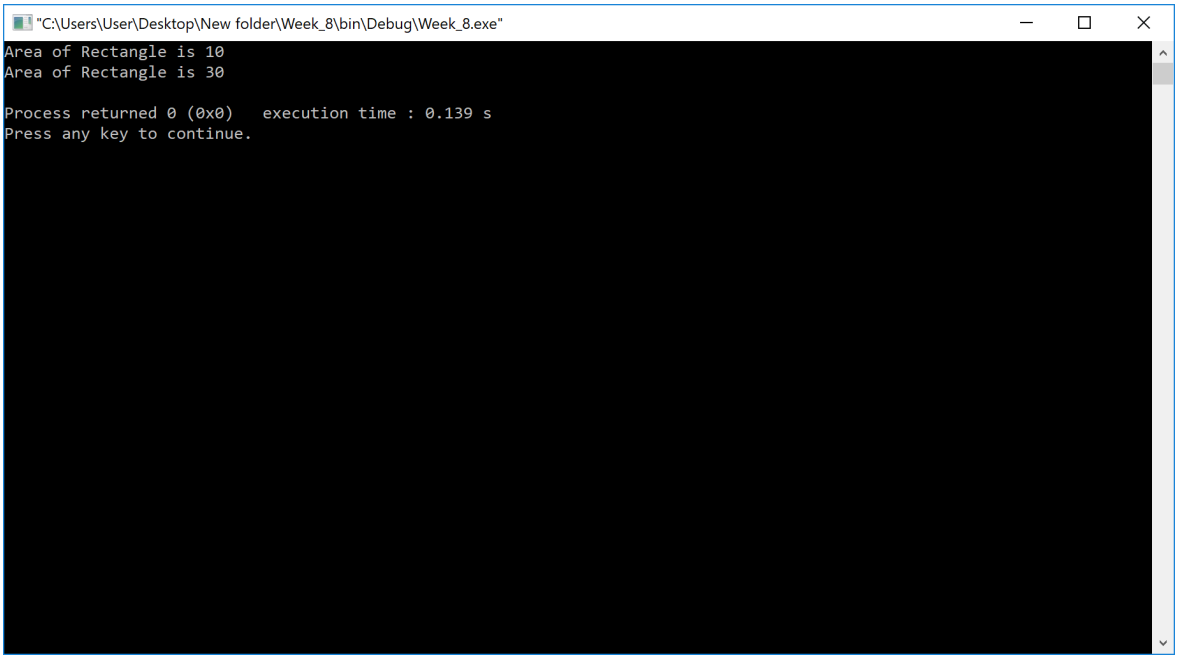
OUTPUT:



// new with user-defined data type variable (Objects)

```
#include <iostream>
using namespace std;
class Rectangle{
   int length, width;
   public:
      Rectangle(){
         length = 2;
         width = 5;
      }
      Rectangle(int l, int w){
         length = l;
         width = w;
      }
      void area(){
         cout<< "Area of Rectangle is " << length * width << endl;
      }
};
int main()
{
   Rectangle *obj_1, *obj_2;
   obj_1 = new Rectangle();  // Dynamic memory allocation
   obj_2 = new Rectangle(3, 10);  // Dynamic memory allocation
   obj_1->area();
   obj_2->area();
      return 0;
}
```

OUTPUT:


"C:\Users\User\Desktop\New folder\Week_8\bin\Debug\Week_8.exe"
Area of Rectangle is 10
Area of Rectangle is 30
Process returned 0 (0x0) execution time : 0.139 s
Press any key to continue.

