****

**LAB MANUAL**

**Object Oriented Programming using C++ Lab (CSP-152)**

**Common to all IT Branches of First Year**

**SYLLABUS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CSP-152** | **Object Oriented Programming using C++ Lab** | | **L** | **T** | **P** | **C** |
| Total Contact Hours:60 Hours | | **0** | **0** | **4** | **2** |
| Applicable to IT Branches: Applicable to1st Year students | |
| Prerequisite: Basic C Programming Language Knowledge | | | | | |
| **Marks-100** | | | | | | |
| Internal-60 | | External-40 | | | | |
| **Course Objective** | | | | | | |
| * To enable the students to understand various stages and constructs of C++ programming language and relate them to engineering programming problems. * To improve their ability to analyze and address variety of problems in programming domains. | | | | | | |
| **Course Outcome** | | | | | | |
| 1. It will provide the environment that allows students to understand object-oriented programming concepts. 2. Students will demonstrate basic experimental skills for differentiating between object oriented and procedural programming paradigms and the advantages of object-oriented programs. 3. Ability to demonstrate their coding skill on complex programming concepts and use it for generating solutions for engineering and mathematical problems. 4. Students will develop skills to understand the application of classes, objects, constructors, Destructors, inheritance, operator overloading and polymorphism, pointers, virtual functions, templates, exception handling, file operations and handling. | | | | | | |

**Content of the Syllabus**

|  |  |  |
| --- | --- | --- |
| **Practical #** | **Sub part** | **Aim** |
| **Unit-I** | | |
| 1 | i | WAP to find average marks of five subjects of a student in a class. |
| ii | WAP to swap first and last digits of any number. (For ex:-n=12345, Output:- 52341). |
|  | iii | WAP to generate the Fibonacci series up to user specified limit. Write all the missing terms (e.g. 4, 6, 7, 9, 10, 11, 12, 14, 15…) also at the end. |
| 2 | iv | WAP to input a matrix of dimension m\*n. If base address is 1000. Find the address of (m-1, n-1) element of the matrix. |
| v | Create a class called employee with the following details as variables within it. 1. Name of the employee (string) 2. Age (int) 3. Designation (string) 4. Salary (double) Write a program to create array of objects for the same to access these. Also, make use of member functions to accept values and print the name, age, designation and salary. |
| vi | WAP to illustrate the use of scope resolution operator. Display the various values of the same variables declared at different scope levels. |
| 3 | vii | Write a program to find the largest& smallest of three numbers. (Use inline function MAX and MIN). |
| viii | A dining hall can accommodate only 50 guests. Create a class to store seat number (Generated Automatically) and name of the guests who are seated on first come first seated basis. Define functions to display name of all guests along with seat number. Write a program to show the working of this class using the concept of static data member and static function. |
| ix | WAP to swap private data members of classes named as class\_1, class\_2 using friend function. |
| x | WAP to create a class complex to represent complex numbers. The complex class should use a function to add two complex numbers which are passed as arguments. The function should return an object of type complex representing the sum of two complex numbers. |
| 4 | xi | WAP to find area of rectangle using constructor overloading. Also define destructor to delete the memory allocated to objects. |
| xii | WAP to create database of the following items: Name of the student (String), Roll number of the student (int), Height of the student (cm), Weight of the student (kg/gms) 1) Create a Constructor to initialize values 2) Create display () function to display the details 3) Illustrate the use of copy constructor 4) Also implement the concept of destructor. |
| **Unit-II** | | |
| 5 | xiii | Write a program that takes information about institute staff information for 1) Teacher code, name, subject and publication 2) Officer code, name and grade 3) Typist code, name, speed and daily wages and displays it using multiple inheritance. |
| xiv | Create a class student having student uid and getnumber(),putnumber() as member functions to get the values and display it. Derive a class test having marks in different subjects and getmarks() and putmarks() as member functions to get and display the values. Derive another class sports from student class having sports score and getscore(), putscore() as member functions to get and display the values. Derive a class result from test and sports class and define a function display() to calculate total marks. Implement it with the object of result class. If it gives any error, resolve it by adding the required functionality. |
| xv | WAP to illustrate how the constructors are implemented and the order in which they are called when the classes are inherited. Use three classes named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha &beta. Pass four variable to gamma class object which will further send one integer variable to alpha(),one float type variable to beta().Show the order of execution by invoking constructor of derived class. |
| 6 | xvi | WAP to calculate and display cube of an integer and float variable using function overloading |
| xvii | Program to demonstrate the unary operator overloading for operator ++. Make a class test. Create a default constructor to initialize the variable. 1) Overload operator ++ (Pre) with definition to pre-decrement the value of a variable 2) Overload operator ++ (post) with definition to post-decrement the value of variable. |
| xviii | WAP for creating a matrix class which can handle integer matrices of different dimensions. Overload the operator (+) for addition and (==) comparison of matrices. |
| xix | WAP to create a class Pairs. Objects of type Pairs can be used in any situation where ordered pairs are needed. Our Task is to overload operator >> and << so that objects of class Pairs are to be input and output in the form (5,3) (5,-6) (-5,6) or (-5,-3).There is no need to implement any constructor/method . |
| 7 | xx | WAP to create a class that will maintain the records of person with details (Name and Age) and find the eldest among them. The program must use this pointer to return the result by overloading> operator among two objects. |
| xxi | WAP to access members as mentioned in practical 2.2 using pointer to object members. |
| xxii | WAP to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media).The class should have the functionality for adding new item, issuing, deposit etc. The program should link the objects with concerned function by the concept of runtime polymorphism. |
| **Unit-III** | | |
| 8 | xxiii | WAP to perform exception handling for Divide by zero Exception. |
| xxiv | WAP to implement the exception handling with the functionality of testing the throw restrictions. |
| 9 | xxv | WAP to calculate sum of marks of n students of a class inputted via dynamic memory allocation. |
| xxvi | WAP to allocate memory dynamically for an object of a given class using class’s constructor. |
| 10 | xxvii | WAP to copy the contents of one file to another and display it on output screen. |
| xxviii | WAP to read the class object of student info such as name, age and rollno from the keyboard and to store them on a specified file using read() and write() functions. Again the same file is opened for reading and displaying the contents of the file on the screen. |

**Program No. 1 (i)**

**Aim:-** WAP to find average marks of five subjects of a student in a class.

**ProgramCode:-**

#include <iostream>

using namespace std;

     int main(){

    int subjects, i;

    float marks, total=0.0f, averageMarks, percentage;

   // Input number of subjects

    cout << "Enter number of subjects\n";

    cin >> subjects;

        // Take marks of subjects as input

    cout << "Enter marks of subjects\n";

        for(i = 0; i < subjects; i++){

       cin >> marks;

       total += marks;

    }

         // Calculate Average

    averageMarks = total / subjects;

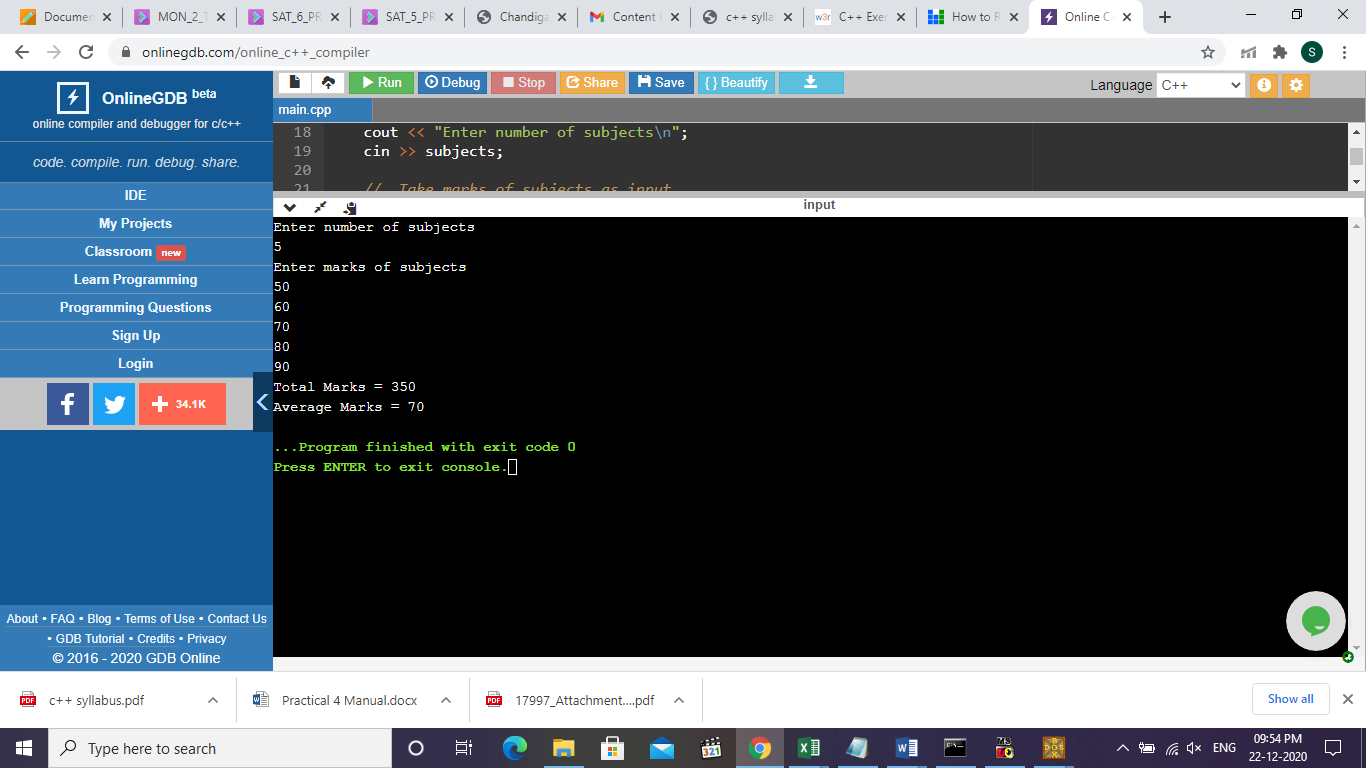
    cout << "Total Marks = "<< total;

    cout << "\nAverage Marks = "<< averageMarks;

        return 0;

