**Q1. Write program to generate following pattern**

**a)**

**1**

**1 2**

**12 3**

**1 2 3 4**

**PROGRAM :**

**//Program to genarate Given pattern**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{ int total\_rows = 4;

for(int row = 1; row<=total\_rows; row++) *//Dynamic initilization of int row*

{

for(int col=1; col<=row; col++)

{

cout<<col;

}

cout<<endl;

}

getch();

return 0;

}

**OUTPUT:**

****

**b)**

\*

\* \*

\* \* \*

\* \* \* \*

**PROGRAM :**

**//Program to genarate pattern**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{ int total\_rows = 4;

for(int row = 0; row<=total\_rows; row++) *//Dynamic initilization of int row*

{

for(int col=0; col<=(total\_rows \* 2)-1; col++)

{

if(col>total\_rows-row && col < total\_rows+row)

{

if((row+col)%2!=0)

cout<<"\*";

else

cout<<" ";

}

else

cout<<" ";

}

cout<<endl;

}

getch();

return 0;

}

**OUTPUT :**



**Q2. WAP in C++ which uses functions to swap two integer & two float numbers by using reference variable.**

**PROGRAM:**

**//Program to swap integer and float value using reference variable**

#include<iostream>

#include<conio.h>

using namespace std;

void swap\_int(int &,int &);

void swap\_float(float &,float &);

*//function declarations*

void swap\_int(int & x1,int & x2) *//function definition*

{

int temp = x1;

x1 = x2;

x2 = temp;

}

void swap\_float(float & x1,float & x2) *// x1 and x2 are reference variable*

{

float temp = x1;

x1 = x2;

x2 = temp;

}

int main()

{

int a1 = 34;

int a2 = 20;

float f1 = 30.01;

float f2 = 24.40;

cout<<"Enter two integers : ";

cin>>a1>>a2;

cout<<"Before swapping a1 = "<<a1<<" and a2 = "<<a2<<endl;

swap\_int(a1,a2); *//function calling*

cout<<"After swapping a1 = "<<a1<<" and a2 = "<<a2<<endl;

cout<<endl<<endl;

cout<<"Enter two float values : ";

cin>>f1>>f2;

cout<<"Before swapping f1 = "<<f1<<" and f2 = "<<f2<<endl;

swap\_float(f1,f2);  *//function calling*

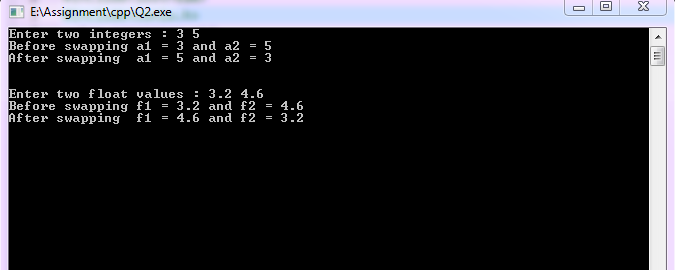
cout<<"After swapping f1 = "<<f1<<" and f2 = "<<f2<<endl;

getch();

return 0;

}

**OUTPUT:**

****

**Q3. Create a single program to perform following tasks without using library functions:**

**To reverse the string accepted as an argument.**

**To count the number of characters in string passed as argument in form of character array.**

**PROGRAM :**

**//Program to calculate length of string and reverse the string**

#include<iostream>

#include<conio.h>

using namespace std;

int string\_length(char);

void reverse\_string(char);

*//Function declearation*

void reverse\_string(char rev[20],int length) *//function definition*

{

int i;

cout<<"Reverse of the string is : ";

for(i =length-1; i>=0; --i)

{

cout<<rev[i];

}

cout<<endl;

}

int string\_length(char p[20])

{

int count;

for(count = 0; p[count]!= '\0'; ++count);

*//loop till getting null character*

return count;

}

int main()

{

char str[20];

cout<<"Enter a string : ";

cin>>str;

cout<<endl;

int len = string\_length(str);

*//storing length of string*

cout<<"Length of string is : "<<len<<endl;

reverse\_string(str,len);

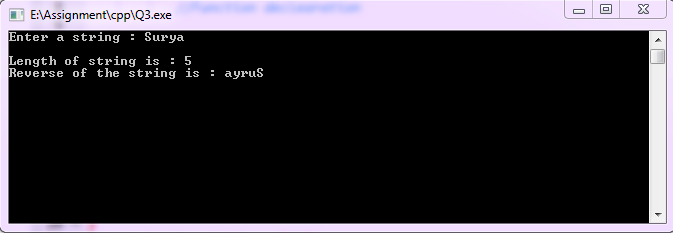
*//passing length and string to the function*

getch();

return 0;

}

**OUTPUT:**

****

**Q4. WAP in C++ to create a structure named complex having data member real and imag. Create member function add\_complex which takes structure as an argument and return structure. Using function add two complex numbers.**

**PROGRAM :**

**//Program to add to structures with trurn type structure(name complex)**

#include<iostream>

#include<conio.h>

using namespace std;

struct complex *//Structure defined*

{

float real;

float imag;  *//data members*

void get\_data()  *//member function definition*

{

cout<<"Enter the real value : ";

cin>>real;

cout<<"Enter the imag value : ";

cin>>imag;

}

void display() *//member function definition*

{

cout<<real<<"+"<<imag<<"i";

}

};

complex add(complex,complex); //function having comlpex type arguments

complex add(complex c1,complex c2)

{

complex c;

c.real = c1.real + c2.real; //adding real parts and storing in c.real

c.imag = c1.imag + c2.imag; //adding imag parts and storing in c.imag

return c; //returning a value of type complex

}

int main()

{

complex comp1,comp2,add\_comp;

comp1.get\_data(); //data members initilization

comp2.get\_data();

add\_comp = add(comp1,comp2);

//calling the function add and passing comp1 and comp2

cout<<endl<<"Addition of : ";

comp1.display();

cout<<" and ";

comp2.display();

cout<<" is : ";

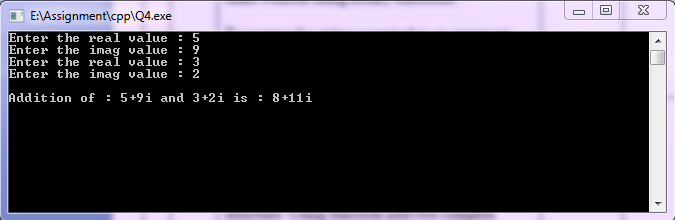
add\_comp.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q.5 Write a program to perform arithmetic operations using inline function.**

**PROGRAM :**

**//Program to perform arithmetic operations using inline function**

#include<iostream>

#include<conio.h>

using namespace std;

inline void add(float a,float b) *//inline function definition*

{

float c = a + b;

cout<<a<<" + "<<b<<" = "<<c<<endl;

}

inline void subtract(float a,float b) *//inline function definition*

{

float c = a - b;

cout<<a<<" - "<<b<<" = "<<c<<endl;

}

inline void multi(float a,float b)  *//inline function definition*

{

float c = a \* b;

cout<<a<<" \* "<<b<<" = "<<c<<endl;

}

inline void divide(float a,float b) *//inline function definition*

{

float c = a / b;

cout<<a<<" / "<<b<<" = "<<c<<endl;

}

int main()

{

float n1,n2;

cout<<"Enter two values to perform arithmetic operations : ";

cin>>n1>>n2;

cout<<endl;

add(n1,n2);

subtract(n1,n2);

multi(n1,n2);

divide(n1,n2);

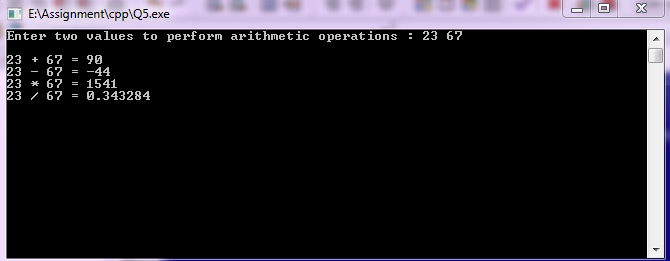
*//inline function calling*

getch();

return 0;

}

**OUTPUT :**

****

**Q6.** **WAP in C++ to calclulate the area of circle, reractangle,square and triangle using inline function.**

**PROGRAM :**

**//Program to calculate the area of circle ,rectangle ,square and tringle using inline function**

#include<iostream>

#include<conio.h>

using namespace std;

inline void area\_of\_circle()  *//inline function definition*

{

float radius;

cout<<"Enter radius of circle : ";

cin>>radius;

cout<<"Area of circle is : "<<22/7.0 \* radius \* radius<<endl; *//area of circle*

}

inline void area\_of\_rectangle() *//inline function definition*

{

float a,b;

cout<<"Enter sides of rectangle : ";

cin>>a>>b;

cout<<"Area of rectangle is : "<<a\*b<<endl;  *//area of rectangle*

}

inline void area\_of\_square() *//inline function definition*

{

float a;

cout<<"Enter side of square : ";

cin>>a;

cout<<"Area of square is : "<<a\*a<<endl; *//area of square*

}

inline void area\_of\_tringle() *//inline function definition*

{

float height,length;

cout<<"Enter height and length : ";

cin>>height>>length;

cout<<"Area of tringle is : "<< 0.5 \* height \* length<<endl;  *//area of tringle*

}

int main()

{

area\_of\_circle(); *//calculate the area of circle*

cout<<endl;

area\_of\_rectangle(); *//calculate the area of rectangle*

cout<<endl;

area\_of\_square(); *//calculate the area of square*

cout<<endl;

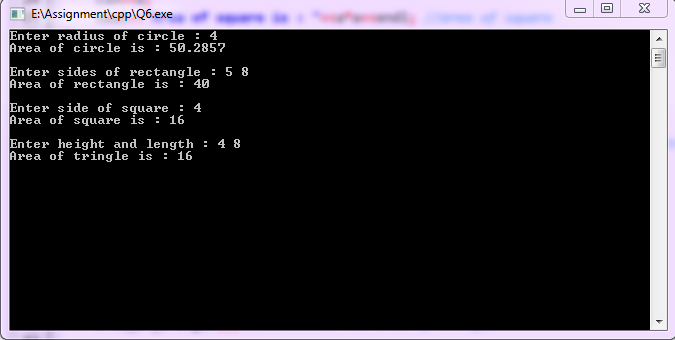
area\_of\_tringle(); *//calculate the area of tringle*

getch();

return 0;

}

**OUTPUT :**

****

**Q7. WAP in c++ To count no. of vowels, consonants in each word of a sentence passed as argument in form of character array.**

**PROGRAM :**

**//Program to count no. of vowels and consonants in giver string**

#include<iostream>

#include<conio.h>

#include<string.h>

using namespace std;

void count\_vowel\_conso(char); *//fumction declaration*

void count\_vowel\_conso(char s[20]) *//function definition*

{

int i;

bool lowercase,upercase; *//bool type variable*

bool conso\_check;

int count\_vowel=0;

int count\_conso=0;

for(i=0; i<strlen(s); i++)

{

conso\_check = ( ( ( s[i]>=97 ) && ( s[i]<=132 ) ) || ( ( s[i]>=65 ) && ( s[i]<=90 ) ) );

lowercase = ( ( s[i] == 'a') ||( s[i] == 'e')|| (s[i] == 'i' )|| (s[i] == 'o' ) || (s[i] == 'u' ) );

upercase = ( ( s[i] == 'A') ||( s[i] == 'E')|| (s[i] == 'I' )|| (s[i] == 'O' ) || (s[i] == 'U' ) );

if(lowercase || upercase )

count\_vowel++;

else if(conso\_check)

count\_conso++;

}

cout<<" Total no. of Vowels = "<<count\_vowel<<endl;

cout<<"Total no. of Consonants = "<<count\_conso<<endl;

}

int main()

{

char str[20];

cout<<"Enter a string : ";

cin>>str;

count\_vowel\_conso(str);

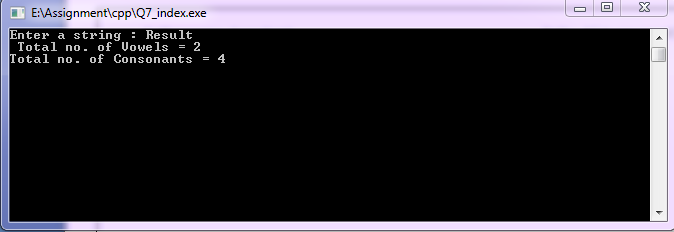
*//function calling and passing the string*

getch();

return 0;

}

**OUTPUT :**

****

**Q8.** **Write program in C++ to calculate simple interest and compound interest using default argument.**

**PROGRAM :**

**//Program to calculate simple interest and compound interest using default argument**

#include<iostream>

#include<conio.h>

using namespace std;

float simple\_interest(float,float,float);

void compound\_interest(float,float);

*//function declaration*

float simple\_interest(float p,float t, float r = 3) *//Default argument r (rate) is 3%*

