# TASK 1:

#include<iostream>

using namespace std;

class polynomial

{

int deg;

int\* coff;

public:

polynomial()

{

this->deg = 0;

this->coff = NULL;

}

int deggetter()

{

return deg;

}

void degsetter(int deg)

{

this->deg = deg;

}

int\* coffgetter()

{

return coff;

}

int\* coff2(int deg)

{

this->coff = new int[deg];

};

friend istream& operator>>(istream& in, polynomial& Polynomial)

{

cout << endl;

cout << "Enter Degree of the Polynomial:";

in >> Polynomial.deg;

Polynomial.coff = new int[Polynomial.deg];

for (int i = 0; i < Polynomial.deg; i++)

{

cout << "Enter the cofficient Number:" << i + 1 << ":";

cin >> Polynomial.coff[i];

}

return in;

}

friend ostream& operator<<(ostream& out, polynomial& Polynomial)

{

cout << endl << "Your Polynomial is:" << endl;

int n = Polynomial.deg;

for (int i = 0; i < Polynomial.deg; i++)

{

out << Polynomial.coff[i];

out << "x^" << "(" << n << ")";

if (i != Polynomial.deg - 1)

{

cout << "+";

}

n--;

}

return out;

}

polynomial& operator+(polynomial& k)

{

polynomial a;

if ((this->deg) > (k.deg))

{

a.deg = this->deg;

a.coff = new int[this->deg];

int n = (this->deg) - (k.deg);

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

{

a.coff[i] = this->coff[i];

}

for (int i = 0; i < k.deg; i++)

{

a.coff[n] = (this->coff[n]) + (k.coff[i]);

}

}

if ((this->deg) < (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

int n = (k.deg) - (this->deg);

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

{

a.coff[i] = k.coff[i];

}

for (int i = 0; i < this->deg; i++)

{

a.coff[n] = (this->coff[i]) + (k.coff[n]);

n++;

}

}

if ((this->deg) == (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

for (int i = 0; i < k.deg; i++)

{

a.coff[i] = (this->coff[i] + k.coff[i]);

}

}

return a;

}

polynomial& operator-(polynomial& k)

{

polynomial a;

if ((this->deg) > (k.deg))

{

a.deg = this->deg;

a.coff = new int[this->deg];

int n = (this->deg) - (k.deg);

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

{

a.coff[i] = this->coff[i];

}

for (int i = 0; i < k.deg; i++)

{

a.coff[n] = (this->coff[n]) - (k.coff[i]);

}

}

if ((this->deg) < (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

int n = (k.deg) - (this->deg);

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

{

a.coff[i] = k.coff[i];

}

for (int i = 0; i < this->deg; i++)

{

a.coff[n] = (this->coff[i]) - (k.coff[n]);

n++;

}

}

if ((this->deg) == (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

for (int i = 0; i < k.deg; i++)

{

a.coff[i] = (this->coff[i] - k.coff[i]);

}

}

return a;

}

polynomial& operator\*(polynomial& k)

{

polynomial a;

if ((this->deg) > (k.deg))

{

a.deg = this->deg;

a.coff = new int[this->deg];

int n = (this->deg) - (k.deg);

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

{

a.coff[i] = this->coff[i];

}

for (int i = 0; i < k.deg; i++)

{

a.coff[n] = (this->coff[n]) \* (k.coff[i]);

}

}

if ((this->deg) < (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

int n = (k.deg) - (this->deg);

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

{

a.coff[i] = k.coff[i];

}

for (int i = 0; i < this->deg; i++)

{

a.coff[n] = (this->coff[i]) \* (k.coff[n]);

n++;

}

}

if ((this->deg) == (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

for (int i = 0; i < k.deg; i++)

{

a.coff[i] = (this->coff[i] \* k.coff[i]);

}

}

return a;

}

polynomial& operator/(polynomial& k)

{

polynomial a;

if ((this->deg) > (k.deg))

{

a.deg = this->deg;

a.coff = new int[this->deg];

int n = (this->deg) - (k.deg);

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

{

a.coff[i] = this->coff[i];

}

for (int i = 0; i < k.deg; i++)

{

a.coff[n] = (this->coff[n]) / (k.coff[i]);

}

}

if ((this->deg) < (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

int n = (k.deg) - (this->deg);

