**Question 1:**

**Header file :**

#pragma once

#include<iostream>

#include<string>

using namespace std;

class ship {

string name, year;

public:

ship();

void setname(string name);

string getname();

void setyear(string year);

string getyear();

virtual void print();

};

class cruiseship :public ship

{

double passengers;

public:

cruiseship();

void setpassengers(double passengers);

double getpassengers();

void print();

};

class cargoship :public ship

{

int cargo\_capacity;

public:

void setcargo(int cargo\_capacity);

int getcargo();

void print();

};

**Implementation file**

#include"Header.h"

ship::ship()

{

{

name = "\0";

year = "\0";

}

}

void ship::setname(string name)

{

this->name = name;

}

string ship::getname()

{

return name;

}

void ship::setyear(string year)

{

this->year = year;

}

string ship::getyear()

{

return year;

}

void ship::print()

{

{

cout << "THE NAME OF THE SHIP IS = " << name << endl;

cout << "THE BUILT YEAR OF THIS SHIP IS = " << year << endl;

}

}

cruiseship::cruiseship()

{

passengers = 0;

}

void cruiseship::setpassengers(double passengers)

{

this->passengers = passengers;

}

double cruiseship::getpassengers()

{

return passengers;

}

void cruiseship::print()

{

ship::print();

cout << "THE MAXIMUM NO OF PSSENGERS ARE = " << passengers << endl;

}

void cargoship::setcargo(int cargo\_capacity) {

this->cargo\_capacity = cargo\_capacity;

}

int cargoship::getcargo()

{

return cargo\_capacity;

}

void cargoship::print()

{

cout << "THE NAME OF THE SHIP IS = ";

cout << ship::getname() << endl;

cout << "THE CARGO CAPACITY = " << cargo\_capacity << endl;

}

**Source file**

#include"Header.h"

int main()

{

ship arham;

arham.setname("titanic");

arham.setyear("1800");

cruiseship arham2;

arham2.setname("titantic 2");

arham2.setyear("1801");

arham2.setpassengers(1000);

cargoship arham3;

arham3.setname("titantic 3");

arham3.setyear("1802");

arham3.setcargo(500);

ship\* baseclass;

baseclass = &arham;

cout << "---------------------THE OUTPUT OF CLASS SHIP-----------------------" << endl;

baseclass->print();

baseclass = &arham2;

cout << "---------------------THE OUTPUT OF CLASS CRUSIE-----------------------" << endl;

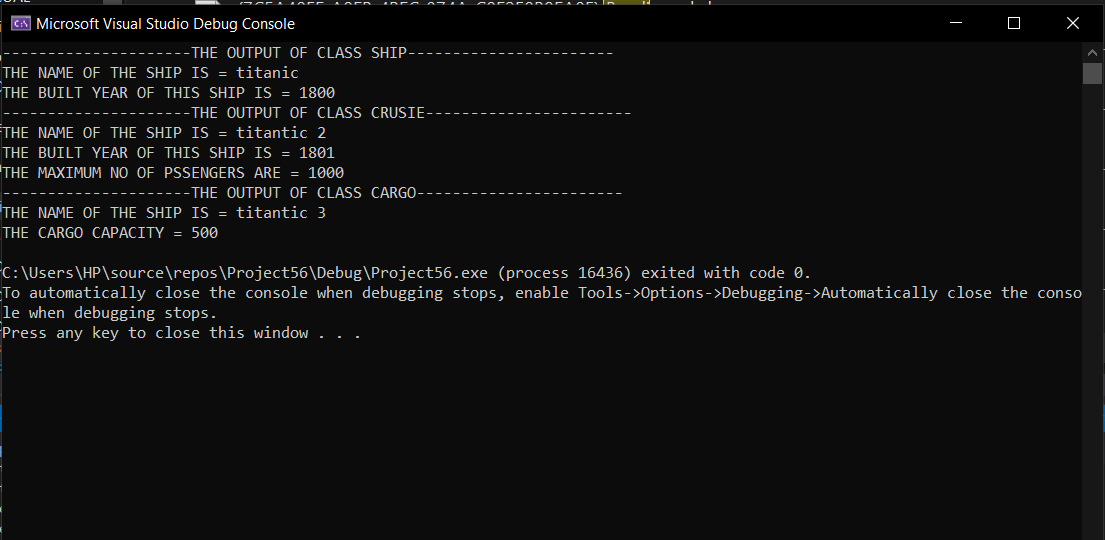
baseclass->print();

baseclass = &arham3;

cout << "---------------------THE OUTPUT OF CLASS CARGO-----------------------" << endl;

baseclass->print();

}



Question 2:

HEADER FILE:

#pragma once

#include<iostream>

using namespace std;

class polygon

{

protected:

float length;float width;

public:

virtual float area() = 0;

virtual float perimeter() = 0;

virtual void display() = 0;