}

**Output:**



**Program No. 1 (ii)**

**Aim:-**WAP to swap first and last digits of any number. (For ex:-n=12345, Output:-52341)  

**Program Code:-**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int n, first, last, sum, digits, nn, a, b;

   cout <<"\n\n Find the number after swapping the first and last digits:\n";

   cout <<"-------------------------------------------------------------\n";

   cout <<" Input any number: ";

   cin >> n;

   digits =(int)log10(n);

   first = n /pow(10, digits);

   last = n %10;

   a = first \*(pow(10, digits));

   b = n % a;

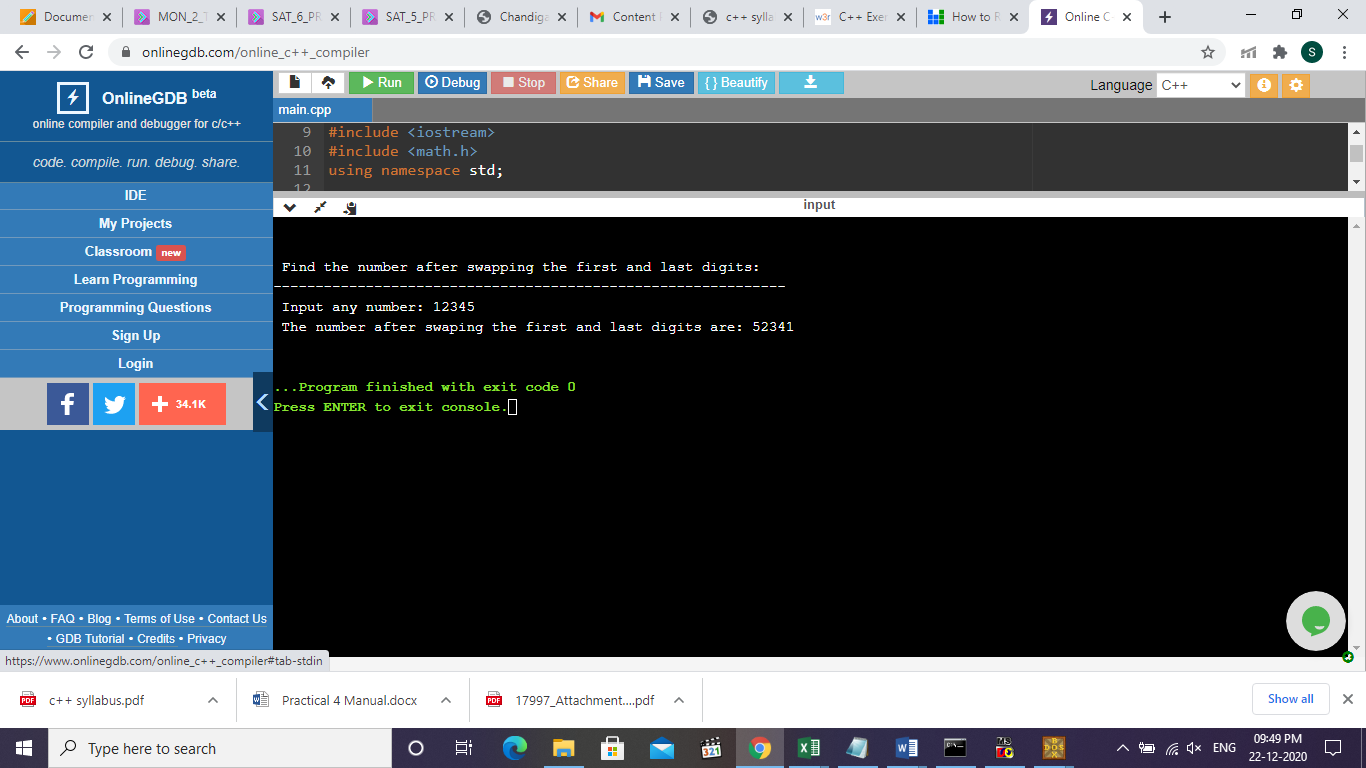
   n = b /10;

   nn = last \*(pow(10, digits))+(n \*10+ first);

   cout <<" The number after swaping the first and last digits are: "<< nn << endl;

}

**Output:-**



**Program No. 1 (iii)**

**Aim:-** WAP to generate the Fibonacci series up to user specified limit. Write all the missing terms (e.g. 4, 6, 7, 9, 10, 11, 12, 14, 15…) also at the end.

**Program Code:-**

#include<iostream>

using namespace std;

int main()

{

int n,c,first=0,second=1,next;

int a[20],i,j=0,count=0;

cout<<"Enter the no. of terms of Fibonacci series=";

cin>>n;

cout<<"Terms of Fibonacci series are"<<endl;

for(c=0;c<n;c++)

{

if(c<=1)

next=c;

else

{

next=first+second;

first=second;

second=next;

}

cout<<next<<endl;

if(next-first>1)

{

for(i=first+1; i<next; i++)

{

a[j]=i;

count++;

j++;

}

}

}

cout<<"Missing numbers of the Fibonacci series are:"<<endl;

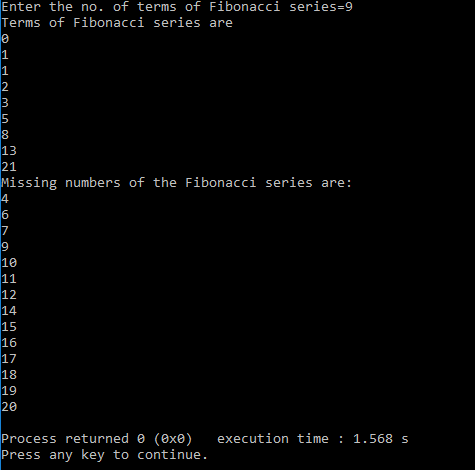
for(j=0; j<count; j++)

cout<<a[j]<<endl;

return 0;

}

**Output:-**



**Program No. 2 (i)**

**Aim:-** WAP to input a matrix of dimension m\*n. If base address is 1000. Find the address of (m-1, n-1) element of the matrix.

**Program Code:-**

#include<iostream>

using namespace std;

  int main()

      {

             int b,i,j,w,lr=0,lc=0,n,m;

             int a[10][10];

             cout<<"enter the no. of rows in matrix";

             cin>>m;

             cout<<"enter no. of columns in matrix";

             cin>>n;

             cout<<"enter the elements in matrix";

             for(i=0;i<m;i++)

             {

                      for(j=0;j<n;j++)

                      {

                                  cin>>a[i][j];

                                                }

                                    }

                                    cout<<"enter the base address";

                                    cin>>b;

                                    cout<<"enter the storage size of one element stored in array";

                                    cin>>w;

                                    i=m-1;

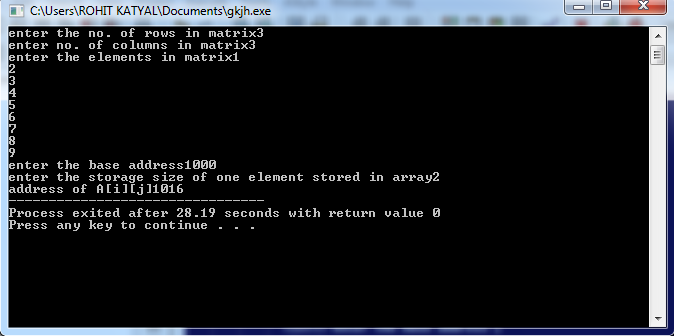
                                    j=n-1;

                                    cout<<"address of A[i][j]"<<b + w\*(n\*(i-lr)+(j-lc));

                                             return 0;

}

**Output:-**

**Program No. 2 (ii)**

**Aim: -** Create a class called employee with the following details as variables within it.

1. Name of the employee (string)

2. Age (int)

3. Designation (string)

4. Salary (double)

Write a program to create array of objects for the same to access these. Also, make use of member functions to accept values and print the name, age, designation and salary.  

