Aim:

To write a C++ program to implement classes and objects.

Coding:

#include<iostream>

using namespace std;

class book

{

private:

int bcode;

char title[100];

char author[100];

char publisher[100];

float price;

int no\_pages;

public:

void get()

{

cout<<"Enter the bookcode,title of the book, author,publisher,price and number of pages in the book";

cin>>bcode;

cin>>title;

cin>>author;

cin>>publisher;

cin>>price;

cin>>no\_pages;

}

void put()

{

cout<<"Book code:"<<bcode;

cout<<"Book title:"<<title;

cout<<"Author:"<<author;

cout<<"Publisher:"<<publisher;

cout<<"Price:"<<price;

cout<<"Number of pages:"<<no\_pages;

}

};

int main()

{

book b;

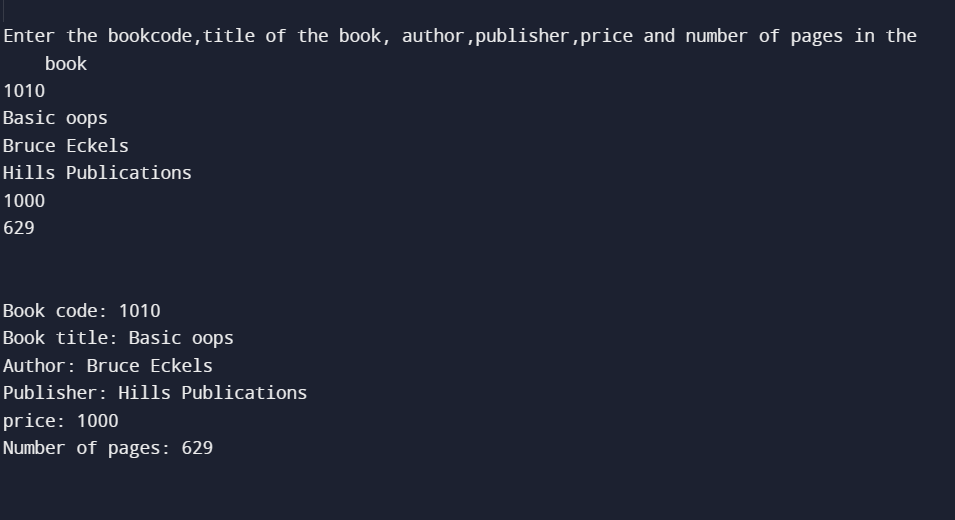
b.get();

b.put();

return 0;

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a C++ program to implement constructor and destructor using array objects .

Coding:

#include <iostream>

using namespace std;

class person

{

private:

int age;

string name;

public:

person()

{

name = "N/A";

age = 0;

}

person(string name,int age = 18)

{

this->name = name;

this->age = age;

}

void display()

{

cout<<name<<"\t"<<age<<endl;

}

~person()

{

cout<<"Destuctor is called"<<endl;

}

};

int main()

{

person p[3]={person("ABC"),person("XYZ",30)

};

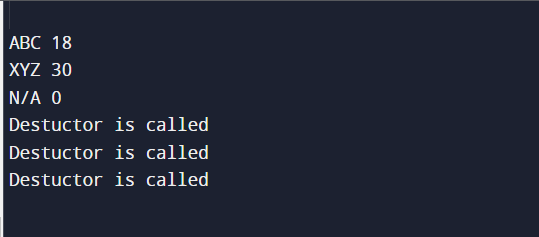
p[0].display();

p[1].display();

p[2].display();

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a C++ program to demonstrate function overloading.

Coding:

#include <iostream>

using namespace std;

void add(int a,int b)

{

cout<<"sum ="<<(a+b)<<endl;

}

void add(double a,double b)

{

cout<<"sum="<<(a+b) )<<endl;

}

void add(int a,double b)

{

cout<<"sum ="<<(a+b) )<<endl;

}

void print(char const \*c)

{

cout<<"Here is char\*"<<c<<endl;

}

int main()

{

add(01,12);

add(1.6,3.2);

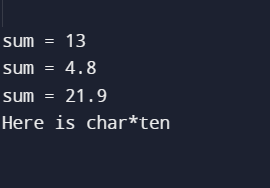
add(15,6.9);

print("ten");

return 0;

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a C++ program to implement string and exception handling.