//virtual ~polygon();

};

class sqaure :public polygon{

public :

float area();

float perimeter();

void display();

};

class reactangle:public polygon{

public:

float area();

float perimeter();

void display();

};

class triangle:public polygon{

public:

float area();

float perimeter();

void display();

};

SOURCE FILE:

#include"Header.h"

float polygon::area() {

return 0; // BEACUSE I WANT TO CALCULATE ALL IN DERIVED CLASSES

}

void polygon::display() {

// ITS VOID SO NO NEED TO RETURN

}

float polygon::perimeter() {

return 0;// BEACUSE I WANT TO CALCULATE ALL IN DERIVED CLASSES

}

float sqaure::area()

{

// HERE AREA FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

return 4 \* length;

}

float sqaure::perimeter()

{

// HERE PERIMETER FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

return 4 \* length;

}

void sqaure::display() {

//CALLING THE OVER RIDDEN FUNCTIONS OF AREA AND PERIMETER AND DISPLAYING THEM

cout << "for class sqaure" << endl;

cout << "the area is equal to = " << area() << endl;

cout << "the perimeter is equal to = " << perimeter() << endl;

}

float reactangle::area()

{

// HERE AREA FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

cout << "enter width " << endl;

cin >> width;

return 2 \* (length\*width);

}

float reactangle::perimeter()

{

// HERE PERIMETER FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

cout << "enter width " << endl;

cin >> width;

return 2 \* (length + width);

}

void reactangle::display() {

//CALLING THE OVER RIDDEN FUNCTIONS OF AREA AND PERIMETER AND DISPLAYING THEM

cout << "for class reactangle " << endl;

cout << "the area is equal to = " << area() << endl;

cout << "the perimeter is equal to = " << perimeter() << endl;

}

float triangle::area()

{

// HERE AREA FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

cout << "enter width " << endl;

cin >> width;

return (length \* width)/2;

}

float triangle::perimeter()

{

// HERE PERIMETER FUNCTION IS BEING OVERRIDDEN HENCE PURE VIRTUAL FUNCTION STATISFY

cout << "enter lenght" << endl;

cin >> length;

cout << "enter width " << endl;

cin >> width;

return (length + width);

}

void triangle::display() {

//CALLING THE OVER RIDDEN FUNCTIONS OF AREA AND PERIMETER AND DISPLAYING THEM

cout << "for class triangle " << endl;

cout << "the area is equal to = " << area() << endl;

cout << "the perimeter is equal to = " << perimeter() << endl;

}

MAIN SOURCE FILE:

#include"Header.h"

int main()

{

sqaure arham1; // OBJ FOR SQAURE CLASS

arham1.display();

reactangle arham2; // OBJ FOR REACTANGLE CLASS

arham2.display();

triangle arham3; // OBJ FOR TRIANGLE CLASS

arham3.display();

polygon \*arham; // I VE USED MY NAME INSTEAD OF BPTR xD

arham = &arham1; // FOR DISPLAYING AREA AND PERIMETER OF SQAURE

arham->display();

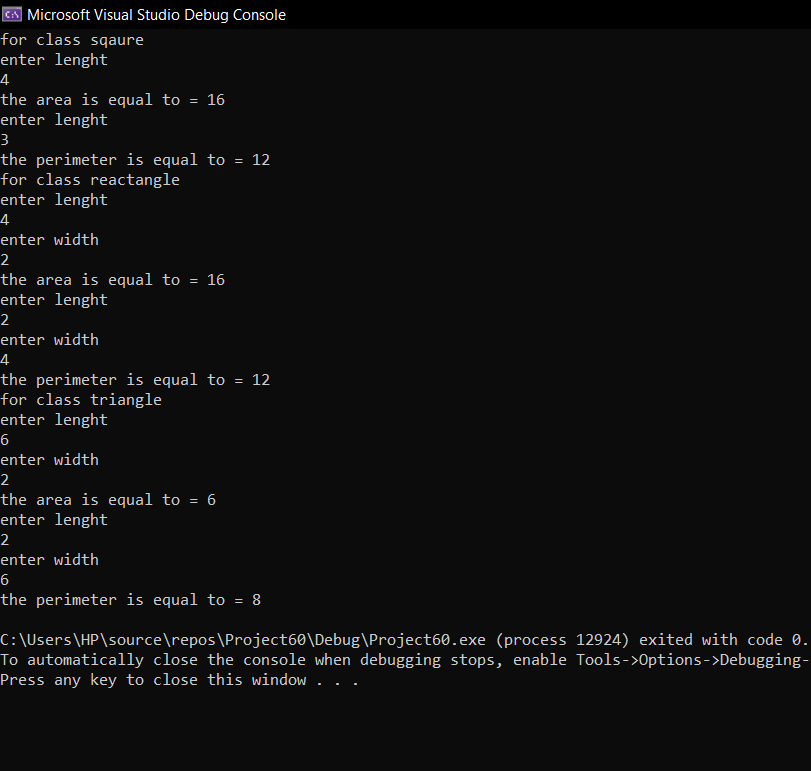
arham = &arham2; // FOR DISPLAYING AREA AND PERIMETER OF TRIANGLE

arham->display();

arham = &arham3; // FOR DISPLAYING AREA AND PERIMETER OF CLASS REACTANGLE

arham->display();