**Solution:**

#include<iostream>

using namespace std;

      class Employee

      {

             char Name[25];

             int Age;

             char Desg[8];

             long Salary;

             public:

             void GetData();

             void PutData();

      };

      void Employee :: GetData()          //Statement 1 : Defining GetData()

      {

              cout<<"\n\tEnter Employee Name : ";

             cin>>Name;

             cout<<"\n\tEnter Employee Age : ";

             cin>>Age;

                     cout<<"\n\tEnter Employee Designation:";

             cin>>Desg;

             cout<<"\n\tEnter Employee Salary : ";

             cin>>Salary;

      }

      void Employee :: PutData()          //Statement 2 : Defining PutData()

      {

             cout<<"\nEmployee Name : "<<Name;

             cout<<"\nEmployee Age : "<<Age;

             cout<<"\nEmployee Designation:"<<Desg;

             cout<<"\nEmployee Salary : "<<Salary;

      }

      int main()

      {

             Employee E[5];          //Statement 3 : Creating Object

             int i;

             for(i=0;i<5;i++)

             {

               E[i].GetData();         //Statement 4 : Calling GetData()

             }

             for(i=0;i<5;i++)

             {

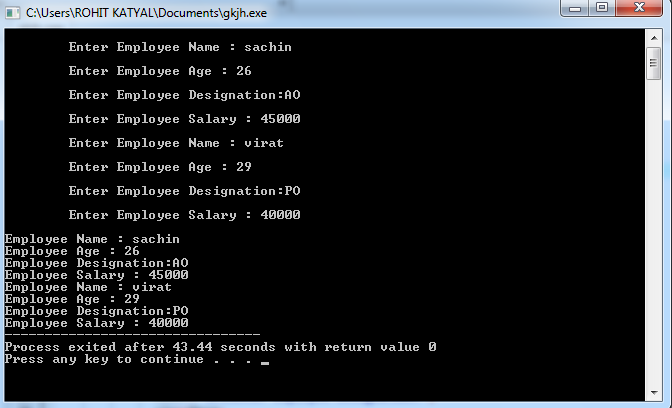
               E[i].PutData();         //Statement 5 : Calling PutData()

             }

             return 0;

      }

**Output:-**

****

**Program No. 2 (iii)**

**Aim: -** WAP to illustrate the use of scope resolution operator. Display the various values of the same variables declared at different scope levels.

**Solution:**

#include<iostream>

using namespace std;

int my\_variable = 10; // Global x

int main()

{

 int my\_variable = 100; // Local x

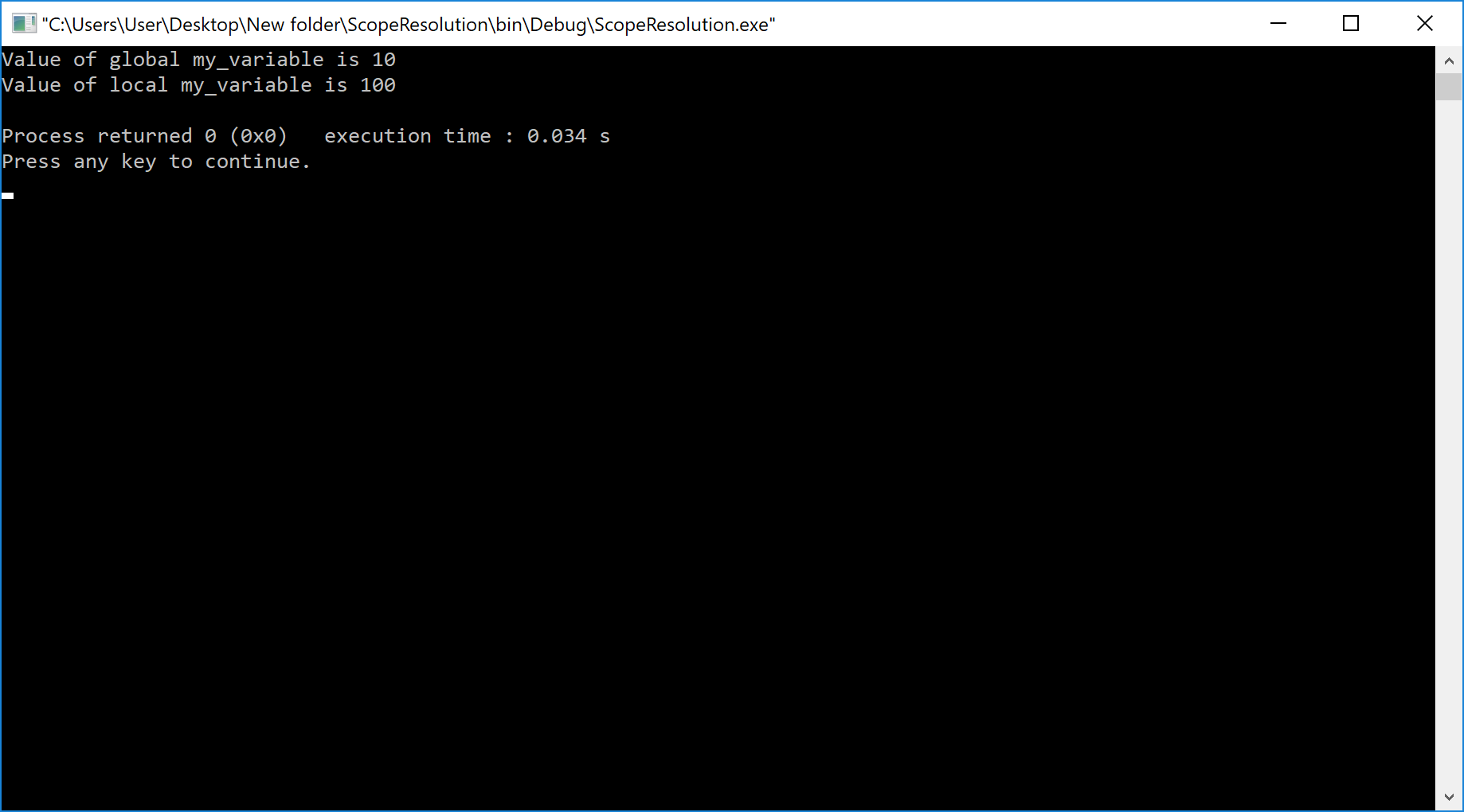
 cout << "Value of global my\_variable is " << ::my\_variable<<endl;

 cout << "Value of local my\_variable is " << my\_variable<<endl;

 return 0;

}

**Output:**

****

**Program No. 3 (i)**

**Aim:-** Write a program to find the largest& smallest of three numbers. (Use inline function MAX and MIN)

**Solution:**

#include <stdio.h>

// macro/inline function to get max of 3 numbers

#define MAX(a,b,c) (a > b && a > c ? a : (b > c ? b : c))

// macro/inline function to get min of 3 numbers

#define MIN(a,b,c) (a < b && a < c ? a : (b < c ? b : c))

int main()

{

int x, y, z, large, small;

   // accept 3 numbers from console

printf("Enter 3 numbers: ");

scanf("%d%d%d", &x, &y, &z);

  // call inline function to get the max and min of inputed numbers

large = MAX(x, y, z);

small = MIN(x, y, z);

   // print the largest and smallest numbers

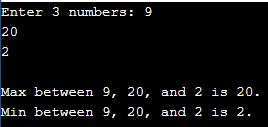
printf("\nMax between %d, %d, and %d is %d.", x, y, z, large);

printf("\nMin between %d, %d, and %d is %d.", x, y, z, small);

return 0;

}

**Output:**



**Program No. 3 (ii)**

**Aim:-** A dining hall can accommodate only 50 guests. Create a class to store seat number (Generated Automatically) and name of the guests who are seated on first come first seated basis. Define functions to display name of all guests along with seat number. Write a program to show the working of this class using the concept of static data member and static function

**Solution:**

#include <iostream>

#define MAX\_SIZE 50

using namespace std;

// definition of guest class as we are required guest name and seat number

class Guest {

public:

char name[50];

int seatno;

};

// defination of Hall class having function to allot and list guest

class Hall {

public:

       // static member data

staticint front, rear;

static Guest allGuest[MAX\_SIZE];

       // static member function for alloting the seat to the guest in FIFO order

staticintalloteSeat() {

if (rear == (MAX\_SIZE - 1)) {

cout<< "Hall is full!";

return 0;

           }

rear++;

cout<< "Enter Guest Name: ";

cin>>allGuest[rear].name;

allGuest[rear].seatno = rear + 1;

return 1;

       }

       // static member function to list the guests with name nad seat number

static void listGuest() {

while(++front <= rear) {

cout<< "\nGuest " <<allGuest[front].name << " is seated on seat S" <<allGuest[front].seatno<< ".";

           }

rear = front = -1;

       }

};

// initlizing the static member data

int Hall::front = -1;

int Hall::rear = -1;

Guest Hall::allGuest[MAX\_SIZE] = {};

int main()

{

   // alloting the seat to the guest

   Hall::alloteSeat();

   Hall::alloteSeat();

   Hall::alloteSeat();

   Hall::alloteSeat();

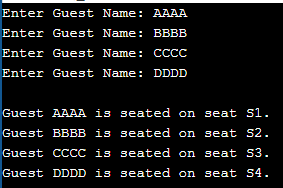
   // listing the guest with seat number

   Hall::listGuest();

return 0;

}

**Output:-**

****

**Program No. 3 (iii)**

**Aim: -** **WAP to swap private data members of classes named as class\_1, class\_2 using friend function.**

#### **Solution:**

#include<iostream>

using namespace std;

class class\_2;

// defining class 1 having friend function swap

class class\_1

{

   // member data

protected:

int num1;

public:

       class\_1()

       {

           num1=10;

       }

       // member function to show the value of member data

void show()

       {

cout<<"\n Value of Class 1 : "<<num1;

       }

       // friend function declaration

friend void swap(class\_1 \*num1, class\_2 \*num2);

};

// defining class 2 having friend function swap

class class\_2

{

   // member data

protected:

int num2;

public:

       class\_2()

       {

           num2=20;

       }

       // member function to show the value of member data

void show()

       {

cout<<"\n Value of Class 2 : "<<num2;

       }

       // friend function declaration

friend void swap(class\_1 \*num1, class\_2 \*num2);

};

// definition of swap friend function

void swap(class\_1 \*no1, class\_2 \*no2)

{

int no3;

   no3=no1->num1;

   no1->num1=no2->num2;

   no2->num2=no3;

}

int main()

{

class\_1 a;

class\_2 b;

cout<< "Values before Swap";

a.show();

b.show();

swap(&a, &b);

cout<< "\n\nValues after Swap";

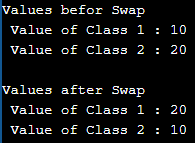
a.show();

b.show();

return 0;

}

**Output:**



**Program No. 3 (iv)**

**Aim: -** WAP to create a class complex to represent complex numbers. The complex class should use a function to add two complex numbers which are passed as arguments. The function should return an object of type complex representing the sum of two complex numbers.