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

{

a.coff[i] = k.coff[i];

}

for (int i = 0; i < this->deg; i++)

{

a.coff[n] = (this->coff[i]) / (k.coff[n]);

n++;

}

}

if ((this->deg) == (k.deg))

{

a.deg = k.deg;

a.coff = new int[k.deg];

for (int i = 0; i < k.deg; i++)

{

a.coff[i] = (this->coff[i] / k.coff[i]);

}

}

return a;

}

};

int main()

{

polynomial a, b, c;

cin >> a;

cout << a << endl;

cin >> b;

cout << b << endl;

c = (a + b);

cout << "Your Polynomial After Addition is:" << c << endl;

c = (a - b);

cout << "Your Polynomial After Subtracting 1 From Second is:" << c << endl;

c = a \* b;

cout << "Your Polynomail After Mutipiliaction is:" << c << endl;

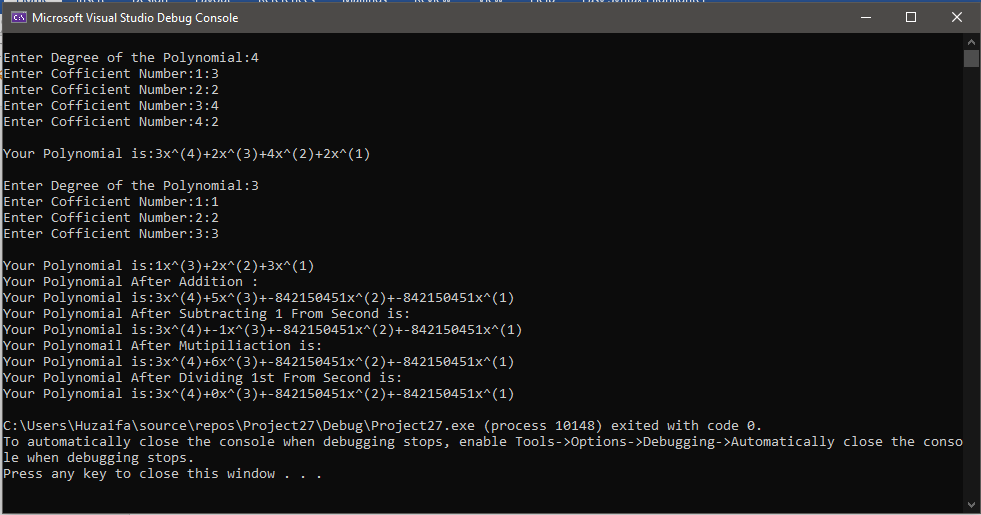
c = a / b;

cout << "Your Polynomial After Dividing 1st From Second is:" << c << endl;

system("pause");

return 0;

}



# TASK 2:

#include<iostream>

using namespace std;

class student {

private:

static int studentNo;

public:

};

class books :public student {

private:

int No\_books;

int day\_b, day\_r, day\_re, time\_b, time\_r;

long long fine\_;

public:

void b\_book() {

cout << "Enter no of book:";

cin >> No\_books;

cout << "Enter date on which you borrow:";

cin >> day\_b;

cout << "Enter day on which you have to return books:";

cin >> day\_r;

cout << "Enter time of borrowing books:";

cin >> time\_b;

}

void r\_book() {

cout << "Enter time of returning books:";

cin >> time\_r;

cout << "Enter day of returning books:";

cin >> day\_re;

}

int fine() {

int d, hour;

d = day\_re - day\_r;

hour = time\_r - time\_b;

hour = (d \* 24) + hour;

fine\_ = (hour \* 5) \* No\_books;

return fine\_;

}

books\* sizeup(books\* ptr, int n) {

books\* ptr1 = new books[n];

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

{

ptr1[i] = ptr[i];

}

delete[] ptr;

ptr = new books[n];

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

{

ptr[i] = ptr1[i];

}

return ptr;

}

};

int student::studentNo = 0;

void menu() {

cout << "Do you want to borrow books." << endl;

cout << "Press 0 if don't." << endl;

cout << "Press 1 if yes." << endl;

}

int main() {

int n, nu = 0;

books\* bk = new books[nu];

menu();

cin >> n;

while (n != 0) {

if (n == 1) {

int i = 0;

bk = bk->sizeup(bk, ++nu);

bk[nu - 1].b\_book();