{

float Si = (p \* t \* r ) / 100;

cout<<"Simple Interest of principle amount ("<<p<<"), time ("<<t<<") and rate("<<r<<") is : "<<Si<<endl<<endl;

return Si;

}

void compound\_interest(float p,float si)

{

float ci = p + si;

cout<<"Compound Interest is : "<<ci<<endl;

}

int main()

{

float princ,time,rate;

cout<<"Enter principle amount : ";

cin>>princ;

cout<<"Enter Time (in months) : ";

cin>>time;

cout<<"Enter Rate (im %) : ";

cin>>rate;

float simp\_int = simple\_interest(princ,time,rate);

*//no. of actual and formal argument are same*

*//Not a condition for default argument*

compound\_interest(princ,simp\_int); *//compount interest*

cout<<endl<<endl;

cout<<"With default rate ..."<<endl;

cout<<"Enter principle amount : ";

cin>>princ;

cout<<"Enter Time (in months) : ";

cin>>time;

simp\_int = simple\_interest(princ,time);

*// no. of actual argument is less than*

*//formal argument( so default argument will be passed)*

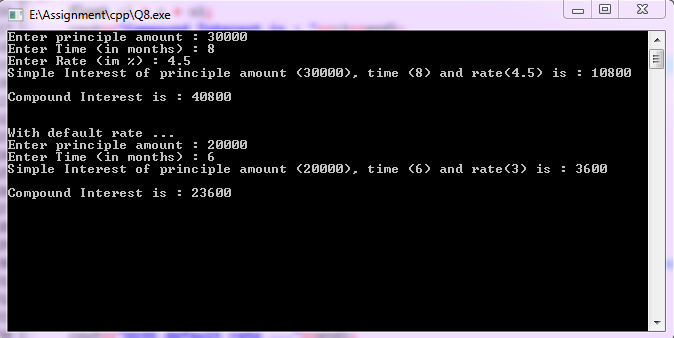
compound\_interest(princ,simp\_int);

getch();

return 0;

}

**OUTPUT :**

****

**Q9.** **Create a class named calculate that uses overloaded function calclulate\_area of circle, reractangle,square and triangle.**

**PROGRAM :**

**//Program to calculate area of circle,square,rectangle and tringle using function overloading**

#include<iostream>

#include<conio.h>

using namespace std;

class calculate

{

float area;

public:

void calculate\_area(float); *// function overloading declaration*

void calculate\_area(float,float); *// function overloading declaration*

void calculate\_area(float,float,float); *// function overloading declaration*

};

void calculate :: calculate\_area(float radius) *//function defining outside of the class*

{

area= 22/7.0 \* radius \* radius;

cout<<"Area Of Circle with radius "<<radius<<" is : "<<area<<endl;

}

void calculate :: calculate\_area(float length,float width)

{

area = length \* width;

if(length == width)

cout<<"Area Of Square with side "<<length<<" is : "<<area<<endl;

else

cout<<"Area Of Rectangle with sides "<<length<<" and "<<width<<" is : "<<area<<endl;

}

void calculate :: calculate\_area(float half,float height,float length)

{

area = half \* height \* length;

cout<<"Area Of Tringle with height "<<height<<" and length "<<length<<" is : "<<area<<endl;

}

int main()

{

calculate obj;

cout<<endl;

obj.calculate\_area(2); *//calculate the area of circle*

cout<<endl;

obj.calculate\_area(4,6); *//calculate the area of rectangle*

cout<<endl;

obj.calculate\_area(5,5); *//calculate the area of square*

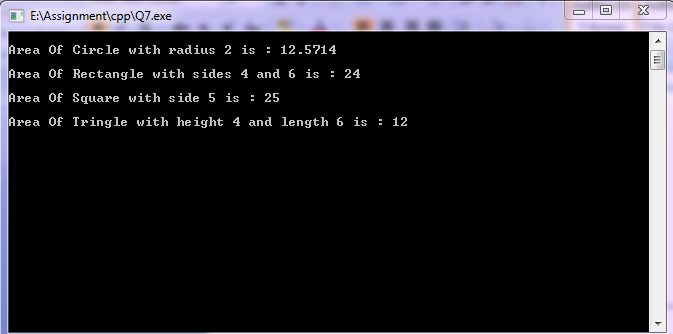
cout<<endl;

obj.calculate\_area(0.5,4,6); *//calculate the area of tringle*

getch();

return 0; }

**OUTPUT :**

****

**Q10.** **Create a class Student having data members to store roll number, name of student, name of three subjects, max marks, min marks, obtained marks. Declared an object of class student. Provide facilities to input data in data members and display result of student?**

**PROGRAM :**

**//Program to store data of class student and display result of student**

#include<iostream>

#include<conio.h>

using namespace std;

class student

{ int rollno;

char stu\_name[20];

char sub\_name[3][20];

float max\_mark[3],min\_mark[3],obt\_mark[3];

*//data members*

public:

void input\_data();

void result();

*//member function declaration*

};

void student::input\_data() *//member function definition outside of class*

{

cout<<"Enter details of the student : "<<endl;

cout<<"Roll no. : ";

cin>>rollno;

cout<<"Name : ";

cin>>stu\_name;

cout<<"Enter Subjects Details "<<endl<<endl;

int i;

for(i=0; i<3; i++)

{

cout<<"Subject no. "<<i+1<<endl<<endl;

cout<<"Name of subject : ";

cin>>sub\_name[i];

cout<<"Maximum marks : ";

cin>>max\_mark[i];

cout<<"Minium marks : ";

cin>>min\_mark[i];

cout<<"Obtained marks : ";

cin>>obt\_mark[i];

}

}

void student::result()

{

cout<<"Result of student "<<stu\_name<<" is : "<<endl<<endl;

float total\_max,total\_obt;

float per;

int i;

for(i=0; i<3; i++)

{

total\_max = total\_max + max\_mark[i];

total\_obt = total\_obt + obt\_mark[i];

}

per = total\_obt \* 100 / total\_max;

cout<<"Total obtained marks out of "<<total\_max<<" = "<<total\_obt<<endl;

cout<<"Total percentage = "<<per<<"%"<<endl;

if(per >= 70)

cout<<"First Division..."<<endl;

else if(per < 70 && per >= 50)

cout<<"Second Division..."<<endl;

else if(per > 33 && 50 > per)

cout<<"Third Division.."<<endl;

else

cout<<"Fail"<<endl;

}

int main()

{

student stu1; *//Object created of type student*

stu1.input\_data();

stu1.result();

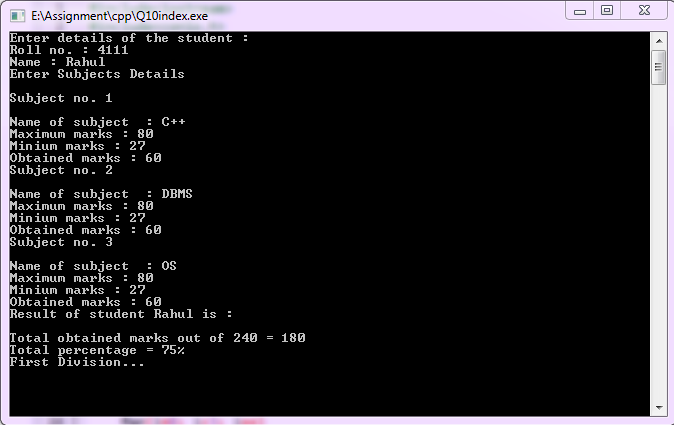
*//calling member function through object*

getch();

return 0;

}

**OUTPUT :**

****

**Q11. Create a class student having data members to store rollno.,name of student, name of 3 subjects , max marks,min marks,obtain marks .use nesting of member function Declare an array of object to input data of 3 students. Provide facilities to display result of all students and to display result of specific student whose roll number is given ?**

**PROGRAM:**

**//Programn to store data of 3 students and display result of all and also display result of specific student**

#include<iostream>

#include<conio.h>

using namespace std;

class student

{ int rollno;

char stu\_name[20];

char sub\_name[3][20];

float max\_mark[3],min\_mark[3],obt\_mark[3];

*//data members*

public:

void input\_data();

void disp\_spec(int);

void disp\_result();

*//member function declaration*

};

void student::input\_data() *//member function definition*

{ cout<<"Roll no. : ";

cin>>rollno;

cout<<"Name : ";

cin>>stu\_name;

cout<<"Enter Subjects Details "<<endl<<endl;

int i;

for(i=0; i<3; i++)

{ cout<<"Subject no. "<<i+1<<endl<<endl;

cout<<"Name of subject : ";

cin>>sub\_name[i];

cout<<"Maximum marks : ";

cin>>max\_mark[i];

cout<<"Minium marks : ";

cin>>min\_mark[i];

cout<<"Obtained marks : ";

cin>>obt\_mark[i];

}

}

void student::disp\_spec(int roll) *//member function definition*

{ if(roll==rollno)

disp\_result()*;* } *//nesting of member function*

void student::disp\_result() *//member function definition*

{ cout<<"Result of student "<<stu\_name<<"........"<<endl;

float total\_max=0,total\_obt=0;

int i;

for(i=0; i<3; i++)

{ total\_max = total\_max + max\_mark[i];

total\_obt = total\_obt + obt\_mark[i];

}

float per;

per = total\_obt \* 100 / total\_max;

cout<<"Total obtained marks in "<<total\_max<<" is = "<<total\_obt<<endl;

cout<<"Percentage is = "<<per<<"%"<<endl;

if(per>=70)

cout<<"First Division.."<<endl;

else if(per<70 && per>=50)

cout<<"Second division.."<<endl;

else if(per>=33 && per<50)

cout<<"Third division..."<<endl;

else

cout<<"Fail.."<<endl;

cout<<endl; }

int main()

{ student stu[3]; */ /array of object created*

int i;

for(i=0; i<3; i++)

{

cout<<"Enter details of the student "<<i+1<<" : "<<endl;

stu[i].input\_data(); *//member function calling*

}

cout<<endl;

for(i=0; i<3; i++)

stu[i].disp\_result(); *//member function calling*

cout<<endl;

int roll;

cout<<"Enter Roll no. of student : ";

cin>>roll;

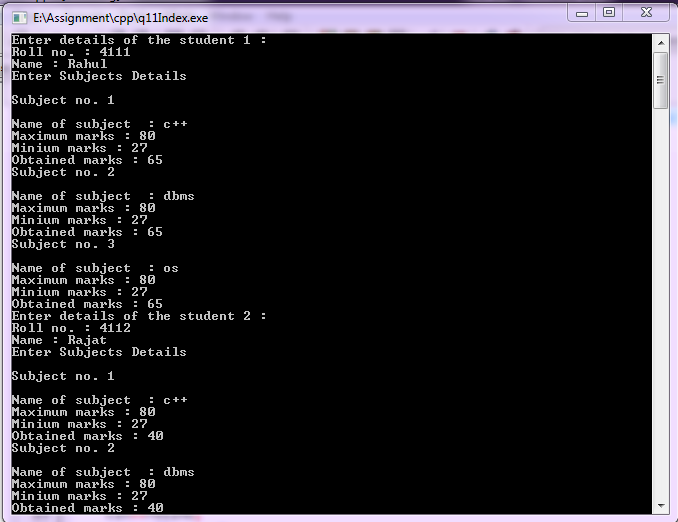
for(i=0; i<3; i++)

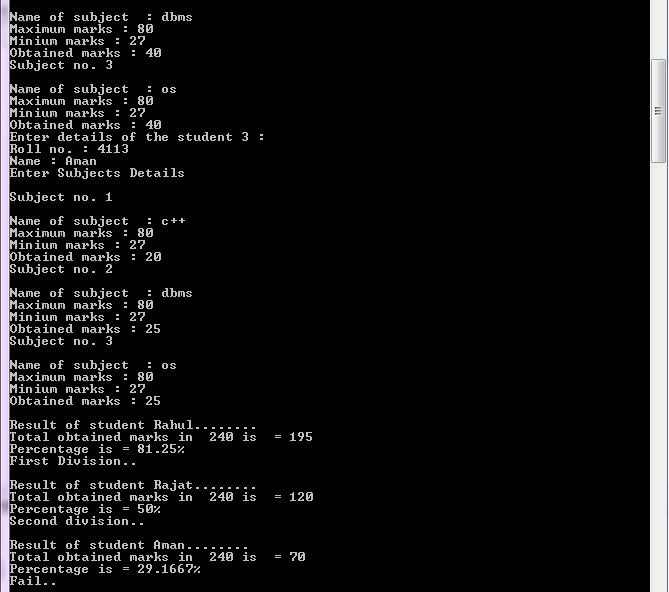
stu[i].disp\_spec(roll); *//member function calling*

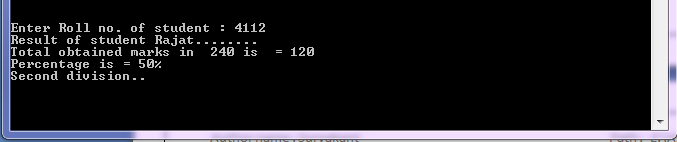
getch();

return 0; }

**OUTPUT:**

****

****

****

**Q12. Create a class named ‘array’ having an array of integers having 5 elements as data member provide following facilities :**