EXPERIMENT-9

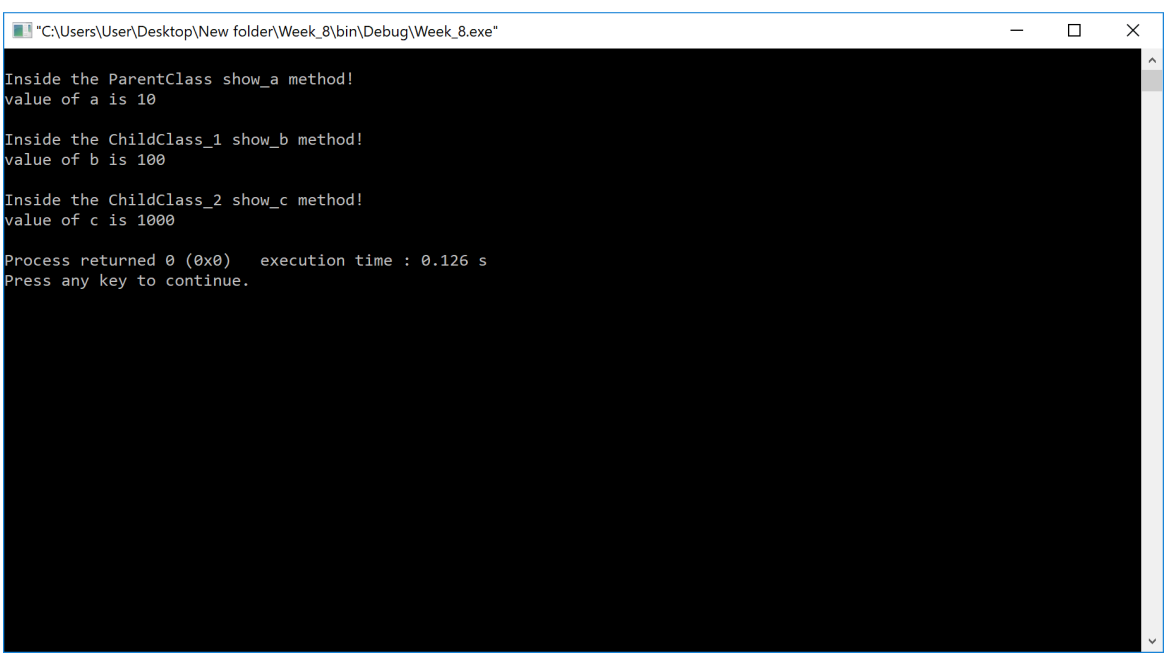
AIM: Write a C++ program to create multilevel inheritance. (Hint: Classes A1, A2, A3)

.PROGRAM:

```
#include <iostream>
using namespace std;

class  ParentClass{
    int a;
    public:
        ParentClass(){
            a = 10;
        }
        void show_a(){
            cout<< endl << "Inside the ParentClass show_a method!" << endl;
            cout<< "value of a is " << a << endl;
        }};
class  ChildClass_1:public ParentClass{
    int b;
    public:
        ChildClass_1(){
            b = 100;
        }
        void show_b(){
            cout<< endl << "Inside the ChildClass_1 show_b method!" << endl;
            cout<< "value of b is " << b << endl;
        }
};
class  ChildClass_2:public ChildClass_1{
    int c;
```

```
    public:
        ChildClass_2(){
            c = 1000;
        }
        void show_c(){
            cout<< endl << "Inside the ChildClass_2 show_c method!" << endl;
            cout<< "value of c is " << c << endl;
        }};
int main()
{
    ChildClass_2 obj;
    obj.show_a();
    obj.show_b();
    obj.show_c();
    return 0;
}
```

OUTPUT:



EXPERIMENT-10

AIM: Write a C++ program to create an array of pointers. Invoke functions using array objects.