}



Question 3:

Header file:

#pragma once

#include<iostream>

using namespace std;

class bill {

protected:

int units, per\_unit\_cost; // protected data members

float calbill;

public:

// member functions

virtual float monthlybill()=0; // pure virtual function

virtual void displaybill(); // virtual function

};

class billjan :public bill{

public:

float monthlybill(); // for cal bill

void displaybill(); // for displaying bill

~billjan(); // destructor

};

class billfeb :public bill{

public:

float monthlybill(); // for cal bill

void displaybill(); // for displaying bill

~billfeb(); //destructor

};

class billmarch :public bill{

public:

float monthlybill(); // for cal bill

void displaybill();// for displaying bill

~billmarch(); // destructor

};

class billapril:public bill{

public:

float monthlybill(); // for cal bill

void displaybill(); // for displaying bill

~billapril(); // destructor

};

class billmay:public bill{

public:

float monthlybill(); // for cal bill

void displaybill(); // for displaying bill

~billmay(); // destructor

};

Source file:

#include"Header.h"

float bill::monthlybill() {

return 0;

}

void bill::displaybill()

{

}

float billjan::monthlybill()

{// cal units

cout << "ENTER TOTAL UNITS= " << endl;

cin >> units;

cout << "ENTER PER UNIT COST= " << endl;

cin >> per\_unit\_cost;

calbill = units \* per\_unit\_cost;

return calbill;

}

void billjan::displaybill()

{//displayin bill

cout << "total bill is = " << calbill << endl;

}

billjan:: ~billjan()

{

cout << "virtual destructor called for bill jan " << endl;

}

float billfeb::monthlybill()

{

cout << "ENTER TOTAL UNITS= " << endl;

cin >> units;

cout << "ENTER PER UNIT COST= " << endl;

cin >> per\_unit\_cost;

calbill = units \* per\_unit\_cost;

return calbill;

}

void billfeb::displaybill()

{

cout << "total bill is = " << calbill << endl;

}

billfeb:: ~billfeb() {

cout << "virtual destructor called for bill feb " << endl;

}

float billmarch:: monthlybill()

{

cout << "ENTER TOTAL UNITS= " << endl;

cin >> units;

cout << "ENTER PER UNIT COST= " << endl;

cin >> per\_unit\_cost;

calbill = units \* per\_unit\_cost;

return calbill;

}

void billmarch:: displaybill()

{

cout << "total bill is = " << calbill << endl;

}

billmarch:: ~billmarch() {

cout << "virtual destructor called for bill march " << endl;

}

float billmay:: monthlybill()

{

cout << "ENTER TOTAL UNITS= " << endl;

cin >> units;

cout << "ENTER PER UNIT COST= " << endl;

cin >> per\_unit\_cost;

calbill = units \* per\_unit\_cost;

return calbill;

}

void billmay::displaybill()

{

cout << "total bill is = " << calbill << endl;

}

billmay:: ~billmay() {

cout << "virtual destructor called for bill may " << endl;

}

float billapril:: monthlybill()

{

cout << "ENTER TOTAL UNITS= " << endl;

cin >> units;

cout << "ENTER PER UNIT COST= " << endl;

cin >> per\_unit\_cost;

calbill = units \* per\_unit\_cost;

return calbill;

}

void billapril::displaybill()

{

cout << "total bill is = " << calbill << endl;

}

billapril::~billapril()

{

cout << "virtual destructor called for bill april " << endl;

}

Main source file:

#include"Header.h"

int main()

{

bill\* ok; // pointer obj ok for pointing to cal and displaying

cout << "for jan= " << endl;

billjan arham;

ok = &arham;

ok->monthlybill(); // calling calulation func

ok->displaybill(); // display func

cout << "for feb" << endl;

billfeb arham1;

ok = &arham1;

ok->monthlybill(); // calling calulation func

ok->displaybill(); // display func

cout << "for marach = " << endl;

billmarch arham2;

ok = &arham2;

ok->monthlybill(); // calling calulation func

ok->displaybill(); // display func

cout << "for april" << endl;

billapril arham3;

ok = &arham3;

ok->monthlybill(); // calling calulation func

ok->displaybill(); // display func

cout << "for may" << endl;

billmay arham4;

ok = &arham4;

ok->monthlybill(); // calling calulation func

ok->displaybill(); // display func

system("pause");

return 0;

}