**Constructor to get number in array elements.**

**Sort the elements.**

**PROGRAM :**

**//Program to sort integers**

#include<iostream>

#include<conio.h>

using namespace std;

class array

{ int a[5]; *//data member*

public:

array(int b[5]) *//parameterized constructor*

{ int i;

for(i=0; i<5; i++)

a[i] = b[i]; }

void sort();

void swap(int &,int &); *//member function declaration*

};

void array::sort() *//member functiion definition*

{ int i,j;

for(i=0; i<5-1; i++)

{ for(j=0; j<5-1; j++)

{ if(a[j] > a[j+1])

swap(a[j],a[j+1]); }  *//swapping the values*

}

cout<<"Array in Ascanding oerder : "<<endl;

for(i=0; i<5; i++)

cout<<a[i]<<endl; }

void array::swap(int &a,int &b) *//member functiion definition*

{ int temp=a;

a=b;

b=temp; }

int main()

{ int ary[5],i;

cout<<"Enter 5 intgers : ";

for(i=0; i<5; i++)

cin>>ary[i];

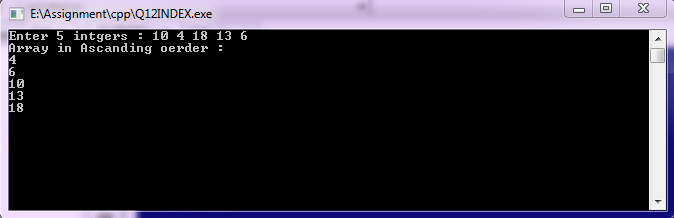
array ary1(ary);  *//creating an object also ary's values passing*

ary1.sort(); *//member function calling*

getch();

return 0; }

**OUTPUT :**

****

**Q13.** **Create a class Static\_demo with static member functions for following tasks:-**

**1. To find factorial by recursive member function**

**2. To check whether a no. is prime or not.**

**PROGRAM :**

**//Program to find factorial and check prime number with static member function**

#include<iostream>

#include<conio.h>

using namespace std;

class Static\_demo

{

public:

static double find\_facto(double); *//static member function declaration*

static void check\_prime(int); *//static member function declaration*

};

double Static\_demo::find\_facto(double n) *//static member function definition*

{

if( n < 1)

return 1;

else

return n\*find\_facto(n-1); *//recursion*

}

void Static\_demo::check\_prime(int a) *//static member function definition*

{

int i,j;

int c=0;

for(i=1; i<=a; i++)

{

if(a%i == 0)

c++;

}

if(c<=2)

cout<<a<<" is a prime number"<<endl;

else

cout<<a<<" is not a prime number"<<endl;

}

int main()

{

double num;

cout<<"Enter a number which factorial you want : ";

cin>>num;

double fact = Static\_demo::find\_facto(num);

*//calling static member function of static\_demo class*

cout<<"Factorial is : "<<fact<<endl;

int n;

cout<<"Enter a numbet to check whether it is prime or not : ";

cin>>n;

Static\_demo::check\_prime(n);

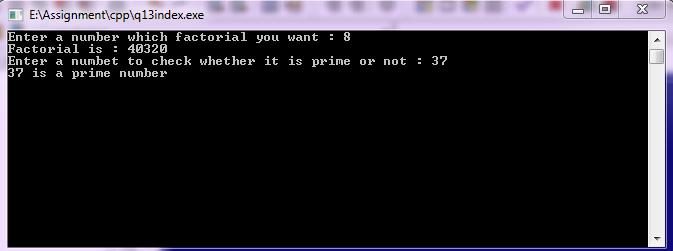
*//calling static member function*

getch();

return 0;

}

**OUTPUT :**

****

**Q14.** **Write a class complex having data members to store real and imaginary part provide following**

**Add two complex no using object as an argument.**

**subtract two complex no using object as an argument.**

**PROGRAM :**

**//Program to add and subtract two complex number**

#include<iostream>

#include<conio.h>

using namespace std;

class complex

{ public:

float real, imag; *//Data members in public mode*

void get\_data();  *//Member function declaration*

};

void complex::get\_data() *//member function definition*

{ cout<<"Enter complex : "<<endl;

cout<<"Real = ";

cin>>real;

cout<<"Imaginary = ";

cin>>imag; }

complex add\_complex(complex,complex); *//function prototyping*

complex sub\_complex(complex,complex); *//function prototyping*

complex add\_complex(complex c1,complex c2)  *//function definition*

{ complex temp;

temp.real = c1.real + c2.real;

temp.imag = c1.imag + c2.imag;

return temp; }

complex sub\_complex(complex c1,complex c2) *//function definition*

{ complex temp;

temp.real = c1.real - c2.real;

temp.imag = c1.imag - c2.imag;

return temp; }

int main()

{ complex comp1,comp2,add,sub;

comp1.get\_data();

comp2.get\_data();

add = add\_complex(comp1,comp2); *//function calling with passing objects as argument*

cout<<"Addition of complext no is : "<<add.real<<" + "<<add.imag<<"i"<<endl;

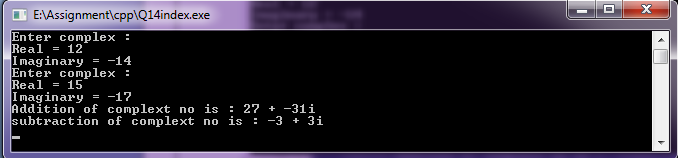
sub = sub\_complex(comp1,comp2); *//function calling with passing objects as argument*

cout<<"subtraction of complext no is : "<<sub.real<<" + "<<sub.imag<<"i"<<endl;

getch();

return 0; }

**OUTPUT :**

****

**Q15. Write swapping program to demonstrate call by value , call by address and call by reference in a single program ?**

**PROGRAM :**

**//Swapping program to demonstrate call by value, call by reference and call by address**

#include<iostream>

#include<conio.h>

using namespace std;

void swap\_by\_call(int,int);

void swap\_by\_ref(int &,int &);

void swap\_by\_add(int \*,int \*);

*// function prototyping*

void swap\_by\_call(int a,int b) *//function declaration*

{

int temp = a;

a = b;

b = temp;

cout<<"After swapping (call by value) In swap\_by\_call function : "<<endl;

cout<<"num1 = "<<a<<" and num2 = "<<b<<endl<<endl;

}

void swap\_by\_ref(int &a,int &b) *//function declaration*

{

int temp = a;

a = b;

b = temp;

cout<<"After swapping (call by reference) in swap\_by\_ref function : "<<endl;

cout<<"num1 = "<<a<<" and num2 = "<<b<<endl<<endl;

}

void swap\_by\_add(int \*a,int \*b) *//function declaration*

{

int temp = \*a;

\*a = \*b;

\*b = temp;

cout<<"After swapping (call by address) in swap\_by\_add function : "<<endl;

cout<<"num1 = "<<\*a<<" and num2 = "<<\*b<<endl<<endl;

}

int main()

{

int x,y;

cout<<"Enter two integer to swap : ";

cin>>x>>y;

cout<<endl;

cout<<"Before swapping : "<<endl;

cout<<"num1 = "<<x<<"\t num2 = "<<y<<endl<<endl;

swap\_by\_call(x,y); *//function calling*

cout<<"After swapping (call by value) in main() function : "<<endl;

cout<<"num1 = "<<x<<" and num2 = "<<y<<endl<<endl;

swap\_by\_ref(x,y); *//function calling*

cout<<"After swapping (call by reference) in main() function : "<<endl;

cout<<"num1 = "<<x<<" and num2 = "<<y<<endl<<endl;

swap\_by\_add(&x,&y);  *//function calling*

cout<<"After swapping (call by address) in main() function : "<<endl;

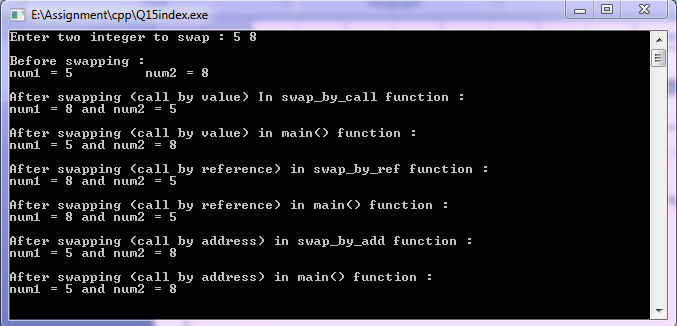
cout<<"num1 = "<<x<<" and num2 = "<<y<<endl<<endl;

getch();

return 0;

}

**OUTPUT :**

****

**Q16.** **Write a program for to create class polar data member radius and angle define constructor of all three types and create destructor and test function in main.**

**PROGRAM :**

**//Program to demonstrate constructors and destrutor**

#include<iostream>

#include<conio.h>

using namespace std;

class polar *//class definition*

{

float radius;

float angle;

public:

polar() *//default construtor definition*

{ radius = 0;

angle = 0;

cout<<"Default constructor Invoked \n";

cout<<"Radius = "<<radius<<"\t Angle = "<<angle<<endl;

}

polar(float r,float a) *//parameterized constructor definition*

{ radius = r;

angle = a;

cout<<"Parameterized constructor Invoked \n";

cout<<"Radius = "<<radius<<"\t Angle = "<<angle<<endl;

}

polar(polar &p) *//copy consttructor definition*

{ radius = p.radius;

angle = p.angle;

cout<<"Copy constructor Invoked "<<endl;

cout<<"Radius = "<<radius<<"\t Angle = "<<angle<<endl;

}

~polar() *//Destructor definition*

{

cout<<"Destructor Invoked"<<endl;

}

};

int main()

{ cout<<"Block 1 begins \n";

polar p1; *//object created and default constructor called*

{

cout<<" Block 2 begins \n";

polar p2(5,7);

*//object created and parameterized constructor called*

{

cout<<" Block 3 begins \n";

polar p3 = p2; *//object created annd copy constructor called*

cout<<" Block 3 ends \n";

*//destructor called automatically*

}

cout<<" Block 2 ends \n";

*//destructor called automatically*

}

cout<<"Block 1 ends \n";

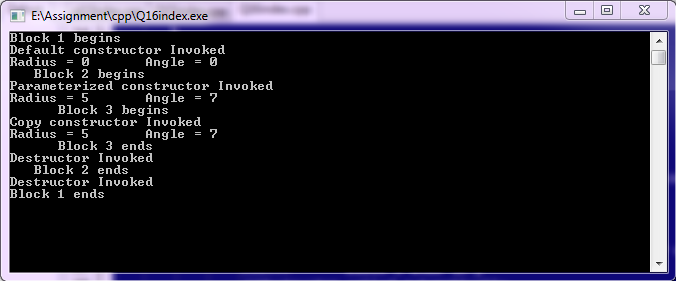
*//destructor called automatically*

getch();

return 0;

}

**OUTPUT :**

****

**Q17.** **WAP to create a class employee having data member employed id,salary.proide member function for data input,output,use pointer to an object information of employee and test the program in function main ?**

**PROGRAM :**

**//program to access the member functions of a class by pointer to an object**

#include<iostream>

#include<conio.h>

using namespace std;

class employee *//class definition*

{

int emp\_id;

float salary;

*//data members*

public: *//public area*

void get\_emp\_data();  *//member function declaration*

void disp\_emp\_data(); *//member function declaration*

};

void employee::get\_emp\_data() *//member function definition*

{

cout<<"Enter employee details : "<<endl;

cout<<"Employee id = ";

cin>>emp\_id;

cout<<"Employee salary = ";

cin>>salary;

}

void employee::disp\_emp\_data() *//member function definition*

{

cout<<"Details of employee is : "<<endl;

cout<<"Employee ID = "<<emp\_id<<" and Salary = "<<salary<<endl;

}

int main()

{

employee emp1;

*//object emp1 created of employee class*

cout<<"Accessing through object : "<<endl;

emp1.get\_emp\_data();

*//data entered with the help of object emp1*

employee \*emp\_ptr;

*//creating pointer emp\_ptr of type employee*

emp\_ptr = &emp1;  *//emp\_ptr holds address of object emp1*

*//emp\_ptr is pointer to object(emp1)*

cout<<"Accessing through pointer to an object : "<<endl;

emp\_ptr->disp\_emp\_data();

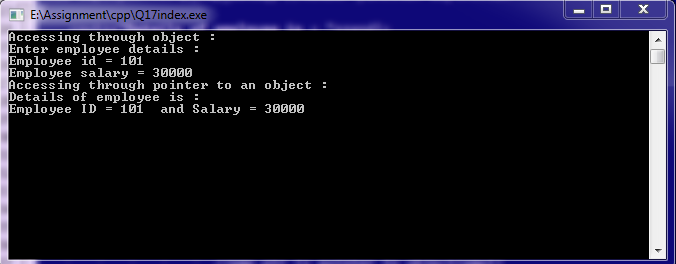
*//Displaying data stored in employee class via emp\_ptr (pointer to an object)*

getch();

return 0;