PROCEDURE:

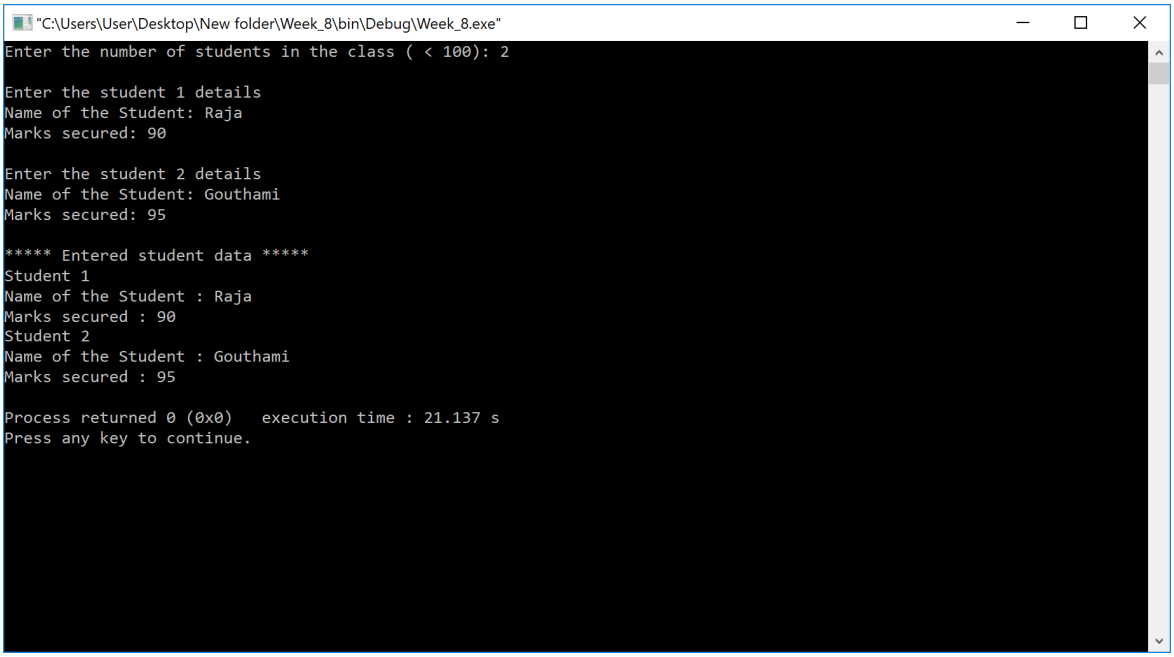
- **Step 1 -** Include the required header files (iostream.h and conio.h).
- **Step 2 -** Create a class (Student) with the following class members as public members. student name, marks as data members. getStudentInfo() and displayStudentInfo() as member functions.
- **Step 3 -** Implement the getStudentInfo() and displayStudentInfo() member functions.
- **Step 4 -** Create a main() method.
- **Step 5 -** Create an array of stud object wigth max size and a pointer object of Student class.
- **Step 6 -** Assign base address of object array to pointer object and make function calls to getStudentInfo() and displayStudentInfo() using class pointer object.

PROGRAM:

```
#include <iostream>
using namespace std;
#define max 100
class Student
{
   string stud_name;
   int marks;
   public:
      void getStudentInfo(int i)
      {
         cout<< endl << "Enter the student " << i << " details" << endl;
         cout<< "Name of the Student: ";
         cin>> stud_name;
         cout<< "Marks secured: ";
         cin>> marks;
      }
```

```
    void displayStudentInfo()
    {
        cout << "Name of the Student : " << stud_name << endl;
        cout << "Marks secured : " << marks << endl;
    }
};
int main()
{
    Student stud[max],*ptr;
    int class_size;
    ptr=stud;
    cout<< "Enter the number of students in the class ( < " << max << "): ";
    cin>> class_size;
    for( int i=1; i<=class_size; i++ )
    {
        (ptr+i)->getStudentInfo(i);
    }
    cout<< endl << "***** Entered student data *****" << endl;
    for( int i=1; i<=class_size; i++ )
    {
        cout << "Student " << i << endl;
        (ptr+i)->displayStudentInfo();
    }
    return 0;
}
```

OUTPUT:


"C:\Users\User\Desktop\New folder\Week_8\bin\Debug\Week_8.exe"
Enter the number of students in the class (< 100): 2
Enter the student 1 details
Name of the Student: Raja
Marks secured: 90
Enter the student 2 details
Name of the Student: Gouthami
Marks secured: 95
***** Entered student data *****
Student 1
Name of the Student : Raja
Marks secured : 90
Student 2
Name of the Student : Gouthami
Marks secured : 95
Process returned 0 (0x0) execution time : 21.137 s
Press any key to continue.