Coding:

#include <iostream>

using namespace std;

int main()

{

string s = "clow";

try

{

if(s == "clown")

{

throw(s);

cout<<"after throw/n";

}

else

{

cout<<"throw..";

}

}

catch(string s)

{

cout<<"caught"<<s;

}

return 0;

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a program to implement using Multiple inheritance.

Coding:

#include <iostream>

using namespace std;

class student\_mark

{

protected:

int rollno,mark1,mark2;

public:

void get()

{

cout<<"Enter the roll no";

cin>>rollno;

cout<<"enter the two marks";

cin>>mark1>>mark2;

}

};

class co\_mark

{

protected:

int comark;

public:

void getcm()

{

cout<<"Enter the marks of co curricular";

cin>>comark;

}

};

class result:public student\_mark,public co\_mark

{

int avg\_mark,total\_mark;

public:

void display()

{

total\_mark=mark1+mark2+comark;

avg\_mark=total\_mark/3;

cout<<"The total mark is:"<<total\_mark;

cout<<"\n The average mark is:"<<avg\_mark;

}

};

int main()

{

result r;

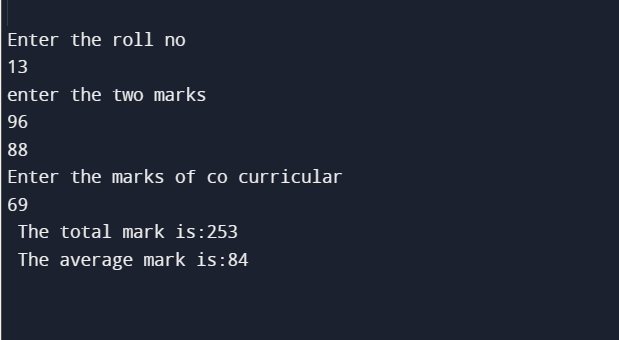
r.get();

r.getcm();

r.display();

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a program to implement Multilevel inheritance.

Coding:

#include<iostream>

using namespace std;

class arith\_op

{

public:

int a = 5;

int b = 30;

};

class add : public arith\_op

{

public:

void addtwonum()

{

cout<<a + b;

}

};

class mul : public add

{

public:

void multwonum()

{

cout<<a\*b;

}

};

int main()

{

mul Numbers;

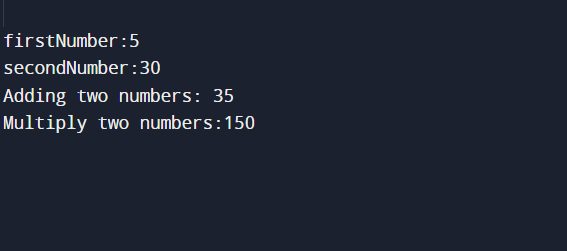
cout<<"firstNumber:"<<Numbers.a<<endl;

cout<<"secondNumber:"<<Numbers.b<<endl;

cout<<"Adding two numbers:";Numbers.addtwonum();

cout<<"Multiply two numbers:";Numbers.multwonum();

}

Output:

Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a program to implement hybrid inheritance.