}

**OUTPUT :**

****

**Q18. Write program-using class and to store data about books(book id,Title,Author,Price,Edition)**

**provide following facilities :**

**Addition of new books.**

**Searching for availability of books if provide author.**

**PROGRAM :**

**//Program to add new book and search the book**

#include<iostream>

#include<conio.h>

#include<string.h>

using namespace std;

class books

{

int book\_id;

char title[20];

char author[20];

float price;

char edition[20]; *//data members*

public:

void add\_book();

int search\_book(char);

void display\_books(); *//member function declaration*

};

void books::add\_book() *//member fun. definition*

{ cout<<"Enter book's details : "<<endl;

cout<<"Book ID : ";

cin>>book\_id;

cout<<"Title : ";

cin>>title;

cout<<"Author : ";

cin>>author;

cout<<"Price : ";

cin>>price;

cout<<"Edition : ";

cin>>edition;

}

int books::search\_book(char tmp\_author[20]) *//member fun. definition*

{

if(strcmp(tmp\_author,author)

{ display\_books();

return 1; }

else

return 0; }

void books::display\_books() *//member fun. definition*

{ cout<<"Books deatails....."<<endl<<endl;

cout<<"Book ID : "<<book\_id;

cout<<"\t Title : "<<title<<endl;

cout<<"Author : "<<author;

cout<<"\t Price : "<<price<<endl;

cout<<"Edition : "<<edition<<endl; }

void loop(char c)

{ int j;

char ch;

ch = c;

for(j=0; j<80; j++)

cout<<ch; }

int main()

{ books \*ptr,book[20];

ptr = book;

int inc;

int total\_books=0;

int k;

do{

int op;

cout<<endl;

cout<<"Enter 1 for add book"<<endl;

cout<<"Enter 2 for search book"<<endl;

cout<<"Enter 3 for display all books"<<endl;

cout<<"Enter 4 for exit from the program "<<endl;

cout<<endl<<"Option please : ";

cin>>op;

cout<<endl;

inc=0;

switch(op)

{

case 1:  *//add book*

{ book[total\_books].add\_book();

cout<<"\t\t\t One book added..."<<endl;

inc++;

}

break;

case 2: *//search book*

{

string tmp\_auth;

cout<<"Who is the author of the book? please enter : ";

cin>>tmp\_auth;

int i;

int found\_count=0;

for(i=0; i<total\_books; i++)

found\_count = found\_count + book[i].search\_book(tmp\_auth);

if(found\_count > 0)

cout<<"\t\t\t "<<found\_count<<" book(s) found..."<<endl;

else

cout<<"\t\t\t No book found..."<<endl;

}

break;

case 3: *//display all book*

{ for(k=0; k<total\_books; k++)

{ loop('-');

cout<<"Book "<<k+1<<endl;

book[k].display\_books();

loop('-');

cout<<endl;

}

}

break;

case 4:

exit(1); //exit statement

break;

default:

cout<<"Choose right option "<<endl;

} //switch case ends

total\_books = total\_books + inc;

}while(1); //do...while ends

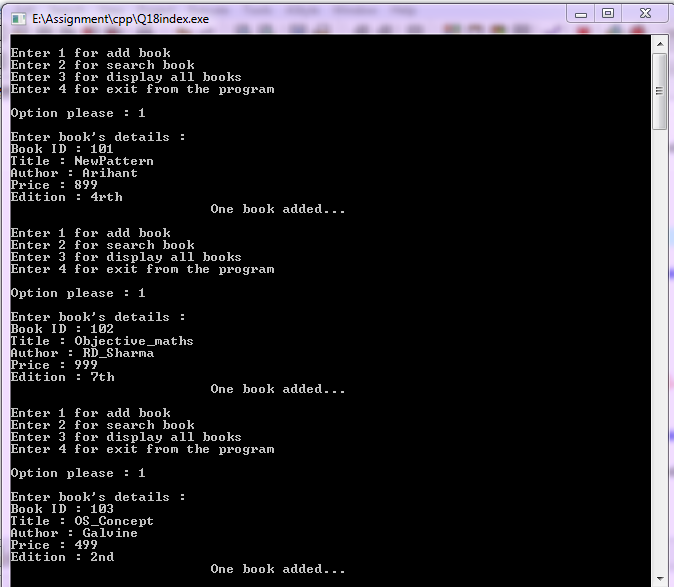
getch();

return 0;

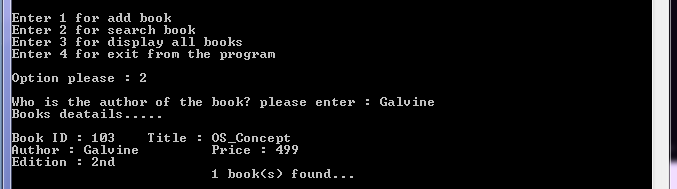
}

**OUTPUT :**

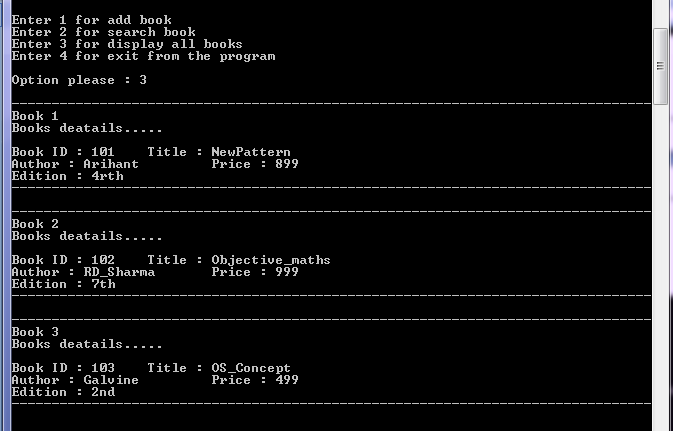
**Adding books :**

****

**Searching book by Author name :**

****

**Displaying all books :**

****

**Q19**. **Define structure student. Structure student has data members for storing name, rollno, name of three subjects and marks. Write member function to store and print data.**

**PROGRAM :**

**//program to store the data of a structure student and print them**

#include<iostream>

#include<conio.h>

using namespace std;

struct student

{

char name[20];

int rollno;

char sub\_name[3][20];

float max\_marks[3],min\_marks[3],obt\_marks[3];

*//Data members of structure*

void getdata()

{

cout<<"Enter name of student and roll no : ";

cin>>name>>rollno;

int i;

for(i=0; i<3; i++)

{

cout<<"Enter subject no "<<i+1<<" name : ";

cin>>sub\_name[i];

cout<<"Enter Maximum marks ,Minimum marks and Obtained marks : ";

cin>>max\_marks[i]>>min\_marks[i]>>obt\_marks[i];

}

}

void disp\_data()

{

cout<<"Name of student : "<<name<<endl;

cout<<"Roll : "<<rollno<<endl;

int i;

for(i=0; i<3; i++)

{ cout<<endl;

cout<<"Name of subject "<<i+1<<" : "<<sub\_name[i]<<endl;

cout<<"Maximum marks : "<<max\_marks[i]<<endl;

cout<<"Minimum marks : "<<min\_marks[i]<<endl;

cout<<"Obtained marks : "<<obt\_marks[i]<<endl;

}

}

*//member function definitions of structure*

};

int main()

{

struct student s1; /*/structure variable created*

s1.getdata();

cout<<endl;

cout<<"Details of Student : "<<endl;

s1.disp\_data();

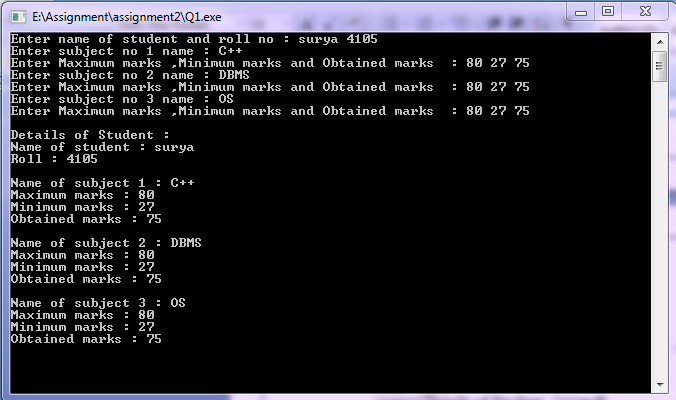
*//accessing the member functions of structure*

getch();

return 0;

}

**OUTPUT :**

****

**Q20.** **Write program to create a class Polar which has data member radius and angle, define overloaded constructor to initialize object and copy constructor to initialize one object by another existing object keep name of parameter of parameterized constructor same as data members. Test function of the program in main function.**

**PROGRAM :**

**//Programm to demonstrate constructor overloading**

#include<iostream>

#include<conio.h>

using namespace std;

class Polar

{

float radius;

float angle;

public:

Polar() *//default constructor*

{

radius = 0;

angle = 0;

cout<<"Defalut constructor invoked "<<endl;

}

Polar(float radius,float angle) *//parameterized constructor*

{ *//formal argument is same as data member*

this->radius = radius;

this->angle = angle; *//this poiter is used to avoid conflict*

cout<<"Parameterized constructor invoked "<<endl;

}

Polar(Polar &p) *//copy constructor*

{

radius = p.radius;

angle = p.angle;

cout<<"Copy constructor invoked "<<endl;

}

*//contructor overloaded*

void display()

{

cout<<"Radius : "<<radius<<endl;

cout<<"Angle : "<<angle<<endl;

}

};

int main()

{

Polar p1;

*//default constructor invoked*

p1.display();

Polar p2(4,5);

*//Parameterized constructor invoked*

p2.display();

Polar p3 = p2;

*//copy constructor invoked*

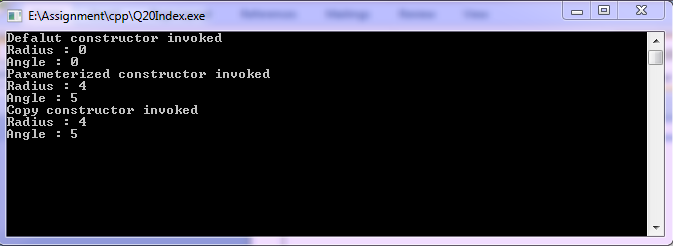
p3.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q21.** **Write program to create a class Polar which has data member radius and angle, use**

**constructor with default arguments to avoid constructor overloading and copy constructor to initialize one object by another existing object keep name of parameter of parameterized**

**constructor same as data members. Test functioning of the program in main function.**

**PROGRAM :**

**//Programm to demonstrate constructor with default arguments**

#include<iostream>

#include<conio.h>

using namespace std;

class Polar

{ float radius;

float angle;

public:

Polar(float radius=0,float angle=0) *//parameterized constructor with defalut arguments*

{  *//formal argument is same as data member*

this->radius = radius;

this->angle = angle; *//this poiter is used to avoid conflict*

cout<<"Parameterized constructor invoked "<<endl;

}

Polar(Polar &p) //copy constructor

{ radius = p.radius;

angle = p.angle;

cout<<"Copy constructor invoked "<<endl;

}

void display()

{ cout<<"Radius : "<<radius<<endl;

cout<<"Angle : "<<angle<<endl; }

};

int main()

{ Polar p1; *//default argument parameterized constructor invoked*

p1.display();

Polar p2(4); *//default argument Parameterized constructor invoked*

p2.display();

Polar p3(2,3); *//Parameterized constructor invoked*

p3.display();

Polar p4 = p3; *//copy constructor invoked*

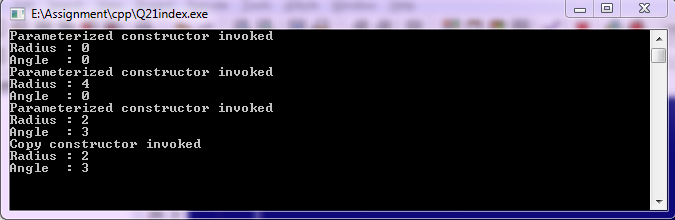
p3.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q22.** **Write a class ArraySort that uses static overloaded function to sort an array of floats, an array of integers.**

**PROGRAM :**

**//Program to sort array of float and int using static overloaded function**

#include<iostream>

#include<conio.h>

#include<string.h>

using namespace std;

class ArraySort

{

public:

static void sort\_array(float f[5]);

static void sort\_array(int f[5]);  *//static member function overloading*

static void swap(float &a,float &b);

static void swap(int &a,int &b);

};

void ArraySort :: sort\_array(float f[5]) *//static member function definition*

{

int i,j;

for(i=0; i<5-1; i++)

{

for(j=0; j<5-1; j++)

{

if(f[j] > f[j+1])

swap(f[j],f[j+1]);  *//nesting static member function*

}

}

cout<<"sorted float array is : "<<endl;

for(i=0; i<5; i++)

cout<<f[i]<<endl;

}

Void ArraySort :: sort\_array(int f[5]) *//static member function overloading*

{

int i,j;

for(i=0; i<5-1; i++)