EXPERIMENT-11

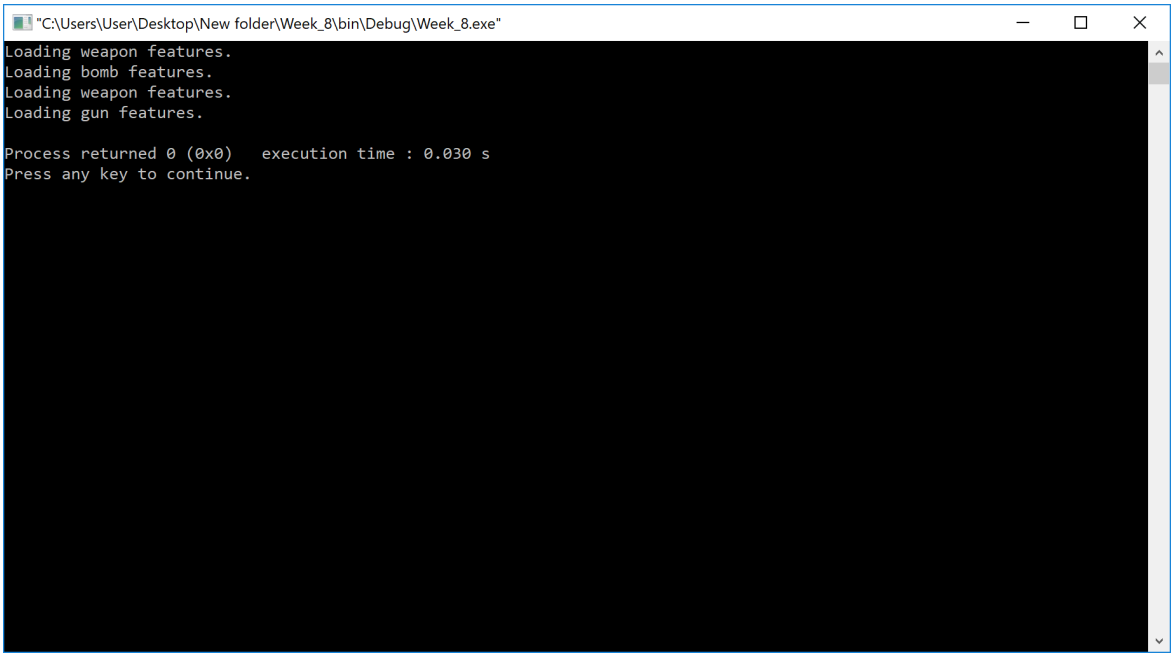
AIM: Write a C++ program to use pointer for both base and derived classes and call the member function. Use Virtual keyword.

PROGRAM:

```
#include <iostream>
using namespace std;

class Weapon
{
   public:
   virtual void features()
   {
      cout << "Loading weapon features.\n";
   }
};
class Bomb : public Weapon
{
   public:
   void features()
   {
      this->Weapon::features();
      cout << "Loading bomb features.\n";
   }
};
class Gun : public Weapon
{
   public:
   void features()
   {
      this->Weapon::features();
```

```
        cout << "Loading gun features.\n";
    }
};
class Loader
{
    public:
    void loadFeatures(Weapon *weapon)
    {
        weapon->features();
    }
};
int main()
{
    Loader *l = new Loader;
    Weapon *w;
    Bomb b;
    Gun g;
    w = &b;
    l->loadFeatures(w);
    w = &g;
    l->loadFeatures(w);
    return 0;
}
```

OUTPUT:


"C:\Users\User\Desktop\New folder\Week_8\bin\Debug\Week_8.exe"
Loading weapon features.
Loading bomb features.
Loading weapon features.
Loading gun features.
Process returned 0 (0x0) execution time : 0.030 s
Press any key to continue.