}

else if (n == 2) {

cout << "Select the player number you want to update : ";

cin >> nu;

bk[nu - 1].b\_book();

}

else if (n == 3) {

int m;

cout << "Enter number of student:";

cin >> m;

bk[m - 1].r\_book();

}

else if (n == 4) {

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

{

cout << "Fine of " << i + 1 << " " << bk[nu - 1].fine() << endl;

}

}

cout << endl;

menu();

cout << "Press 2 if you want to update:" << endl;

cout << "Press 3 if you want to return book:" << endl;

cout << "Press 4 if you want to display fine:" << endl;

cout << "Enter the operation you want to perform:";

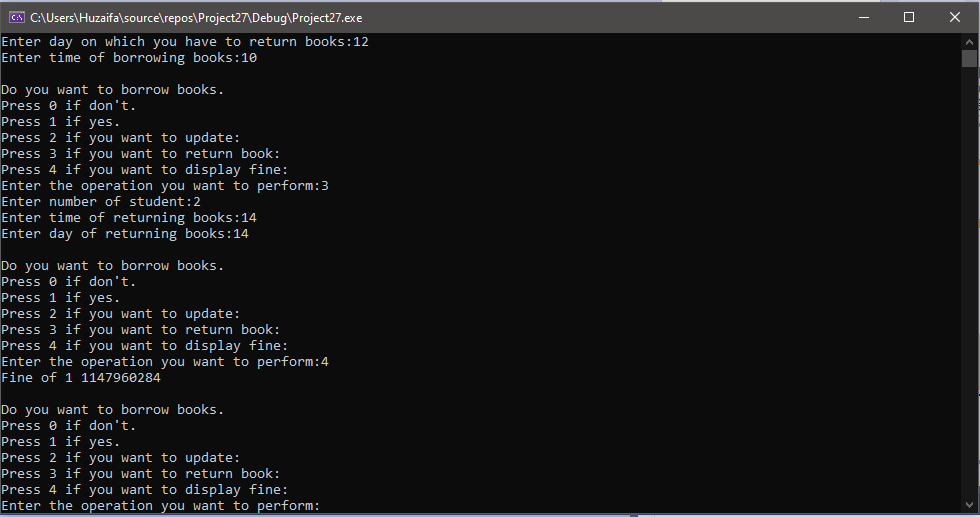
cin >> n;

}

system("pause");

return 0;

}



# TASK 3:

#include<iostream>

using namespace std;

class shop {

private:

long long ID, salary;

string name, post, status;

public:

shop() {

ID = 9166;

salary = 25000;

name = "abdul";

post = "Owner";

status = "working";

}

void input() {

}

void operator -() {

cout << "Enter working status:";

cin >> status;

}

void operator\*() {

salary = salary + (10 / 100) \* salary;

}

friend istream& operator>>(istream& in, shop& obj) {

cout << "Enter ID:";

in >> obj.ID;

cout << "Enter name:";

in >> obj.name;

cout << "Enter Post:";

in >> obj.post;

cout << "Enter status:";

in >> obj.status;

cout << "Enter salary:";

in >> obj.salary;

return in;

}

friend ostream& operator<<(ostream& out, shop& obj) {

cout << "ID:";

out << obj.ID;

cout << "\nName:";

out << obj.name;

cout << "\nPost:";

out << obj.post;

cout << "\nStatus:";

out << obj.status;

cout << "\nSalary:";

out << obj.salary;

return out;

}

void operator++() {

salary += 5000;

}

void operator--() {

salary -= 5000;

}

};

int main() {

shop ob[5];

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

-ob[i];

\*ob[i];

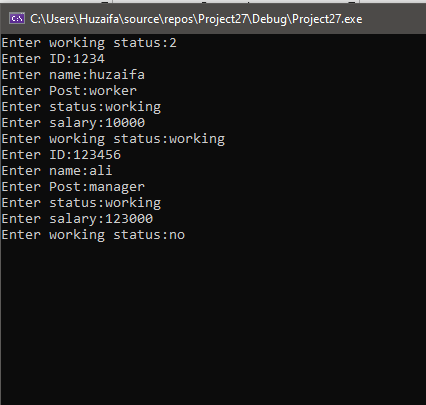
cin >> ob[i];

++ob[i];

--ob[i];