{

for(j=0; j<5-1; j++)

{

if(f[j] > f[j+1])

swap(f[j],f[j+1]); *//nesting static member function*

}

}

cout<<"sorted integer array is : "<<endl;

for(i=0; i<5; i++)

cout<<f[i]<<endl;

}

void ArraySort :: swap(float &a,float &b)

{

float temp = a;

a = b;

b = temp;

}

Void ArraySort :: swap(int &a,int &b)

{

int temp = a;

a = b;

b = temp;

}

*//member functions definition*

int main()

{

ArraySort a1;

float fary[5];

cout<<"Enter float array : ";

int i;

for(i=0; i<5; i++)

cin>>fary[i];

a1.sort\_array(fary);

*// static member function calling and passing float array*

*//sort\_array(float) invoked*

int ary[5];

cout<<"Enter Integer array : ";

for(i=0; i<5; i++)

cin>>ary[i];

a1.sort\_array(ary);

*// static member function calling and passing int array*

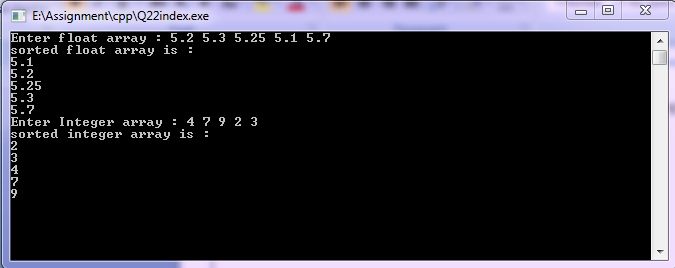
*//sort\_array(int) invoked*

getch();

return 0;

}

**OUTPUT :**

****

**Q23.** **Create a class Counter having a static data member, which keeps track of no. of objects created of type Counter. One static member function must be created to increase value of static data member as the object is created. One static member function must be created to decrease value of static data member as the object is destroyed. One static member function must be created to display the current value of static data member. Use main function to test the class Counter.**

**PROGRAM :**

**//Program to make a class counter having static member functions to keep track count of object (current,when created,when destroyed)**

#include<iostream>

#include<conio.h>

using namespace std;

class counter

{

static int count\_obj;

public:

counter() *// default constructor*

{

cout<<endl<<"..........Ojject created"<<endl;

inc\_count();

*//nested static member function calling*

}

~counter() *//destructor*

{

cout<<endl<<"Object destroyed........"<<endl;

dec\_count();

*//nested static member function calling*

}

static void inc\_count()

{

count\_obj++;

cout<<"Value of count is : "<<count\_obj<<endl;

}

static void dec\_count()

{

count\_obj--;

cout<<"Value of count is : "<<count\_obj<<endl;

}

static void cur\_count()

{

cout<<"Current value of count is : "<<count\_obj<<endl;

}

*//static member functions definition*

};

int counter::count\_obj;

int main()

{ counter c1;

*//object c1 created and constructor invoked (count = 1)*

{

cout<<endl<<"Block start-----> "<<endl;

counter c2;

*//object c2 created and constructor invoked (count = 2)*

counter::cur\_count(); *//current value (count =2)*

*//calling static member function*

cout<<endl<<"<------Block ends"<<endl<<endl;

}

*//object c2 destroyed and destrutor invoked (count = 1)*

counter c3;

*//object c3 created and construtor invoked(count=2)*

*//object c3 destroyed and destrutor invoked (count = 1)*

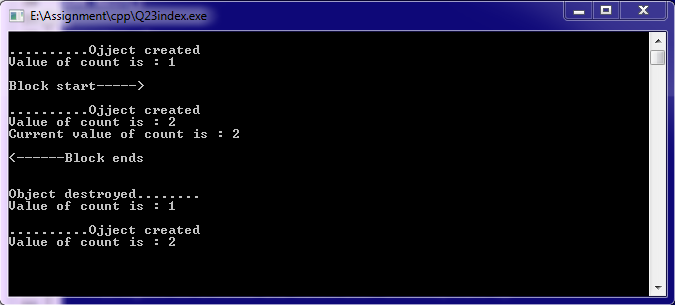
*//object c1 destroyed and destrutor invoked (count = 0)*

getch();

return 0;

}

**OUTPUT :**

****

**Q24. Create a class student. The student class has data members such as roll number, name of student,contact number and address .create the derived class test which contains data members reperesenting name of subject, and test marks of 5 subjects. Display all the information of student.**

**PROGRAM :**

**//program to store data about student and test using class**

#include<iostream>

#include<conio.h>

using namespace std;

class student

{

int rollno;

char name[20];

char contact\_no[20];

char addr[20];

*//Data members*

public:

void get\_student\_data();

void display\_student\_data();

*//member function declaration*

};

class test:public student *//derived class test from base calss student*

{

char sub\_name[5][20];

float marks[5];

*//Data members*

public:

void get\_test\_data();

void display\_test\_data();

*//member function declaration*

};

void student::get\_student\_data() *//student class member function definition*

{ cout<<endl;

cout<<"Enter student details"<<endl;

cout<<"Roll no = ";

cin>>rollno;

cout<<"Name = ";

cin>>name;

cout<<"Contact no = ";

cin>>contact\_no;

cout<<"Address = ";

cin>>addr;

}

void student::display\_student\_data() *//student class member function definition*

{

cout<<endl<<"Details of student... "<<endl<<endl;

cout<<"Roll no = "<<rollno<<"\t\t\t Name = "<<name<<endl;

cout<<"Contact no = "<<contact\_no<<"\t\t\t Address = "<<addr<<endl;

}

void test::get\_test\_data() *//test class member function definition*

{

int i;

for(i=0; i<5; i++)

{

cout<<"Enter subject "<<i+1<<" name and Test marks : ";

cin>>sub\_name[i]>>marks[i];

}

}

void test::display\_test\_data() *//test class member function definition*

{

int i;

cout<<endl<<"Test marks are :"<<endl<<endl;

for(i=0; i<5; i++)

{

cout<<"Subject "<<i+1<<endl;

cout<<"name = "<<sub\_name[i]<<endl;

cout<<"Marks = "<<marks[i]<<endl;

cout<<endl;

}

}

int main()

{

test t1; *// Crteating object of derived class*

t1.get\_student\_data(); *//calling base class member function*

t1.get\_test\_data();  *//own member function calling*

t1.display\_student\_data(); *//calling base class member function*

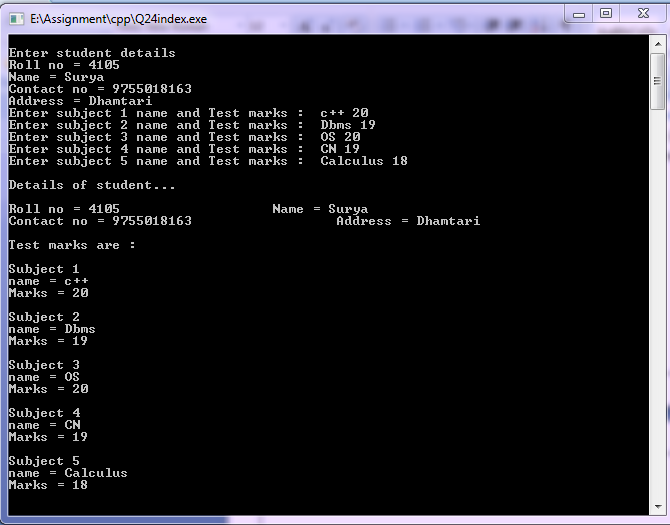
t1.display\_test\_data();  *//own member function calling*

getch();

return 0;

}

**OUTPUT :**

****

**Q25.** **Write a program in c++ for multiple inheritance using book as derived class having different base classes Journals,Magzines,Newpaper.**

**PROGRAM :**

**//Program to demonstrate multiple inheritence**

#include<iostream>

#include<conio.h>

using namespace std;

class Journals *//base class 1 definition*

{ char journal\_name[20];

float price;

public:

void get\_data()

{ cout<<endl<<"Enter journal name : ";

cin>>journal\_name;

cout<<"Enter price : ";

cin>>price;

}

void disp\_data()

{ cout<<endl<<"journal name : "<<journal\_name<<endl;

cout<<"price : "<<price<<endl;

}

*//member function definitions*

};

class Magzines *//base class 2 definition*

{

char magz\_name[20];

float price;

public:

void get\_data()

{

cout<<endl<<"Enter Magzine name : ";

cin>>magz\_name;

cout<<"Enter price : ";

cin>>price;

}

void disp\_data()

{ cout<<endl<<"Magzine name : "<<magz\_name<<endl;

cout<<"price : "<<price<<endl;

}

*//member function definitions*

};

class Newspaper *//base class 3 definition*

{ char news\_name[20];

float price;

public:

void get\_data()

{

cout<<endl<<"Enter Newspaper name : ";

cin>>news\_name;

cout<<"Enter price : ";

cin>>price;

}

void disp\_data()

{

cout<<endl<<"Newspaper name : "<<news\_name<<endl;

cout<<"price : "<<price<<endl;

}

*//member function definitions*

};

class book:public Journals,public Magzines,public Newspaper

{ *//derived class definition*

*//multiple inheritence*

char book\_what[20];

public:

void booking\_what()

{

cout<<"What you want to book... "<<endl;

cout<<"Journals or Magzines or Newspaper : ";

cin>>book\_what;

}

void get\_book\_data()

{

if(book\_what == "Journals")

Journals::get\_data();  *//fun. overriding*

else if(book\_what == "Magzines")

Magzines::get\_data(); *//fun. overriding*

else if(book\_what == "Newspaper" )

Newspaper::get\_data();  *//fun. overriding*

}

void display\_booked()

{

if(book\_what == "Journals")

Journals::disp\_data(); *//fun. overriding*

else if(book\_what == "Magzines")

Magzines::disp\_data(); *//fun. overriding*

else if(book\_what == "Newspaper")

Newspaper::disp\_data(); *//fun. overriding*

}

};

int main()

{

book obj; *//Object of derived class*

char op;

do{

obj.booking\_what();

obj.get\_book\_data();

obj.display\_booked(); //accessing own mwmber function

cout<<endl<<"Press Y to continue : ";

cin>>op;

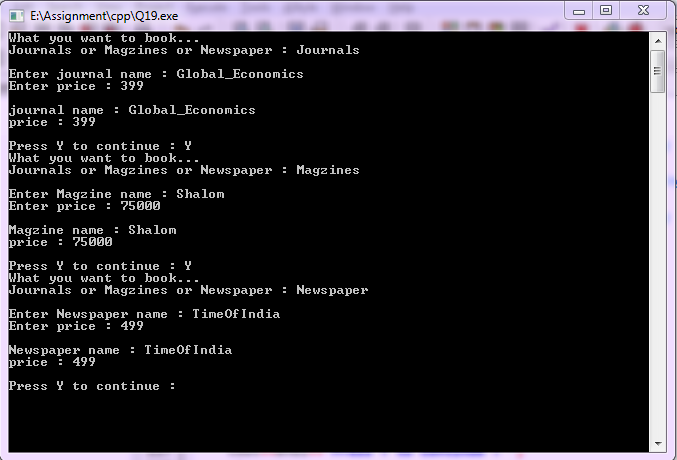
}while( op == 'y' || op == 'Y');

getch();

return 0;

}

**OUTPUT :**

****

**Q26.** **Consider an example of declaring the examination result.design 3 classes student,exam,result. The student class has data members such as that reperesenting number, name of student ,create the class exam,which contains data members reperesenting name of subject,minmum marks,maximum marks, obtained marks for 3 subject derive class result from both student and exam classes. Test the result class in main function ?**

**PROGRAM :**

**//Program to demostrate three classes(student , exam and result ) which is in multiple inheritance**

#include<iostream>

#include<conio.h>

using namespace std;

class student *//base class 1*

{

int rollno;

char name[20];  *//Data members*

public:

void get\_student\_data()

{ cout<<"Enter roll no and name of student : ";

cin>>rollno>>name;

}

void show\_student\_data()

{

cout<<"Roll no = "<<rollno<<"\t Name = "<<name<<endl;

}

*//Member function declaration*

};

class exam *// base class 2*

{

protected:

char sub\_name[3][20];

float min\_marks[3],max\_marks[3],obt\_marks[3];

*//Data members in protected mode*

public:

void get\_exam\_data();

void display\_exam\_data();

*//Member function* *declaration*

};

class result:public student,public exam *//multiple inheritence*

{

float total\_max;

float total\_obt,per;

*//Data members*

public:

result*() //Default constructor*

{ total\_max = 0;

total\_obt = 0;

per = 0; }

void get\_result();

void display\_result();

*//Member function declaration*

};

void exam::get\_exam\_data() *//Member function definition of exam*

{ int i;

for(i=0; i<3; i++)

{

cout<<"Enter subject "<<i+1<<" name = ";

cin>>sub\_name[i];

cout<<"Minimum marks = ";

cin>>min\_marks[i];

cout<<"Maximum marks = ";

cin>>max\_marks[i];

cout<<"Obtained marks = ";

cin>>obt\_marks[i];