Coding:

#include <iostream>

using namespace std;

class student

{

public:

int id ;

char name[20];

void getstudent()

{

cout<<"Enter student Id,student name";

cin>>id>>name;

}

};

class marks : public student

{

protected:

int marks\_math,marks\_phy,marks\_chem;

public :

void getmarks()

{

cout<<"Enter 3 subject mark";

cin>>marks\_math>>marks\_phy>>marks\_chem;

}

};

class sports

{

public:

int spmarks;

void getsports()

{

cout<<"Enter sports marks";

cin>>spmarks;

}

};

class result : public marks , public sports

{

public:

int total\_marks;

float avg\_marks;

void display()

{

total\_marks = marks\_math+marks\_phy+marks\_chem;

avg\_marks = total\_marks/3.0;

cout<<"total marks ="<<total\_marks<<endl;

cout<<"Average marks ="<<avg\_marks<<endl;

cout<<"average + sports marks="<<avg\_marks+spmarks;

}

};

int main()

{

result res;

res.getstudent();

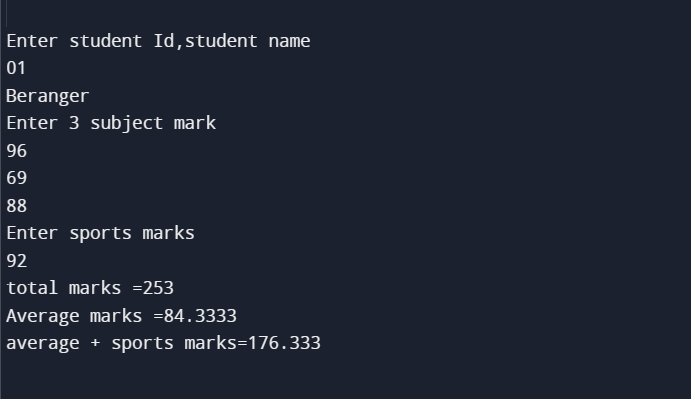
res.getmarks();

res.getsports();

res.display();

return 0;

}

Output:

Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a program implementing virtual functions to demonstrate the use of runtime polymorphism.

Coding:

#include <iostream>

using namespace std;

class Car

{

public:

virtual void s()

{

cout<<"In base"<<endl;

}

};

class Bug : public Car

{

public :

void s()

{

cout<<"In derived"<<endl;

}

} ;

int main()

{

Bug bug;

Car\* c = &bug;

c->s();

return 0;

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.

Aim:

To write a program to implement class and function templates.

Coding:

#include<iostream>

using namespace std;

template <class T>

class calculator

{

private:

T num1,num2;

public:

calculator(T n1,T n2)

{

n1 = num1;

n2 = num2;

}

void displayResult()

{

cout<<"Numbers:"<<num1<<"and"<< num2<<endl;

cout<<num1<<"+"<<num2<<"="<< add()<<endl;

cout<<num1<<"-"<<num2<<"="<< sub()<<endl;

cout<<num1<<"\*"<<num2<<"="<< mul()<<endl;

cout<<num1<<"/"<<num2<<"="<< divd()<<endl;

}

T add()

{

return (num1 + num2);

}

T sub()

{

return num1 - num2;

}

T mul()

{

return num1 \* num2;

}

T divd()

{

return num1 / num2;

}

};

int main()

{

calculator<int>intcalc(2,1);

calculator<float>floatcalc(2.4,1.2);

cout<<"int results :"<<endl;

intcalc.displayResult();

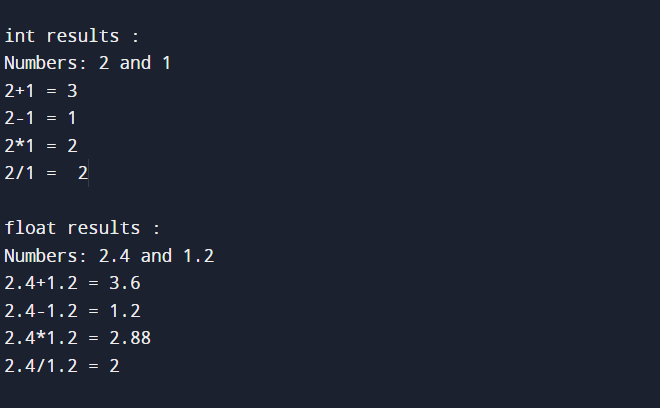
cout<<endl<<"float results"<<endl;

floatcalc.displayResult();

return 0;

}

Output:



Result:

Thus the above program have been successfully executed and the output is verified.