ADDITIONAL PROGRAMS

1.Write a c++ program for Inline function.

```
#include <iostream>
using namespace std;
class operation
{
        int a,b,add,sub,mul;
        float div;
public:
        void get();
        void sum();
        void difference();
        void product();
        void division();
};
inline void operation :: get()
{
        cout << "Enter first value:";
        cin >> a;
        cout << "Enter second value:";
        cin >> b;
}

inline void operation :: sum()
{
        add = a+b;
        cout << "Addition of two numbers: " << a+b << "\n";
}

inline void operation :: difference()
{
        sub = a-b;
        cout << "Difference of two numbers: " << a-b << "\n";
}

inline void operation :: product()
{
        mul = a*b;
        cout << "Product of two numbers: " << a*b << "\n";
}

inline void operation ::division()
{
```

```
        div=a/b;
        cout<<"Division of two numbers: "<<a/b<<"\n" ;
}

int main()
{
        cout << "Program using inline function\n";
        operation s;
        s.get();
        s.sum();
        s.difference();
        s.product();
        s.division();
        return 0;
}
```

OUTPUT:

```
Enter first value: 45
Enter second value: 15
Addition of two numbers: 60
Difference of two numbers: 30
Product of two numbers: 675
Division of two numbers: 3
```

2. Write a C++ program on Single level inheritance

```
#include <iostream>

using namespace std;

class base    //single base class

{

  public:

    int x;

  void getdata()

  {

    cout << "Enter the value of x = "; cin >> x;

  }
```

```
 };
class derive : public base    //single derived class
{
   private:
    int y;
   public:
   void readdata()
   {
     cout << "Enter the value of y = "; cin >> y;
   }
   void product()
   {
     cout << "Product = " << x * y;
   }
 };
int main()
 {
    derive a;     //object of derived class
    a.getdata();
    a.readdata();
    a.product();
    return 0;
 }
```

OUTPUT:

```
Enter the value of x = 3
Enter the value of y = 4
Product = 12
```

3. Write a c++ program for multiple catch statement

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int a=2;
      try
      {
         if(a==1)
            throw a;                //throwing integer exception
         else if(a==2)
            throw 'A';              //throwing character exception
         else if(a==3)
            throw 4.5;              //throwing float exception
      }
      catch(int a)
      {
         cout<<"\nInteger exception caught.";
      }
      catch(char ch)
```

```
    {
        cout<<"\nCharacter exception caught.";
    }
    catch(double d)
    {
        cout<<"\nDouble exception caught.";
    }
    cout<<"\nEnd of program.";
}
```

OUTPUT:

```
Character exception caught.
End of program.
```

4. Write a C++ program for user defined manipulators.

```
#include< iostream.h>
#include< iomanip.h>

void main (void)
{
int value;
float a,b,c;
a = 350;
b = 200;
c = a/b;

cout << " Enter number" << endl;
cin >> value;
cout << " Hexadecimal base =" << hex << value << endl;
cout << " Octal base =" << oct << value << endl;
cout << " hexadecimal base = " << setbase (16);
cout << value << endl;
```

```
cout << " Octal base = " << setbase (8) << value << endl;
cout << setfill ('*');
cout << setw (5) << a << setw (5) << b << endl;
cout << setw (6) << a << setw (6) << b << endl;
cout << setw (7) << a << setw (7) << b << endl;

cout << fixed << setprecision (2) << c << endl;

}
```

OUTPUT:

```
Enter number
100
Hexadecimal base =64
Octal base =144
hexadecimal base = 64
Octal base = 144
**350**200
***350***200
****350****200
1.75
```