}

}

void exam:: display\_exam\_data()  *//Member function definition of exam*

{

int i;

for(i=0; i<3; i++)

{

cout<<"Subject "<<i+1<<" name = "<<sub\_name[i]<<endl;

cout<<"Minimum marks = "<<min\_marks[i]<<endl;

cout<<"Maximum marks = "<<max\_marks[i]<<endl;

cout<<"Obtained marks = "<<obt\_marks[i]<<endl;

}

}

void result::get\_result() *//Member function definition of result*

{

int i;

for(i=0; i<3; i++)

{

total\_max = total\_max + max\_marks[i];

total\_obt = total\_obt + obt\_marks[i];

}

per = total\_obt \* 100 / total\_max;

}

void result::display\_result() *//Member function definition of result*

{

cout<<"Result of student is :"<<endl;

cout<<"Total obtained marks in "<<total\_max<<" is = "<<total\_obt<<endl;

cout<<"Percentage is = "<<per<<"%"<<endl;

if(per>=70)

cout<<"First Division...."<<endl;

else if(per<70 && per>=50)

cout<<"Second Division..."<<endl;

else if(per<50 && per>=33)

cout<<"Third Division..."<<endl;

else

cout<<"Fail..."<<endl;

}

int main()

{

result res1; *//Object created and initialized of type(result) derived class*

res1.get\_student\_data(); *//Accessing base class 1 member function*

cout<<endl;

res1.get\_exam\_data(); *//Accessing base class 2 member function*

res1.get\_result(); *//Accessing own member function*

cout<<endl;

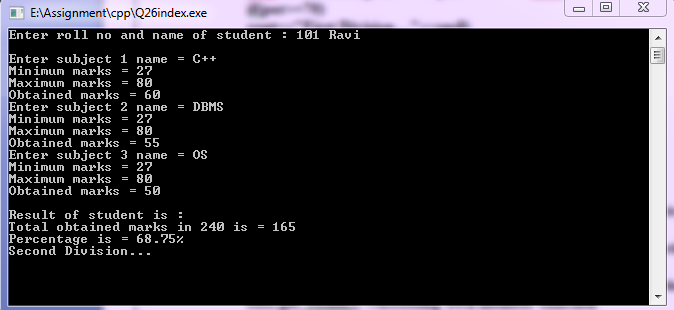
res1.display\_result();

getch();

return 0;

}

**OUTPUT :**

****

**Q27.** **WAP to generate fibbonacci series use the concept of function overriding.**

**PROGRAM :**

**//Program for fibbonacci series using function overriding**

#include<iostream>

#include<conio.h>

using namespace std;

class base //base class

{ public:

void fibbo(int); *//member fun. declaration*

};

void base::fibbo(int n) *//member fun. definition*

{ int n1=0,n2=1,n3;

int i;

cout<<"Series is : "<<endl<<endl;

cout<<n1<<" "<<n2;

for(i=0; i<n-2; i++)

{ n3=n1+n2;

cout<<" "<<n3;

n1 = n2;

n2 = n3; }

cout<<endl<<"This is base class fibbonacci.."<<endl<<endl;

}

class derived:public base *//publically derivation of base class*

{

public:

int fibbo(int n) *//same name as base class function*

{ if(n==0 || n==1)

return n;

else

return fibbo(n-1)+fibbo(n-2);

}  *//member fun. Definition*

};

int main()

{ derived d; *//object of derived class*

int n;

cout<<"How many terms you want : ";

cin>>n;

d.base::fibbo(n); *//function overriding*

int i=0;

cout<<endl<<endl;

cout<<"How many terms you want : ";

cin>>n;

cout<<"Series is :"<<endl<<endl;

while(i<n){

cout<<" "<<d.derived::fibbo(i); *//function overriding*

i++;

}

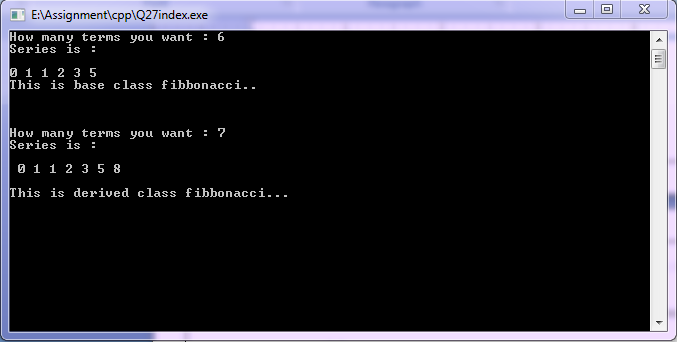
cout<<endl<<endl<<"This is derived class fibbonacci..."<<endl;

getch();

return 0;

}

**OUTPUT :**

****

**Q28.** **Write a program to solve Diamond problem(Hybrid inheritance and virtual base class).**

**PROGRAM :**

**//Program to solve Diomand problem**

#include<iostream>

#include<conio.h>

using namespace std;

class base *//base class*

{

public:

void display\_base()

{

cout<<"This is base class"<<endl;

}

};

class mid\_base1: virtual public base *//virtual base class*

{

public:

*//display\_base() inherited from base class*

void display\_mid1()

{

cout<<"This is intermediate base class1"<<endl;

}

};

class mid\_base2:public virtual base //vitual base class

{

public:

*//display\_base() inherited from base class*

void display\_mid2()

{

cout<<"This is intermediate base class2"<<endl;

}

};

class derived:public mid\_base1,public mid\_base2

{

public:

*//display\_mid1() inheruted from mid\_base1 class*

*//display\_base() inherited from mid\_base2 class*

*//which display\_base() should be inherited from mid\_base1 or from mid\_base2*

*//problem solved because base class is virtual otherwise it shows ambigiuty*

void display\_derived()

{

cout<<"This is derived class"<<endl;

}

};

int main()

{

derived d; *//created object of derived class*

d.display\_base();

d.display\_mid1();

d.display\_mid2();

*//accessing inherited member functions*

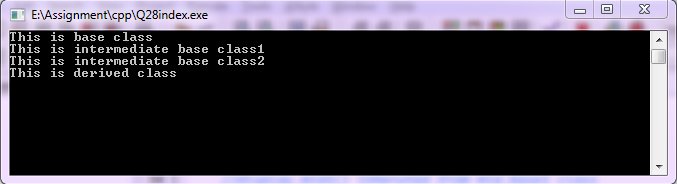
d.display\_derived();

getch();

return 0;

}

**OUTPUT :**

****

**Q29. Write a program in c++ using constructor and destructor in Multiple and multilevel inheritance.**

**PROGRAM :**

**//Program to demostrate multiple and multilevel inheritance**

#include<iostream>

#include<conio.h>

using namespace std;

class base1

{

int a;

public:

base1(int a1) *//constructor of base class*

{

a = a1;

cout<<"This is base1 class contructor"<<endl;

}

~base1() *//destructor*

{

cout<<"This is base1 class destructor"<<endl;

}

void display()

{

cout<<"Value of a = "<<a<<endl;

}

};

class base2

{

int b;

public:

base2(int b1) *//contrutor of base class*

{

b = b1;

cout<<"This is base2 class contructor"<<endl;

}

~base2() *//destructor*

{

cout<<"This is base2 class destructor"<<endl;

}

void display()

{

cout<<"Value of b = "<<b<<endl;

}

};

class mid\_base1 : public base1,public base2 //multiple inheritance

{

int c;

public:

mid\_base1(int c1,int c2,int c3):base1(c2),base2(c3)

{  *//base classes contructor calling(multiple inheri.)*

*//3 argumnts passed to mid\_base1*

*//c2 passed to base1 and c3 passed to base2*

c = c1;

cout<<"This is mid\_base1 class contructor"<<endl;

}

~mid\_base1() *//destructor*

{

cout<<"This is mid\_base2 class destructor"<<endl;

}

void display()

{

cout<<"Value of c = "<<c<<endl;

}

};

class mid\_base2:public base1 //mid\_base2 is derived from base1 class

{

int d;

public:

mid\_base2(int d1,int d2):base1(d2) *//calling constructor of base of mid\_base2 class*

{ *//2 arguments passed to mid\_base2*

*//d2 passed to class base1*

d = d1;

cout<<"This is mid\_base2 class contructor"<<endl;

}

~mid\_base2() *//destructor*

{

cout<<"This is mid\_base2 class destructor"<<endl;

}

void display()

{

cout<<"Value of d = "<<d<<endl;

}

};

class derived: public mid\_base2 *//multilevel inheritance*

{

int e;

public:

derived(int e1,int e2,int e3):mid\_base2(e2,e3)

*//base classes contrutors calling(multilevel)*

{ *//Passing 3 arguments to derived class*

*//e2 and e3 passed to class mid\_base2*

e = e1;

cout<<"This is derived class contructor"<<endl;

}

~derived() *//destructor*

{

cout<<"This is derived class destructor"<<endl;

}

void display()

{

cout<<"Value of e = "<<e<<endl;

}

};

int main()

{

cout<<"This is multilevel inheritance : "<<endl<<endl;

{

*//multileval inheritance*

derived d(2,3,4); *//passing three agrument to derivered class object*

d.base1::display(); *//fun. overriding*

d.mid\_base2::display();

d.derived ::display();

}

cout<<endl<<endl;

cout<<"This is multiple inheritance : "<<endl<<endl;

{ *//multiple inheritance*

mid\_base1 m(6,7,8);  *//passing three agrument to derivered class object*

m.base1::display();

m.base2::display();

m.mid\_base1::display();

*//funcvtion overriding*

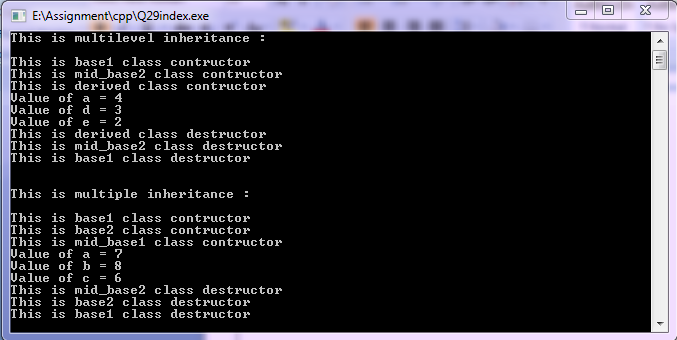
}

getch();

return 0;

}

**OUTPUT :**

****

**Q30.** **Write a program in c++ to demonstrate pointer to an object and this pointer.**

**PROGRAM :**

**//Program to demonstrate pointer to an object and this pointer**

#include<iostream>

#include<conio.h>

using namespace std;

class student

{

int rollno;

char name[20];

*//Data members*

public:

student() { } *//Default constructor*

student(int rollno,char name[20]) *//Parameterized constructor*

{

this->rollno = rollno; *//this pointer stores address of caller object*

this->name = name;

}

void display() *//member function definition*

{

cout<<endl<<"Roll no = "<<rollno<<endl;

cout<<"Name = "<<name<<endl;

}

};

int main()

{

int rollno1;

char name1[20];

cout<<"Enter Roll no = ";

cin>>rollno1;

cout<<"Enter name = ";

cin>>name1;

student s(rollno1,name1);

*//object created and parameterized constructor invoked*

student \*ptr; *//pointer of type student*

ptr = &s;

*//Pointer to an object*

cout<<endl<<"Accessing through pointer to an object "<<endl;

ptr->display();

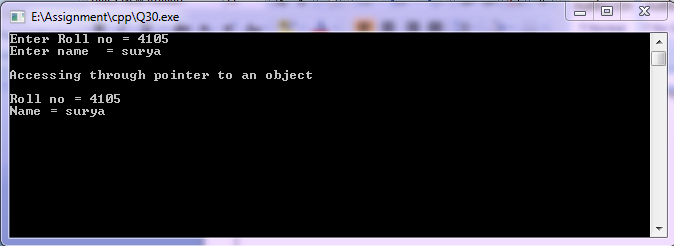
*//Accessing member function through pointer to an object*

getch();

return 0;

}

**OUTPUT :**

****

**Q31.** **Write a program in c++ to demonstrate pointer to derived class.**

**PROGRAM :**

**//Program to demostrate pointer to derived class**

#include<iostream>

#include<conio.h>

using namespace std;

class base

{

public:

void disp()

{

cout<<"This is base class"<<endl;

}

};

class derived:public base //single imheritance

{

public:

void disp()

{

cout<<"This is derived class"<<endl;

}

};

int main()

{

base \*bptr,b; *//bptr is a pointer of type base class*

derived d,\*dptr; *//dptr is a pointer of type derived class*

bptr = &d; *//points address of derived class object*

cout<<"Base pointer holds address of deriverd class object "<<endl;

bptr->disp();

*//disp() of base class cause of bptr = &d is just ignoured by compiler at compile time*

cout<<endl;

dptr = &d; *//points address of own class object*

cout<<"Derived pointer holds address of deriverd class object "<<endl;