**Solution:**

#include<iostream>

using namespace std;

// defining a class to represent a complex number

class complex

{

   // member data to store the complex number parts

private:

       // real part

               float r;

                  // imaginary part

               float i;

public:

       // set the values

               void set(float real, float img)

               {

                               r = real;

                               i = img;

               }

       // member function to sum the self and one another complex number

               complex sum(complex c)

       {

               complex t;

               t.r = r + c.r;

               t.i = i + c.i;

               return t;

       }

       // function to print the complex number

               void disp()

                  {

                      // since the imaginary number multiplicant of some real number

                      // and thus when 1 is multiplied with any number will remain same

               if (i == -1) {

               cout<< r << " + -i" <<endl;

                      }

               else if (i == 1) {

               cout<< r << " + i" <<endl;

                      }

               else if (i == 0) {

                          // since imaginary part is zero so only real part will be available

               cout<< r <<endl;

                      }

               else {

               cout<< r << " + " <<i<< "i" <<endl;

                      }

       }

};

int main()

{

               complex c1, c2, c3;

               c1.set(2.5, 3.5);

               c2.set(1.5, 5.5);

              c3 = c1.sum(c2);

               cout<<"Complex Number 1 = ";

               c1.disp();

               cout<<"Complex Number 2 = ";

               c2.disp();

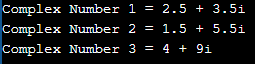
               cout<<"Complex Number 3 = ";

               c3.disp();

               return 0;

}

**Output:**



**Program No. 4 (i)**

**Aim:-** WAP to find area of rectangle using constructor overloading. Also define destructor to delete the memory allocated to objects.

#### **Solution:**

#include<iostream>

using namespace std;

class area

{

  int a,l,b;

  public:

  area() // simple constructor definition.

  {

     l=5;

     b=6;

     cout<<"Simple constructor called\n";

     cout<<"length="<<l<<"\nbreadth="<<b<<endl;

  }

  area(int x,int y) // parameterised constructor

  {

     l=x;

     b=y;

  }

  void calc();

  void print();

  ~area();

};

void area::calc()

{

   a=l\*b;

}

void area::print()

{

   cout<<"Area is : "<<a<<endl;

}

area::~area()

{

cout << "Object is being deleted" << endl;

}

int main()

{

   int l,b;

   area a1; // simple constructor is called.

   a1.calc();

   a1.print();

   cout<<"Enter length and breadth for parameterised        constructor:\n";

   cin>>l>>b;

   area a2(l,b); // parameterised constructor is called.

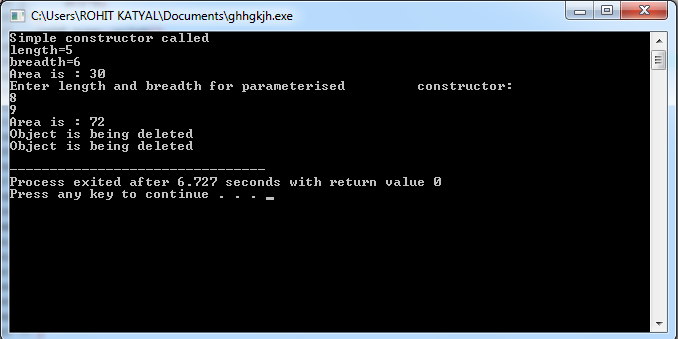
   a2.calc();

   a2.print();

   return 0;

}

**Output:**



**Program No. 4 (ii)**

**Aim: -** WAP to create database of the following items: Name of the student (String), Roll number of the student (int), Height of the student (cm), Weight of the student (kg/gms)

1) Create a Constructor to initialize values

2) Create display () function to display the details

3) Illustrate the use of copy constructor

4) Also implement the concept of destructor.

**Solution:**

#include <iostream>

using namespace std;

class student

{

           private:

                       string name;

                       int  rollNo;

                       int  height;

                       int weight;

           public:

                       student(string n,int r,int h,int w)

               {

                   name=n;

                   rollNo=r;

                   height=h;

                   weight=w;

                }

                       //member function to print student's details

                       void display(void);

                       ~student();

};

void student::display(void){

           cout << "Student details:\n";

           cout << "Name:"<< name << ",Roll Number:" << rollNo << ",Height:" << height << ",Weight:" << weight;

}

student :: ~student()

{

           cout<<"destructor is called\n";

}

int main()

{

           student std("mohit",1234,6,78);                    //object creation

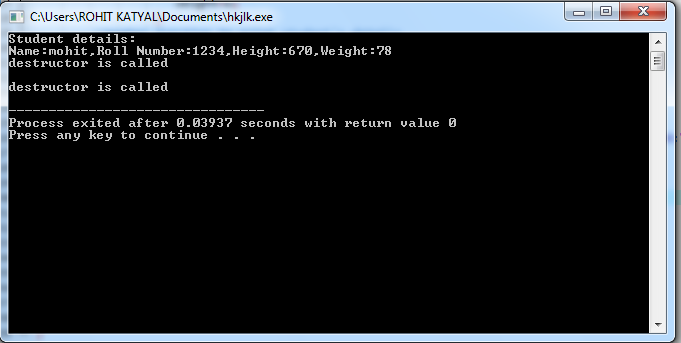
           student std1=std;

           std1.display ();

            return 0;

}

**Output:**



**Program No. 5 (i)**

**AIM: -**Write a program that takes information about institute staff information for

1) Teacher code, name, subject and publication

2) Officer code, name and grade

3) Typist code, name, speed and daily wages and displays it using hierarchal inheritance

**Program Code:-**

#include <iostream>

#include <conio.h>

using namespace std;

class staff

{

 protected:

     int code;

     char name[20];

 public:

     void getstaff(void)

     {

   cout<<"\n\nEnter code :-";

   cin>>code;

   cout<<"Enter name :-";

   cin>>name;

     }

     void dispstaff(void)

     {

      cout<<"\nNAME      :-"<<name;

      cout<<"\nCODE      :-"<<code;

     }

};

class teacher : public staff

{

     char sub[20];

     char pub[20];

 public:

     void create(void)

     {

   getstaff();

   cout<<"Enter Subject :-";

   cin>>sub;

   cout<<"Enter Publication :-";

   cin>>pub;

     }

     void display(void)

     {

    dispstaff();

    cout<<"\nSUBJECT   :-"<<sub;

    cout<<"\nPUBLICATION:-"<<pub;

     }

};

class officer : public staff

{

      char grade;

  public:

   void create(void)

   {

      getstaff();

      cout<<"Enter Grade :-";

      cin>>grade;

   }

   void display(void)

   {

      dispstaff();

      cout<<"\nGRADE     :-"<<grade;

   }

};

class typist : public staff

{

      float speed;

 public:

     void gettypist(void)

     {

   getstaff();

   cout<<"Enter speed (wpm):-";

   cin>>speed;

     }

     void disptypist(void)

     {

   dispstaff();

   cout<<"\nSPEED     :-"<<speed;

     }

};

class casual : public typist

{

    float dailywages;

  public:

      void create(void)

   {

     gettypist();

     cout<<"Enter Daily Wages :-";

     cin>>dailywages;

   }

   void display(void)

   {

     disptypist();

     cout<<"\nDAILY WAGES:-"<<dailywages;

   }

};

int main()

{

teacher o1t[10];

casual o1c[10];

officer o1o[10];

int choice,i;

char test;

while(1)

{

int count;

start:

  cout<<"\n=====EDUCATION INSTITUTION DATABASE=====\n\n\n";

  cout<<"Choose Category of Information\n";

  cout<<"1)  Teachers\n";

  cout<<"2)  Officer\n";

  cout<<"3)  Typist\n";

  cout<<"4)  Exit\n";

  cout<<"Enter your choice:-";

  cin>>choice;

  switch(choice)

  {

     case 1 : while(1)

       {

          cout<<"\n=====TEACHERS INFORMATION=====\n\n";

       cout<<"\nChoose your choice\n";

       cout<<"1) Create\n";

       cout<<"2) Display\n";

       cout<<"3) Jump to Main Menu\n";

       cout<<"Enter your choice:-";

       cin>>choice;

       switch(choice)

       {

        case 1 : for(count=0,i=0;i<10;i++)

              {

                cout<<endl;

                o1t[i].create();

                count++;

                cout<<endl;

                cout<<"\n\nAre you Interested in entering data\n";

                cout<<"Enter y or n:-";

                cin>>test;

                if(test=='y' || test=='Y')

               continue;

                else goto out1;

                }

                out1:

                break;

        case 2 : for(i=0;i<count;i++)

              {

               cout<<endl;

                o1t[i].display();

                cout<<endl;

              }

              getch();

              break;

        case 3 : goto start;

        default: cout<<"\nEnter choice is invalid\ntry again\n\n";

        }

        }

     case 2 :  while(1)

        {

       cout<<"\n=====OFFICERS INFORMATION=====\n\n";

       cout<<"\nChoose your choice\n";

       cout<<"1) Create\n";

       cout<<"2) Display\n";

       cout<<"3) Jump to Main Menu\n";

       cout<<"Enter your choice:-";

       cin>>choice;

       switch(choice)

       {

        case 1 : for(count=0,i=0;i<10;i++)

              {

               cout<<endl;

               o1o[i].create();

                count++;

                cout<<endl;

                cout<<"\n\nAre you Interested in entering data\n";

                cout<<"Enter y or n:-";

                cin>>test;

                if(test=='y' || test=='Y')

               continue;

                else goto out2;