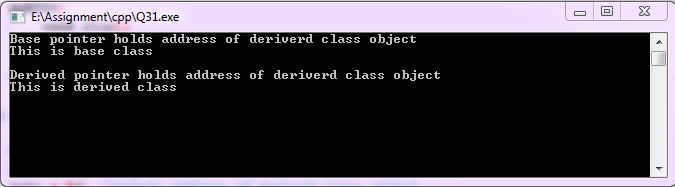
dptr->disp(); *//disp() of derived class*

getch();

return 0;

}

**OUTPUT :**

****

**Q32.** **Create a program having pointer to void to store address of integer variable then print value of integer variable using pointer to void. Perform the same operation for float variable.**

**PROGRAM :**

**//Program to demonstrate void pointer**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

void \*ptr;  *//void pointer*

int a=76;

ptr = &a; *//ptr holds address of a*

cout<<"Value of a is : "<<a<<endl;

cout<<"Value stored in which ptr points to is : "<<\*(int\*)ptr<<endl;

*//type casting of void pointer to int*

float b=90.99;

ptr = &b;

cout<<"Value of b is : "<<b<<endl;

cout<<"Value stored in which ptr points to is : "<<\*(float\*)ptr<<endl;

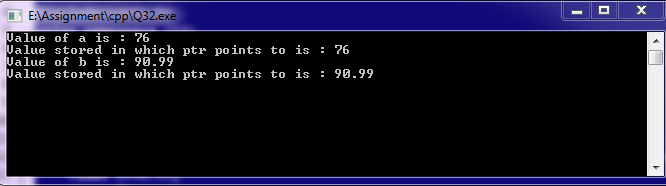
*//type casting of void pointer to float*

getch();

return 0;

}

**OUTPUT :**

****

**Q33.** **Create a class account that stores customer name, account number and type of account. From this derive the classes cur\_acct and sav\_acct to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:**

**a) Accept deposit from customer.**

**b) Display the balance**

**PROGRAM :**

**//Program to accept deposit and display the balance of saving or current account**

#include<iostream>

#include<conio.h>

using namespace std;

class Account *//base class*

{

protected:

char cust\_name[30],acct\_type[30];

int account\_number;

float amount;

public:

Account() *//default constructor*

{

amount=2000;

}

void get\_detail() *//member function definition*

{

cout<<"Enter customer name : ";

cin>>cust\_name;

cout<<"Enter account number : ";

cin>>account\_number;

}

};

class cur\_acct:public Account *//derived class*

{

float deposit;

public:

void current();

void get\_deposit();

void show\_deposit(); *//member fun. declaration*

};

void cur\_acct::current() *//member fun. definition*

{

get\_detail();

cout<<"\nWeicome "<<cust\_name<<"...."<<endl;

cout<<"It is your current account"<<endl;

cout<<"account no. : "<<account\_number<<endl;

cout<<"my amount : "<<amount<<endl;

}

void cur\_acct:: get\_deposit()

{

cout<<"enter deposit amount : ";

cin>>deposit;

amount=amount+deposit;

}

void cur\_acct:: show\_deposit()

{

cout<<"deposit amount is : "<<deposit<<endl;

cout<<"current balance : "<<amount<<endl;

}

class sav\_acct:public Account

{

float deposit;

public:

void saving();

void get\_deposit();

void show\_deposit(); //member fun. declaration

};

void sav\_acct::saving() //member fun. defintion

{

get\_detail();

cout<<"\nWeicome "<<cust\_name<<"...."<<endl;

cout<<"It is your saving account"<<endl;

cout<<"account no. : "<<account\_number<<endl;

cout<<"My amount : "<<amount<<endl;

}

void sav\_acct:: get\_deposit()

{

cout<<"Enter deposit amount : ";

cin>>deposit;

amount=amount+deposit;

}

void sav\_acct:: show\_deposit()

{

cout<<"Deposit amount is : "<<deposit<<endl;

cout<<"Current balance : "<<amount<<endl;

}

int main()

{

int i;

cur\_acct obj1;

sav\_acct obj2; //objects of derived class

do{

cout<<endl<<endl;

cout<<"Enter 1 for saving account."<<endl;

cout<<"Enter 2 for current account."<<endl;

cout<<"Enter 3 to exit."<<endl;

cout<<"Enter option : ";

cin>>i;

switch(i)

{

case 1:

obj1.current();

obj1.get\_deposit();

obj1.show\_deposit();

break;

case 2:

obj2.saving();

obj2.get\_deposit();

obj2.show\_deposit();

break;

case 3:

exit(0);

default:

cout<<"Enter valid option..."<<endl;

}

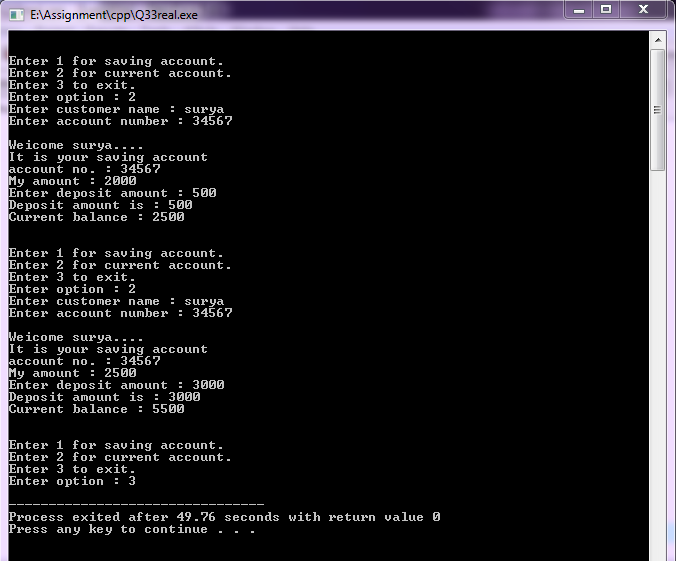
}while(1);

getch();

return 0;

}

**OUTPUT :**

****

**Q34.** **Create a class circle with data member radius; provide member function to calculate area. Derive a class sphere from class circle; provide member function to calculate volume.Derive class cylinder from class sphere with additional data member for height and member function to calculate volume.**

**PROGRAM :**

**//Program to calculate volume of cylinder and sphere and area of circle through multilevel inheritance**

#include<iostream>

#include<conio.h>

using namespace std;

const float pi=22/7.0; *//constant variable pi*

class circle *//base class*

{

protected:

float radius;

*//Data member in protected mode*

public:

circle(float radius) *//parameterized constructor*

{

this->radius = radius;

}

void circle\_area() *//member function definition*

{

cout<<"Area of circle with radius "<<radius<<" is : "<<(pi) \* radius \* radius<<endl;

}

};

class sphere:public circle *//intermediate base class*

{

public:

sphere(float r):circle(r) *//constructor calling statement for base class*

{

}

void sphere\_volume() *//member function defintion*

{

float volume = (4.0/3)\*(pi)\*radius\*radius\*radius;

cout<<"Volume of Sphere with radius "<<radius<<" is : "<<volume<<endl;

}

};

class cylinder:public sphere *//derived class (multilevel inheritance)*

{

float height; *//data member*

public:

cylinder(float r,float height):sphere(r)

*//r passed to base class and height initialized with data member height*

{

this->height = height;

}

cylinder(float r):sphere(r) *//only radius is passed to base class*

{

height=0;

}

void cylinder\_volume() *//member function definition*

{

float volume = pi \* radius \* radius \* height;

cout<<"Volume of Cylinder with radius "<<radius<<" and height "<<height<<" is : "<<volume<<endl;

}

};

int main()

{

float r;

cout<<"Enter Radius of Circle : ";

cin>>r;

cout<<endl;

cylinder obj1(r); *//constructor cylinder(float) invoked*

*//object of derived(cylinder) class created and radius passed to base class(circle*)

obj1.circle\_area(); *//calling member fun. of base (circle) class*

cout<<endl<<endl;

cout<<"Enter Radius of Sphere : ";

cin>>r;

cout<<endl;

cylinder obj2(r); *//constructor cylinder(float) invoked*

obj2.sphere\_volume(); *//calling member fun. of mid base (sphere) class*

cout<<endl<<endl;

float h;

cout<<"Enter Radius and Height of Cylinder : ";

cin>>r>>h;

cout<<endl;

cylinder obj3(r,h); *//constructor cylinder(float,float) invoked*

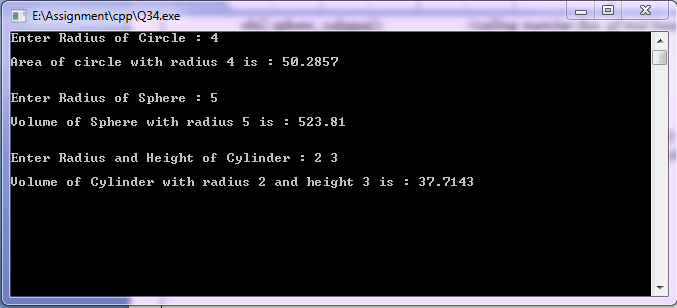
obj3.cylinder\_volume(); *//calling member fun. of own (cylinder) class*

getch();

return 0;

}

**OUTPUT :**

****

**Q35.** **Write a program in c++ for overloading of unary operator.**

**PROGRAM :**

**//Program to overload unary operator**

#include<iostream>

#include<conio.h>

using namespace std;

class vector

{

float x;

float y;

float z; //data members

public:

vector(){ } *//default constructor*

vector(float x,float y,float z) *//parameterized constructor*

{

this->x = x;

this->y = y;

this->z = z;

}

void display(); *//member functions declaration*

void operator-(); *//- operator overloading declaration*

void operator++(); *//++ operator overloading declaration*

void operator--(); //-- operator overloading declaration

};

void vector::display() *//member function definition*

*{*

cout<<"Vector : "<<x<<"i + "<<y<<"j + "<<z<<"k"<<endl;

}

void vector::operator-() *//- operator overloading definition*

{

x = -x;

y = -y;

z = -z;

}

void vector::operator++() *//++ operator overloading definition*

{

++x;

++y;

++z;

}

void vector::operator--()  *//-- operator overloading definition*

{

--x;

--y;

--z;

}

int main()

{

float a,b,c;

cout<<"Enter three values : ";

cin>>a>>b>>c;

vector v1(a,b,c); *//object created and parameterized constructor invoked*

v1.display(); *//member function calling*

cout<<"-v1 : "<<endl;

-v1; *//operator- function calling*

v1.display();

cout<<"++v1 : "<<endl;

++v1; *//operator++ function calling*

v1.display();

cout<<"--v1 : "<<endl;

--v1;  *//operator-- function calling*

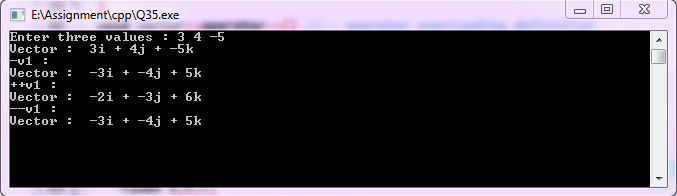
v1.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q36.** **Write a program in c++ for overloading of binary operator.**

**PROGRAM :**

**//Program for overloading binary operators**

#include<iostream>

#include<conio.h>

using namespace std;

class sample //class definition

{ float x;

float y;

float z; *//data members*

public:

void get\_sample(); *//member function declaretion*

*//operator overloading*

sample operator+(sample); *//operator+(binary) member function declaration*

sample operator\*(sample); *//operator\*(binary) member function declaration*

sample operator/(sample); *//operator/ (binary) member function declaration*

bool operator==(sample); *//operator== (binary) member function declaration*

void disp\_sample(); *//member functiom declaration*

};

void sample::get\_sample() *//member functiom definition*

{ cout<<"Enter three value : ";

cin>>x>>y>>z; }

void sample::disp\_sample()  *//member functiom declaration*

{ cout<<"x = "<<x<<"\t y = "<<y<<"\t z = "<<z<<endl<<endl; }

sample sample::operator+(sample s) *//operator+(binary) member function definition*

{ sample temp;

temp.x = x + s.x;

temp.y = y + s.y;

temp.z = z + s.z;

return temp;

}

sample sample::operator\*(sample s) *//operator\* (binary) member function definition*

{ sample temp;

temp.x = x \* s.x;

temp.y = y \* s.y;

temp.z = z \* s.z;

return temp;

}

sample sample::operator/(sample s) *//operator/ (binary) member function definition*

{ sample temp;

temp.x = x / s.x;

temp.y = y / s.y;

temp.z = z / s.z;

return temp; }

bool sample::operator==(sample s) *//operator== (binary) member function definition*

{ if(x==s.x && y==s.y && z==s.z)

return true;

else

return false;

}

int main()