                }

                out2:

                break;

        case 2 : for(i=0;i<count;i++)

              {

               cout<<endl;

                o1o[i].display();

                cout<<endl;

              }

              getch();

              break;

        case 3 : goto start;

        default: cout<<"\nInvalid choice\ntry again\n\n";

        }

        }

     case 3 : while(1)

       {

       cout<<"\n=====TYPIST INFORMATION=====\n\n";

       cout<<"\nChoose your choice\n";

       cout<<"1) Create\n";

       cout<<"2) Display\n";

       cout<<"3) Jump to Main Menu\n";

       cout<<"Enter your choice:-";

       cin>>choice;

       switch(choice)

       {

        case 1 : for(count=0,i=0;i<10;i++)

              {

                cout<<endl;

                o1c[i].create();

                count++;

                cout<<endl;

                cout<<"\n\nAre you Interested in entering data\n";

                cout<<"Enter y or n:-";

                cin>>test;

                if(test=='y' || test=='Y')

               continue;

                else goto out3;

                }

                out3:

                break;

        case 2 : for(i=0;i<count;i++)

              {

                cout<<endl;

                o1c[i].display();

                cout<<endl;

              }

              getch();

              break;

        case 3 : goto start;

        default: cout<<"\nInvalid choice\ntry again\n\n";

        }

        }

     case 4 : goto end;

   }

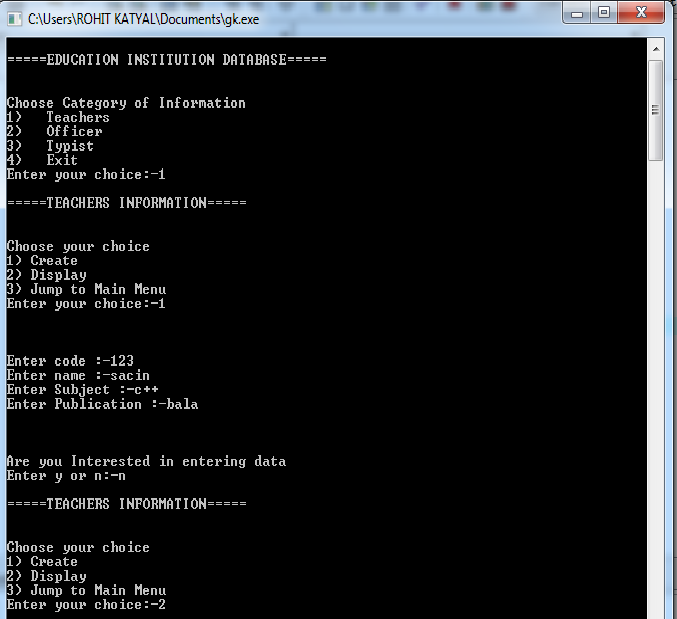
 }

 end:

               return 0;

 }

**Output:**



**Program No. 5 (ii)**

**AIM: -** Create a class student having student uid and getnumber(),putnumber() as member functions to get the values and display it. Derive a class test having marks in different subjects and getmarks() and putmarks() as member functions to get and display the values. Derive another class sports from student class having sports score and getscore(), putscore() as member functions to get and display the values. Derive a class result from test and sports class and define a function display() to calculate total marks. Implement it with the object of result class. If it gives any error, resolve it by adding the required functionality.

**Program Code:-**

#include<iostream>

#include<conio.h>

using namespace std;

class student {

int rno;

public:

void getnumber() {

cout << "Enter Roll No:";

cin>>rno;

}

void putnumber() {

cout << "\n\n\tRoll No:" << rno << "\n";

}

};

class test : virtual public student {

public:

int part1, part2;

void getmarks() {

cout << "Enter Marks\n";

cout << "Part1:";

cin>>part1;

cout << "Part2:";

cin>>part2;

}

void putmarks() {

cout << "\tMarks Obtained\n";

cout << "\n\tPart1:" << part1;

cout << "\n\tPart2:" << part2;

}

};

class sports : public virtual student {

public:

int score;

void getscore() {

cout << "Enter Sports Score:";

cin>>score;

}

void putscore() {

cout << "\n\tSports Score is:" << score;

}

};

class result : public test, public sports {

int total;

public:

void display() {

total = part1 + part2 + score;

putnumber();

putmarks();

putscore();

cout << "\n\tTotal Score:" << total;

}

};

int main() {

result obj;

obj.getnumber();

obj.getmarks();

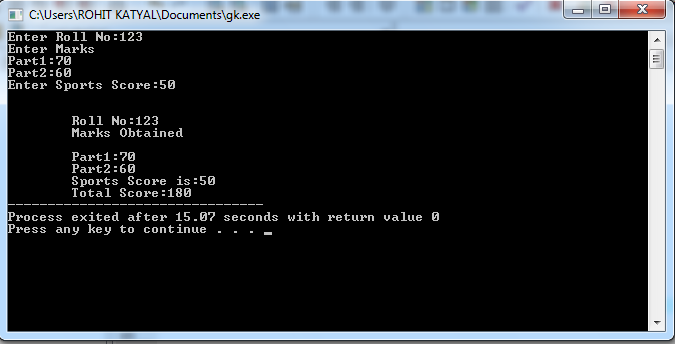
obj.getscore();

obj.display();

return 0;

}

**Output:-**



**Program No. 5 (iii)**

**Aim: -** WAP to illustrate how the constructors are implemented and the order in which they are called when the classes are inherited. Use three classes named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha &beta. Pass four variable to gamma class object which will further send one integer variable to alpha(),one float type variable to beta().Show the order of execution by invoking constructor of derived class.

**Program Code:-**

#include<iostream>

#include<conio.h>

using namespace std;

class alpha

{

int x;

public:

alpha(int i)

{

x=i;

cout<<"alpha initialized\n";

}

void show\_x()

{

cout<<"x="<<x<<"\n";

}

};

class beta

{

float y;

public:

beta(float j)

{

y=j;

cout<<"beta initialized\n";

}

void show\_y()

{

cout<<"y="<<y<<"\n";

}

};

class gamma : public beta,public alpha

{

int m,n;

public:

gamma(int a,float b,int c,int d): alpha(a),beta(b)

{

m=c,n=d;

cout<<"gamma initialized\n";

}

void show\_mn()

{

cout<<"m="<<m<<"\n";

cout<<"n="<<n<<"\n";

}

};

int main()

{

gamma g(5,10.75,20,30);

g.show\_x();

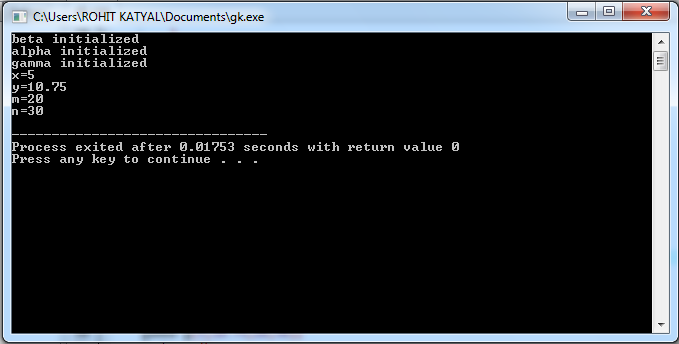
g.show\_y();

g.show\_mn();

return 0;

}

**Output:**



**Program No. 6 (i)**

**AIM: -WAP to calculate and display cube of an integer and float variable using function overloading.**

Operator overloading is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. It is used to perform operation on user-defined data type.

The unary operators operate on a single operand and following are the examples of Unary operators −

* The increment (++) and decrement (--) operators.
* The unary minus (-) operator.
* The logical not (!) operator.

The unary operators operate on the object for which they were called and normally, this operator appears on the left side of the object, as in !obj, -obj, and ++obj but sometime they can be used as postfix as well like obj++ or obj--.

**Program Code:-**

#include <iostream>

using namespace std;

int cube(int );

float cube(float);

int main() {

         int a = 5;

         float b = 5.5;

cout<< "Cube of integer number " << a << " is " << cube(a) <<endl;

cout<< "Cube of float number " << b << " is " << cube(b) <<endl;

         return 0;

}

int cube(int x) {

return x\*x\*x;

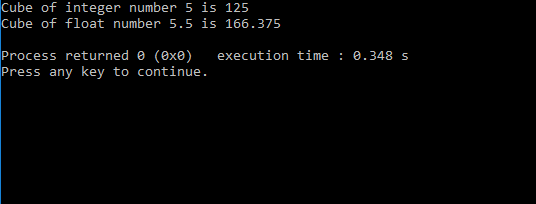
}

float cube(float y){

return y\*y\*y;

}

**Output:-**



**Program No. 6 (ii)**

**Aim: -** **Program to demonstrate the unary operator overloading for operator ++. Make a class test. Create a default constructor to initialize the variable. 1) Overload operator ++ (Pre) with definition to pre-decrement the value of a variable 2) Overload operator ++ (post) with definition to post-decrement the value of variable.**

**Program Code:-**

#include <iostream>

using namespace std;

class Test {

private:

int num;

public:

       // required constructors

       // default constructor to initlize the variable

Test() {

num = 0;

       }

       // parameterized constructor to return object after incrementing

Test(int n) {

num = n;

       }

       // method to display time

void display() {

cout<< "Number: " <<num<<endl;

       }

       // overloaded prefix ++ operator

       Test operator++ () {

           // increment this object

           ++num;

           // return object with increment value

return Test(num);

       }

       // overloaded postfix ++ operator

       Test operator++( int ) {

           // save the orignal value

           Test t(num);

           // increment current object

           ++num;

           // return old original value

return t;

       }

};

int main() {

  Test T1(11), T2(11), T3;

  ++T1;           // increment T1

T1.display ();   // display T1

  T2++;           // increment T2

T2.display ();   // display T2

T3.display ();   // display T3

  T3 = T2++;      // increment T2 again and assign pre-incremented value to T3

T2.display ();   // display T2

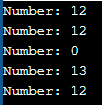
T3.display ();   // display T3

return 0;

}

**Output:-**

After incrementing T1

After incrementing T2

Initial value of T3

After incrementing T2

After assignment of Pre-increment object T2

**Program No. 6 (iii)**

**Aim:-WAP for creating a matrix class which can handle integer matrices of different dimensions. Overload the operator (+) for addition and (==) comparison of matrices.**

**Program Code:-**

#include <iostream>

#define MAXROWS 50

#define MAXCOLS 50

using namespace std;

// Class for Matrix operator overloading

class Matrix {

public:

         // For input Matrix

         intarr[MAXROWS][MAXCOLS];

         int rows, cols;

Matrix() {

rows = cols = 2;

   }

   // Overloaded constructor to initlize the Matrix with dimensions

Matrix(int r, int c, int mat[MAXROWS][MAXCOLS]) {

rows = r;

cols = c;

for (int i = 0; i< rows; i++) {

                  for (int j = 0; j < cols; j++) {

                            arr[i][j] = mat[i][j];

                  }

         }

   }

           // Function to display the elements of Matrix

         void display() {

         for (int i = 0; i< rows; i++) {

         cout<< " [ ";

                  for (int j = 0; j < cols; j++) {

                               // Print the element

                            cout<<arr[i][j] << ", ";

                     }

                  cout<< "]" <<endl;

            }

         cout<<endl;

         }

         // Function for + operator overloading

         Matrix operator+(Matrix x) {

         if (x.rows != rows || x.cols != cols || (rows == 0 && cols == 0)) {

         return Matrix();

            }

        // To store the sum of Matrices

         int mat[MAXROWS][MAXCOLS];

         // Traverse the Matrix x

         for (int i = 0; i< rows; i++) {

                  for (int j = 0; j < cols; j++) {

                            // Add the corresponding blocks of Matrices

                            mat[i][j] = arr[i][j] + x.arr[i][j];

                  }

         }

         return Matrix(rows, cols, mat);

         }

          // Function for == operator overloading

         int operator==(Matrix x) {

         if (x.rows != rows || x.cols != cols) {

         return 0;

            }

            // Travarse the Matrix x

         for (int i = 0; i< rows; i++) {

                  for (int j = 0; j < cols; j++) {

                            // Add the corresponding blocks of Matrices

                            if (arr[i][j] != x.arr[i][j]) {

                            return 0;

                            }

                  }

         }

         return 1;

         }

};

int main()

{

int arr1[MAXROWS][MAXCOLS], arr2[MAXROWS][MAXCOLS];

   // inputing values to array 1

arr1[0][0] = 1;

arr1[0][1] = 2;

arr1[1][0] = 3;

arr1[1][1] = 4;

   // inputing values to array 2

arr2[0][0] = 4;

arr2[0][1] = 3;

arr2[1][0] = 2;

arr2[1][1] = 1;

         // Declare Matrices

         Matrix mat1(2, 2, arr1);

         Matrix mat2(2, 2, arr1);

         Matrix mat3(2, 2, arr2);

         Matrix mat4;

   // Printing the elements of first matrix

cout<< "Elements of Matrix 1:" <<endl;

mat1.display ();

   // Printing the elements of second matrix

cout<< "Elements of Matrix 2:" <<endl;

mat2.display ();

   // Printing the elements of third matrix

cout<< "Elements of Matrix 3:" <<endl;

mat3.display ();

         // Addition of two matrices using operator overloading

  mat4 = mat1 + mat3;

cout<< "Elements of Matrix after addition of Matrix 1 and Matrix 3:" <<endl;

mat4.display ();

         // Equating two matrices using operator overloading

if (mat1 == mat2) {

cout<< "Matrix 1 is equals to Matrix 2" <<endl;

   }

else {

cout<< "Matrix 1 is not equals to Matrix 2" <<endl;

   }

         // Equating two matrices using operator overloading

if (mat1 == mat3) {

cout<< "Matrix 1 is equals to Matrix 3" <<endl;

   }

else {

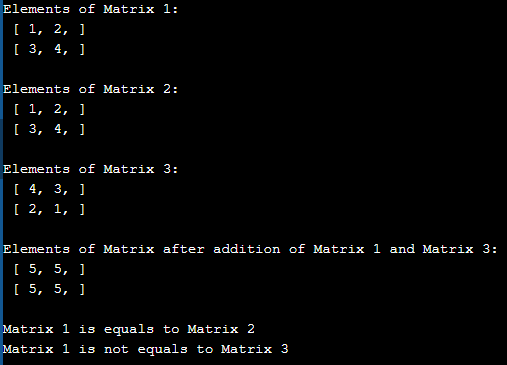
cout<< "Matrix 1 is not equals to Matrix 3" <<endl;

   }

         return 0;

}

**Output:-**

****

**Program No. 6 (iv)**

**Aim:-** WAP to create a class Pairs. Objects of type Pairs can be used in any situation where ordered pairs are needed. Our Task is to overload operator >> and << so that objects of class Pairs are to be input and output in the form (5,3) (5,-6) (-5,6) or (-5,-3).There is no need to implement any constructor/method .

**Program Code:-**

#include <iostream>

// for string functions

#include <cstring>

using namespace std;

class Pairs {

private:

char numpair[20];

public:

       // friend function to overload the output operator

friend ostream&operator<<(ostream&output, const Pairs &p) {

output<<p.numpair;

return output;

       }

       // friend function to overload the input operator

friend istream&operator>>(istream&input, Pairs &p) {

char pair[20];

           // inputting the pair values

input>> pair;

           // getting the length of the inputed string

intlen = strlen(pair);

           // as per given format "(x,y)" the min length of string should be 5 and also

           // the string should contain the first character "(", the last character ")"

           // and a comma in between the string

if (len< 5 || pair[0] != '(' || pair[len - 1] != ')' || !strstr(pair, ",")) {

cout<< "Invalid pair value found!" <<endl;

               // in case of Invalid value assing blank string to the pair value

strcpy(p.numpair, "");

           }

else {

               // in case of valid value copy the input value to the object member datq

strcpy(p.numpair, pair);

           }

return input;

       }

};

int main() {

Pairs p;

cout<< "Enter the value of pair object: ";

cin>> p;

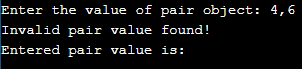
cout<< "Entered pair value is: " << p <<endl;

return 0;

}

**Output:-**

****

****

**Program No. 7 (i)**

**AIM: -** WAP to create a class that will maintain the records of person with details Name and Age) and find the eldest among them. The program must use this pointer to return the result by overloading> operator among two objects.

**Program Code:-**

#include<iostream>

using namespace std;

/\*Write a C++ program to maintain the records of person with details (name and

age) and find the eldest among them. The program must use this pointer to return

the result \*/

// by Aniruddha

class Records

{

int age;

string name;

public:

Records() {};

Records(string n,int a): name(n),age(a) {}

void show()

{

cout<<name<<" : "<<age<<endl;

}

Records eldest(Records o)

{

return (o.age>age)? o: \*this;

}

};

int main()

{

Records ob[3]={Records("Ani",21),Records("Arka",50),Records("Ram",30)};

Records res;

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

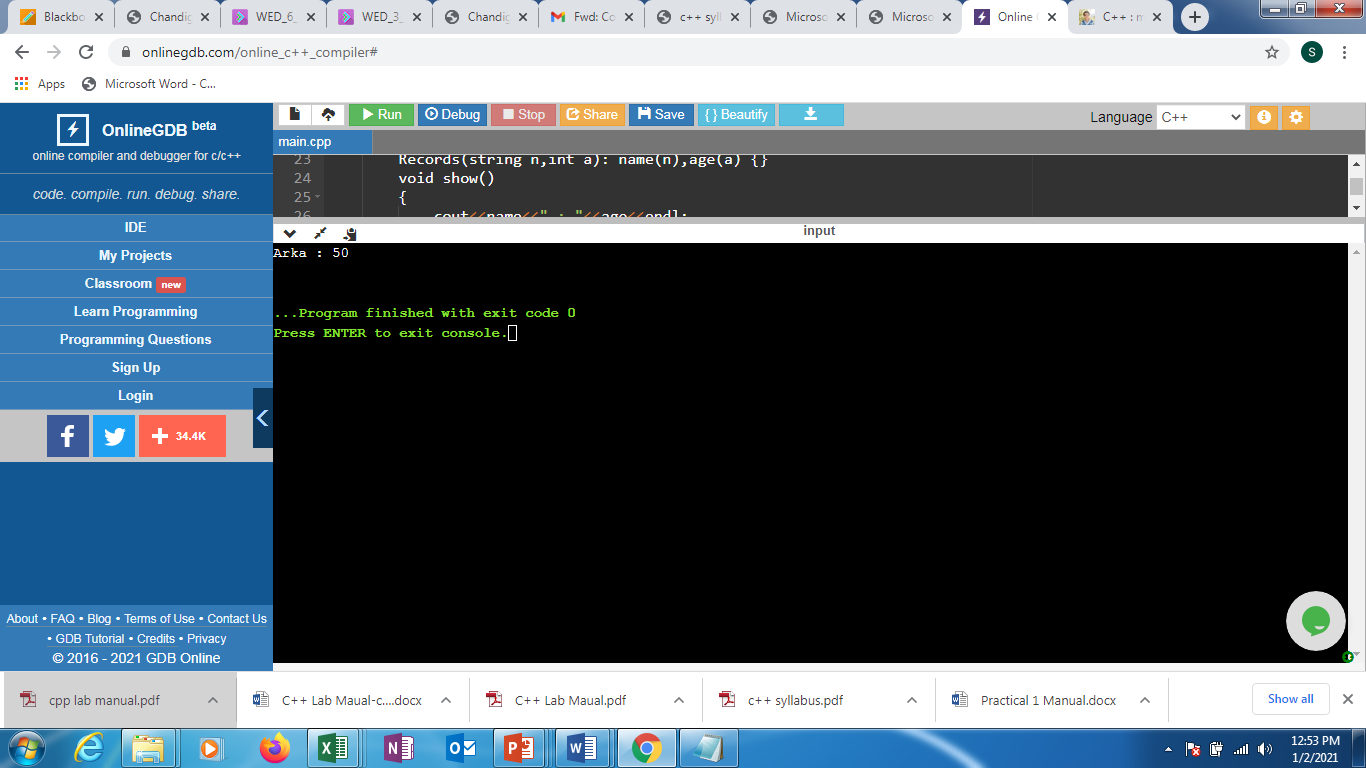
res = ob[i].eldest(ob[i+1]);

res.show();

return 0;