{

sample samp1,samp2,samp3;

samp1.get\_sample();

samp2.get\_sample();

cout<<"samp1 + samp2 = "<<endl;

samp3=samp1+samp2; *//calling operator+ function*

*//same as samp3 = samp1.operator+(samp2);*

samp3.disp\_sample();

cout<<"samp1 \* samp2 = "<<endl;

samp3=samp1 \* samp2; *//calling operator\* function*

samp3.disp\_sample();

cout<<"samp1 / samp2 = "<<endl;

samp3=samp1 / samp2;  *//calling operator/ function*

samp3.disp\_sample();

samp1.get\_sample();

samp2.get\_sample();

if(samp1==samp2) *//calling operator== function*

cout<<"Both are equal "<<endl;

else

cout<<"Both are not equal "<<endl;

samp1.get\_sample();

samp2.get\_sample();

if(samp1==samp2) *//calling operator== function*

cout<<"Both are equal "<<endl;

else

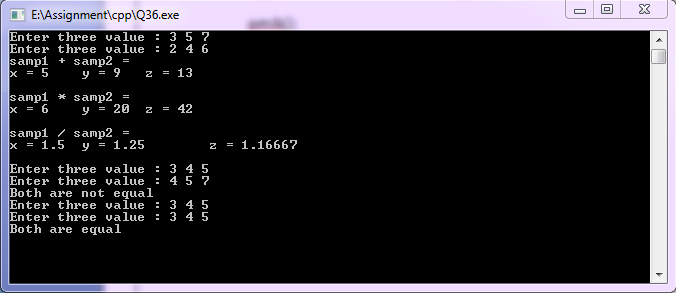
cout<<"Both are not equal "<<endl;

getch();

return 0;

}

**OUTPUT :**



**Q37.** **Create class Polar having data members radius and angle.It contains member functions for taking input in data member function for displaying value of data members . Class Polar contains declaration of friend function add which accepts two objects of class Polar and returns objects of class Polar after addition.Test the class using main function and object of class Polar.**

**PROGRAM :**

**//Program to add two objects of a class using friend function**

#include<iostream>

#include<conio.h>

using namespace std;

class Polar

{

float radius;

float angle; //data members

public:

void input()

{

cout<<"Enter radius : ";

cin>>radius;

cout<<"Enter angle : ";

cin>>angle;

}

void display() *//member fun. definition*

{

cout<<"Radius = "<<radius<<endl;

cout<<"Angle = "<<angle<<endl;

}

friend Polar add(Polar,Polar); *//friend function declaration*

};

Polar add(Polar p1,Polar p2) *//friend function definition*

{

Polar temp;

temp.radius = p1.radius + p2.radius;

temp.angle = p1.angle + p2.angle;

return temp;

*//Adding two Polar object then return a Polar object*

}

int main()

{

Polar p1,p2,addition; *//objects created*

p1.input();

p2.input();

cout<<endl;

addition = add(p1,p2); *//calling friend function add(Polar,Polar)*

cout<<endl<<"First object : "<<endl;

p1.display();

cout<<endl<<"Second object : "<<endl;

p2.display();

cout<<endl<<"Addition of first & second : "<<endl;

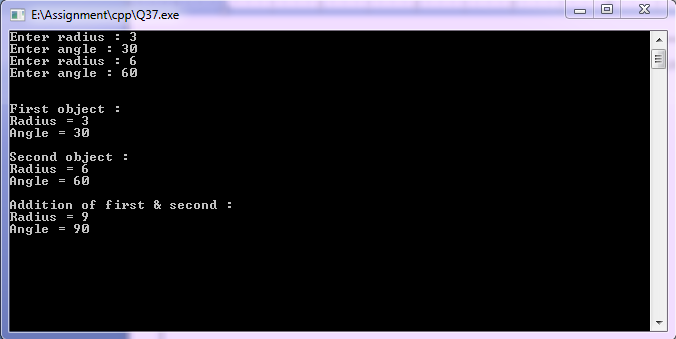
addition.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q38. Write program to create a class distance having data members feet and inch (A single object will store distance in form such as 5 feet 3 inch).**

**It contains member functions for taking input in data members and member function for displaying value of data members.**

**Class Distance contains declaration of friend finction add which accepts two objcts of class distance and return objects of class Distance after addition .**

**Class Distance contains declaration of another friend finction Subtract that accepts two objcts of class distance and return objects of class Distance after subtraction.**

**Test the class using main function and objects of class Distance.**

**PROGRAM :**

**//Program to add and subtract two Distance objects using friend function**

#include<iostream>

#include<conio.h>

#include<math.h>

using namespace std;

class Distance

{

int feet;

int inch; *//Data members*

public:

void input()

{

cout<<"Enter Feet : ";

cin>>feet;

cout<<"Enter Inch : ";

cin>>inch;

}

void display() //member fun. definition

{

cout<<"Distance is = "<<abs(feet)<<" Feet "<<abs(inch)<<" Inch"<<endl;

}

friend Distance add(Distance,Distance);

friend Distance subtract(Distance,Distance);

//*Friend function declaration*

};

Distance add(Distance d1,Distance d2) //*friend function definition*

{

Distance temp;

temp.feet = d1.feet + d2.feet;

temp.inch = d1.inch + d2.inch;

if(temp.inch>12)

{

temp.feet = temp.feet + temp.inch / 12;

temp.inch = temp.inch % 12;

}

return temp;

}

Distance subtract(Distance d1,Distance d2) *//friend function definition*

{

Distance temp;

if(d1.feet>d2.feet)

{

if(d1.inch<d2.inch)

{

d1.feet = d1.feet - 1;

d1.inch = d1.inch + 12;

}

temp.feet = d1.feet - d2.feet;

temp.inch = d1.inch - d2.inch;

}

if(d1.feet<d2.feet)

{

if(d1.inch>d2.inch)

{

d2.feet = d2.feet - 1;

d2.inch = d2.inch + 12;

}

temp.feet = d2.feet - d1.feet;

temp.inch = d2.inch - d1.inch;

}

return temp;

}

int main()

{

Distance dis1,dis2,addition,subtraction;

dis1.input();

dis2.input();

addition = add(dis1,dis2);  *//friend function calling*

subtraction = subtract(dis1,dis2); *//friend function calling*

cout<<endl<<"First Distance object : ";

dis1.display();

cout<<endl<<"Second Distance object : ";

dis2.display();

cout<<endl<<"Addition : ";

addition.display();

cout<<endl<<"Subtraction : ";

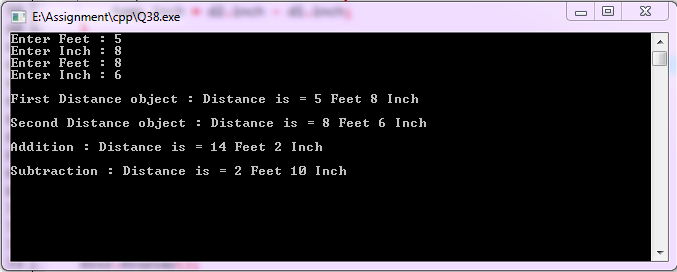
subtraction.display();

getch();

return 0;

}

**OUTPUT :**

****

**Q39.** **Write a program to create class Mother having data member to store salary of Mother,create another class Father having data member to store salary of Father.**

**Write a friend function ,which accepts objects of class Mother , and Father and Prints Sum of salary of Mother and Father objects.**

**PROGRAM :**

**//Program to add salary of Mother and Father using friend function**

#include<iostream>

#include<conio.h>

using namespace std;

class Father; *//forward declaration of class Father*

class Mother

{

float salary;

public:

void input()

{

cout<<"Enter salary of Mother : ";

cin>>salary;

}

friend void add\_salary(Mother,Father);

*//friend function declaration passing Mother and Father type of objcts an argument*

};

class Father

{

float salary;

public:

void input()

{

cout<<"Enter salary of Father : ";

cin>>salary;

}

friend void add\_salary(Mother,Father);  *//friend function declaration*

};

void add\_salary(Mother m,Father f) *//friend function definition*

{

float total\_sal = m.salary + f.salary ;

cout<<"Total salary of Mother and Father is : "<<total\_sal<<endl;

}

int main()

{

Mother m1;  *//object created for mother class*

Father f1; *//object created for father class*

m1.input();

f1.input();

cout<<endl<<endl;

add\_salary(m1,f1);

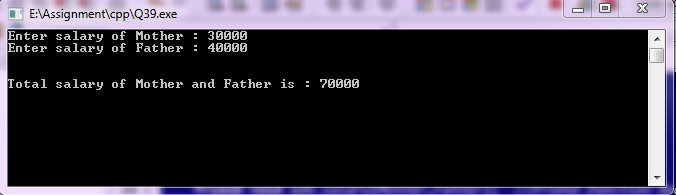
*//calling friend function and passing objects of mother and father*

getch();

return 0;

}

**OUTPUT :**

****

**Q40.** **Write a program to create class having data member to store salary of Mother , create another class Father having data member to store salary of Father. Declare class Father to be friend class of Mother. Write a member function in Father, which accepts object of class Mother and prints Sum of Salary of Mother and Father Objects. Create member function in each class to get input in data member and to display the value of data member.**

**PROGRAM :**

**//Program to add salary of Mother and Father using friend class**

#include<iostream>

#include<conio.h>

using namespace std;

class Father; *//forward declaration of class Father*

class Mother

{

float salary;

public:

void input()

{

cout<<"Enter salary of Mother : ";

cin>>salary;

}

void display()

{

cout<<"salary of Mother : "<<salary<<endl;

}

friend class Father;

*//friend class declaration*

};

class Father

{

float salary;

public:

void input()

{

cout<<"Enter salary of Father : ";

cin>>salary;

}

void display()

{

cout<<"salary of Father : "<<salary<<endl;

}

void add\_salary(Mother); *//Passing Mother type of object an argument of fun.*

};

void Father::add\_salary(Mother m) *//friend function definition*

{

float total\_sal = m.salary + salary ;

cout<<"Total salary of Mother and Father is : "<<total\_sal<<endl;

}

int main()

{

Mother m1;  *//object created for mother class*

Father f1; *//object created for father class*

m1.input();

f1.input();

cout<<endl<<endl;

m1.display();

f1.display();

f1.add\_salary(m1);

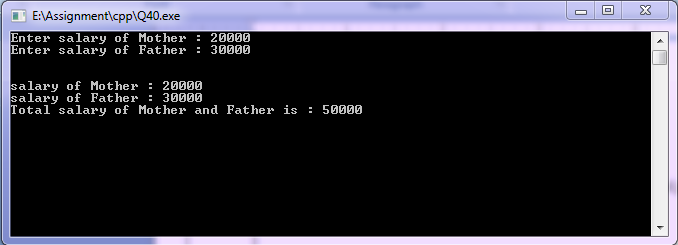
*//calling friend function by class Father object and passing objects of mother*

getch();

return 0;

}

**OUTPUT :**

****

**Q41. Create a base class shape having two data members with two-member function getdata (pure virtual function) and printarea (not pure virtual function).**

**Derive classes triangle and rectangle from class shape and redefine member function printarea in both classes triangle and rectangle and test the functioning of classes using pointer to base class objects and normal objects.**

**PROGRAM :**

**//Program to demonstrate pointer to abstract base class**

#include<iostream>

#include<conio.h>

using namespace std;

class shape *//abstract base class*

{

int a;

float b;

public:

virtual void getdata() = 0; *//pure virtual function definition*

virtual void printarea() *//virtual function definition*

{

cout<<"Lets print the area you want "<<endl;

}

};

class tringle:public shape *//tringle is derived from shape class*

{

float height;

float length;

public:

void getdata()

{

cout<<"Enter height and lenth of tringle : ";

cin>>height>>length;

}

void printarea()

{

cout<<"Area of Tringle is : "<<0.5\*height\*length<<endl;

}

};

class rectangle:public shape *//rectangle is derived from shape class*

{

float width;

float length;

public:

void getdata()

{

cout<<"Enter length and width of rectangle : ";

cin>>length>>width;

}

void printarea()

{

cout<<"Area of Rectangle is : "<<length \* width<<endl;

}

};

int main()

{

shape \*base\_ptr[2]; *//pointer of base class*

tringle t;

rectangle r;

base\_ptr[0] = &t; *//holds address of tringle class object*

base\_ptr[1] = &r; *//holds address of rectangle class object*

cout<<endl<<"Base pointer 1 holds address of tringle class ojject"<<endl;

base\_ptr[0]->getdata();

base\_ptr[0]->printarea();

cout<<endl<<endl<<"Base pointer 2 holds address of rectangle class ojject"<<endl;

base\_ptr[1]->getdata();

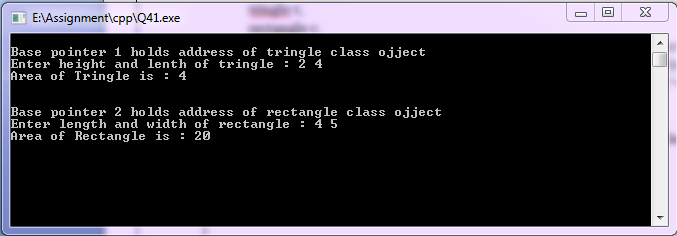
base\_ptr[1]->printarea();

getch();

return 0;

}

**OUTPUT :**

****