}

**Output:-**

****

**Program No. 7 (ii)**

**AIM: -** WAP to access members using pointer to object members.

**Program Code:-**

#include <iostream>

using namespace std;

class Number

{

              private:

               int num;

              public:

               //constructor

              Number(){ num=0; };

               //member function to get input

                void inputNumber (void)

                  {

               cout<<"Enter an integer number: ";

                cin>>num;

                     }

                 //member function to display number

                  void displayNumber()

                  {

                   cout<<"Num: "<<num<<endl;

                    }

};

//Main function

int main()

{

              //declaring object to the class number

              Number N;

              //input and display number using norn object

              N.inputNumber();

              N.displayNumber();

              //declaring pointer to the object

              Number \*ptrN;

              ptrN = new Number; //creating & assigning memory

               //printing default value

              cout<<"Default value... "<<endl;

              //calling member function with pointer

              ptrN->displayNumber();

              //input values and print

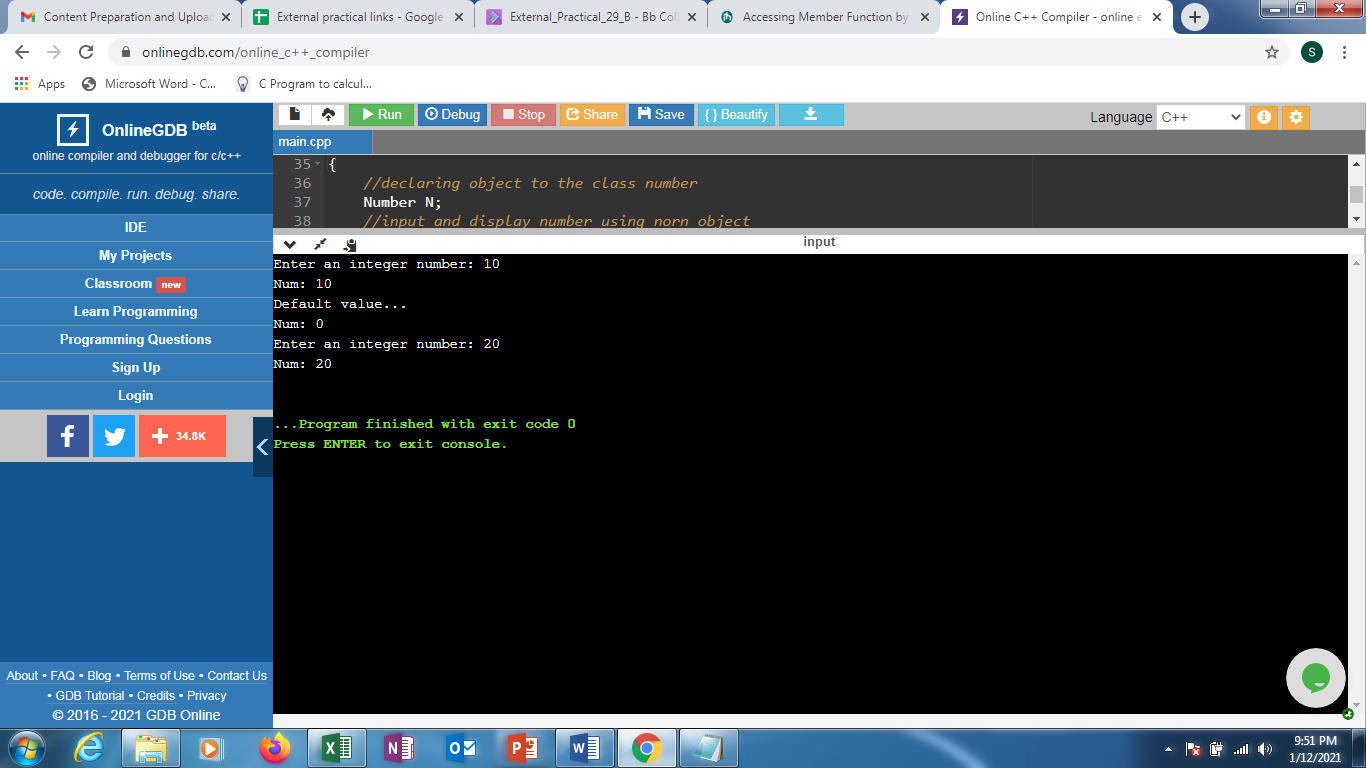
              ptrN->inputNumber();

              ptrN->displayNumber();

              return 0;

}

**Output:-**



**Program No. 7 (iii)**

**Aim: -** WAP to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media).The class should have the functionality for adding new item, issuing, deposit etc. The program should link the objects with concerned function by the concept of runtime polymorphism.

**Program Code:-**

#include<iostream>

#include<string.h>

using namespace std;

class media

{

protected:

char title[50];

float price;

public:

media(char \*s, float a)

{

strcpy(title, s); price = a;

}

virtual void display(){}

};

class book : public media

{

int pages; public:

book(char \*s, float a, int p) : media(s,a)

{

pages = p;

}

void display();

};

class tape : public media

{

float time; public:

tape(char \* s, float a, float t):media(s,a)

{

time =t;

}

void display();

};

void book ::display()

{

cout<<"\n Title:"<<title;

cout<<"\n Pages:"<<pages; cout<<"\n Price:"<<price;

}

void tape ::display ()

{

cout<<"\n Title:"<<title;

cout<<"\n Play Time:"<<time<<"mins"; cout<<"\n Price:"<<price;

}

int main()

{

char \* title = new char[30]; float price, time;

int pages;

cout<<"\n Enter Book Details \n"; cout<<"\n Title:";

cin>>title; cout<<"\n Price:"; cin>>price; cout<<"\n Pages:"; cin>>pages;

book book1(title, price, pages);

cout<<"\n Enter Tape Details";

cout<<"\n Title:";

cin>>title;

cout<<"\n Price:";

cin>>price;

cout<<"\n Play Times(mins):";

cin>>time;

tape tape1(title, price, time);

media\* list[2];

list[0] = &book1;

list[1] = &tape1; cout<<"\n Media Details";

cout<<"\n..............Book. ";

list[0]->display ();

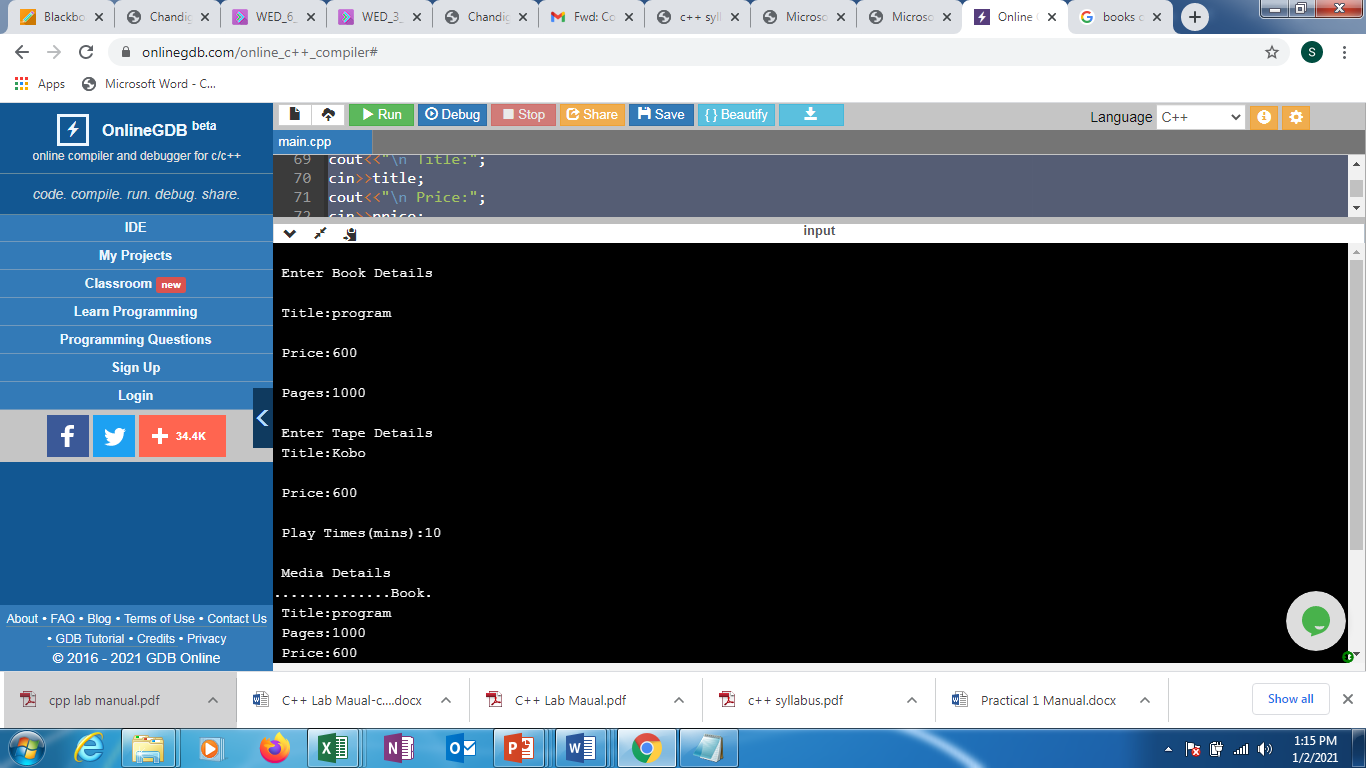
cout<<"\n..............Tape. ";

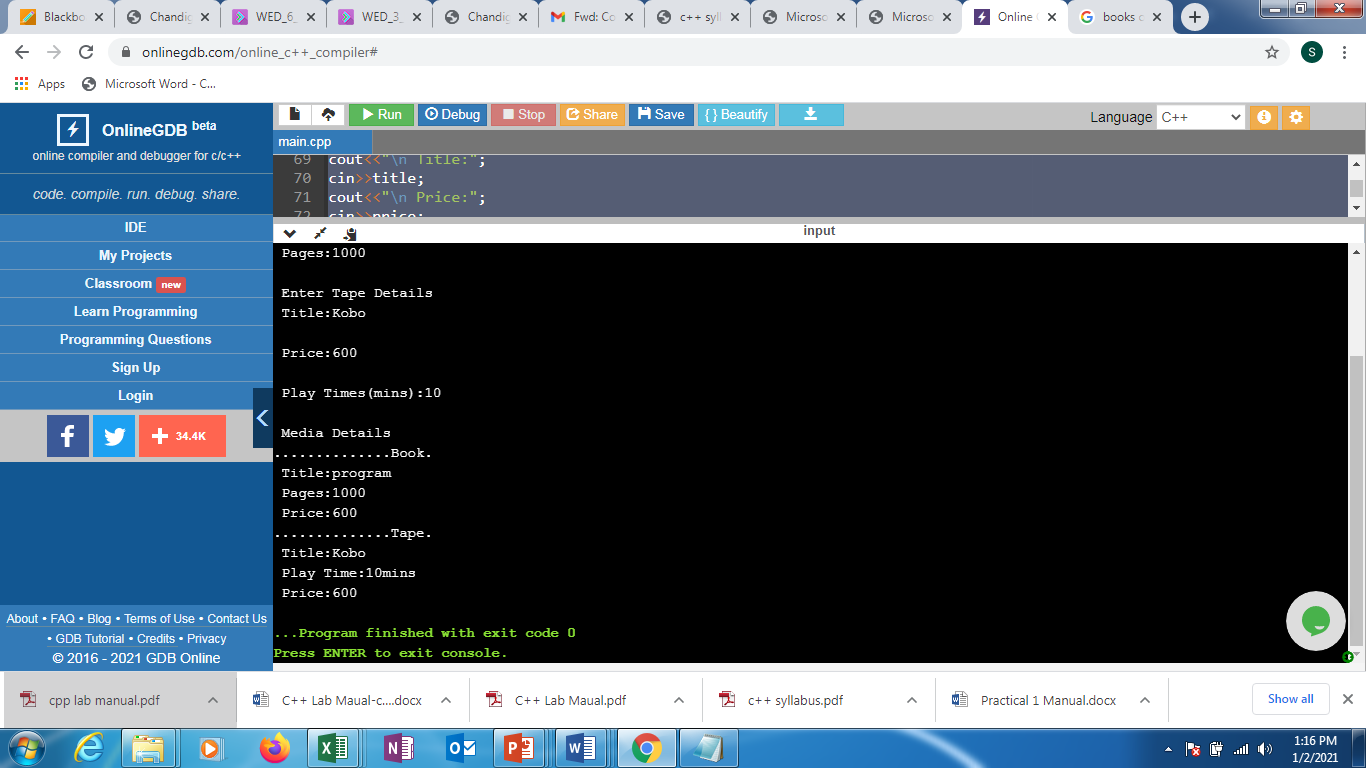
list[1]->display ();

return 0;

}

**Output:-**





**Program No. 8 (i)**

**Aim: -** WAP to perform exception handling for Divide by zero Exception.

**Program Code:-**

|  |
| --- |
| #include <iostream>  using namespace std;  // Defining function Division  float Division(float num, float den)  {      // return the result of division      return (num / den);  } // end Division  int main()  {      // storing 12.5 in numerator      // and 0 in denominator      float numerator = 12.5;      float denominator = 0;      float result;        // calls Division function      result = Division(numerator, denominator);        // display the value stored in result      cout << "The quotient of 12.5/0 is "           << result << endl;    } // end main |

**Output:**

The quotient of 12.5/0 is inf

**Program No. 8 (ii)**

**Aim:-**WAP to implement the exception handling with the functionality of testing the throw restrictions.

**Program Code:-**

#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.

**Program No. 9 (i)**

**Aim: -** WAP to calculate sum of marks of n students of a class inputted via dynamic memory allocation.

**Program Code:**-

#include <iostream>

using namespace std;

int main()

{

int length, sum = 0;

cout << "Enter the number of students in the group" << endl;

cin >> length;

int \*marks = new int[length];

cout << "Enter the marks of the students" << endl;

for( int i = 0; i < length; i++ ) // entering marks of students

{

cin >> \*(marks+i);

}

for( int i = 0; i < length; i++ ) // calculating sum

{

sum += \*(marks+i);

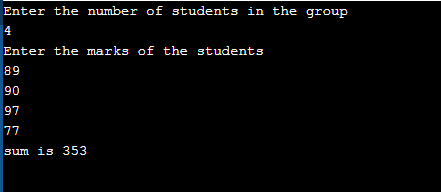
}

cout << "sum is " << sum << endl;

return 0;

}

**Output:**



**Program No. 9 (ii)**

**Aim: -** WAP to allocate memory dynamically for an object of a given class using class’s constructor.

**Program Code:**-

#include <iostream>

using namespace std;

class A

{

public:

A() {

cout << "Constructor" << endl;

}

~A() {

cout << "Destructor" << endl;

}

};

int main()

{

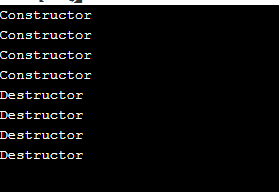
A\* a = new A[4];

delete [] a; // Delete array

return 0;

}

**Output:**



**Program No. 10 (i)**

**Aim: -** WAP to copy the contents of one file to another and display it on output screen.

**Program Code:-**

#include <iostream>

using namespace std;

int main()

{

FILE \*fptr1, \*fptr2;

char filename[100], c;

cout<<"Enter the filename to open for reading \n";

cin>>filename;

// Open one file for reading

fptr1 = fopen(filename, "r");

if (fptr1 == NULL)

{

cout<<"Cannot open file"<< filename;

exit(0);

}

cout<<"Enter the filename to open for writing \n";

cin>> filename;

// Open another file for writing

fptr2 = fopen(filename, "w");

if (fptr2 == NULL)

{

cout<<"Cannot open file "<<filename;

exit(0);

}

// Read contents from file

c = fgetc(fptr1);

while (c != EOF)

{

fputc(c, fptr2);

c = fgetc(fptr1);

}

cout<<"Contents copied to"<<filename;

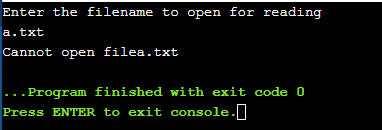
fclose(fptr1);

fclose(fptr2);

return 0;

}

**Output:**



**Program No. 10 (ii)**

**Aim: -** WAP to read the class object of student info such as name, age and rollno from the keyboard and to store them on a specified file using read() and write() functions. Again the same file is opened for reading and displaying the contents of the file on the screen.

**Program Code:**

//C++ program to write and read object using read and write function.

#include <iostream>

#include <fstream>

using namespace std;

//class student to read and write student details

class student

{

private:

char name[30];

int age;

char roll[20];

public:

void getData(void)

{ cout<<"Enter name:"; cin.getline(name,30);

cout<<"Enter age:"; cin>>age;

cout<<"Enter roll number:"; cin.getline(name,20);

}

void showData(void)

{

cout<<"Name:"<<name<<",Age:"<<age<<"Roll:"<<roll<<endl;

}

};

int main()

{

student s;

ofstream file;

//open file in write mode

file.open("aaa.txt",ios::out);

if(!file)

{

cout<<"Error in creating file.."<<endl;

return 0;

}

cout<<"\nFile created successfully."<<endl;

//write into file

s.getData(); //read from user

file.write((char\*)&s,sizeof(s)); //write into file

file.close(); //close the file

cout<<"\nFile saved and closed succesfully."<<endl;

//re open file in input mode and read data

//open file1

ifstream file1;

//again open file in read mode

file1.open("aaa.txt",ios::in);

if(!file1){

cout<<"Error in opening file..";

return 0;

}

//read data from file

file1.read((char\*)&s,sizeof(s));

//display data on monitor

s.showData();

//close the file

file1.close();

return 0;

}

**Output:**