}

}



# TASK 4:

#include<iostream>

using namespace std;

class base {

private:

int base3;

public:

int base1, base2;

base() {

base1 = 0;

base2 = 0;

base3 = 0;

}

base(int a, int b, int c) {

base1 = a;

base2 = b;

base3 = c;

}

void display() {

cout << "base1=" << base1 << endl;

cout << "base2=" << base2 << endl;

cout << "base3=" << base3 << endl;

}

friend class derived;

friend class Friend\_class;

};

class derived :public base {

private:

int derived2;

public:

int derived1;

derived() {

base1 = 1;

base2 = 1;

base3 = 1;

derived1 = 1;

derived2 = 1;

}

derived(int a, int b, int c, int d, int e) :base(a, b, c) {

base1 = a;

base2 = b;

base3 = c;

derived1 = d;

derived2 = e;

}

void display() {

cout << "base1=" << base1 << endl;

cout << "base2=" << base2 << endl;

cout << "base3=" << base3 << endl;

cout << "derived1=" << derived1 << endl;

cout << "derived2=" << derived2 << endl;

}

friend class Friend\_class1;

};

class Friend\_class {

private:

base b;

int fr1, fr2;

public:

void access\_method() {

b.base1 = 2;

b.base2 = 3;

b.base3 = 1;

b.display();

}

friend class Friend\_class2;

};

class Friend\_class1 {

private:

derived dr;

public:

void access\_method() {

dr.derived1 = 2;

dr.derived2 = 3;

dr.display();

}

};

class Friend\_class2 {

private:

Friend\_class fr;

public:

void access\_method() {

fr.fr1 = 2;

fr.fr2 = 1;

fr.b.base1 = 3;

fr.b.base2 = 4;

fr.b.display();

}

};

int main() {

Friend\_class2 f;

f.access\_method();

int a, b, c, d, e;

cout << "Enter base1:";

cin >> a;

cout << "Enter base2:";

cin >> b;

cout << "Enter base3:";

cin >> c;

cout << "Enter derived1:";

cin >> d;

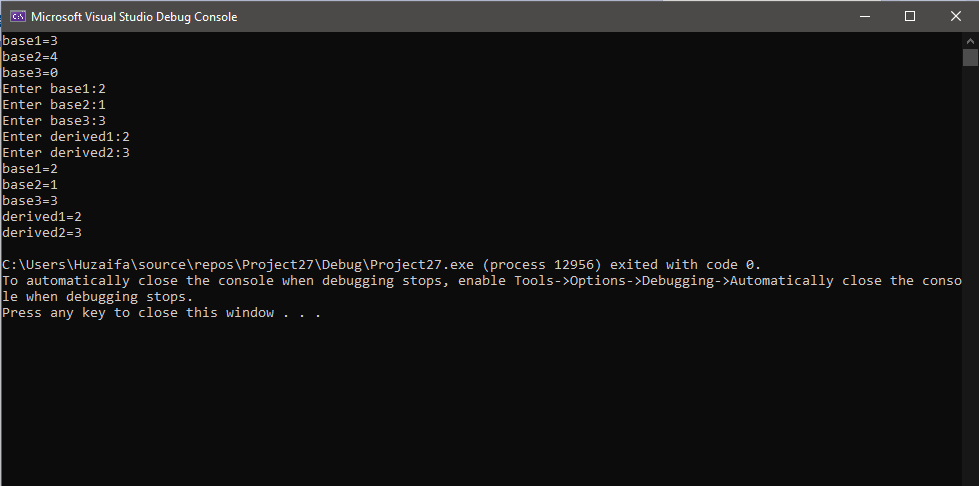
cout << "Enter derived2:";

cin >> e;

derived dr(a, b, c, d, e);

dr.display();

}



# TASK 5:

#include<iostream>

using namespace std;

class Friend {

private:

int a=2, b=3, c=4;

friend void fun(Friend&);

friend void fun1(Friend&);

};

void fun(Friend &a1)

{

a1.a = 1;

a1.b = 1;

a1.c = 1;

cout << "a = " << a1.a << endl;

cout << "b = " << a1.b << endl;

cout << "c = " << a1.c<< endl;

}

void fun1(Friend& a1)

{

cout << a1.a;

cout << endl;

cout << a1.b;

cout << endl;

cout << a1.c;

}

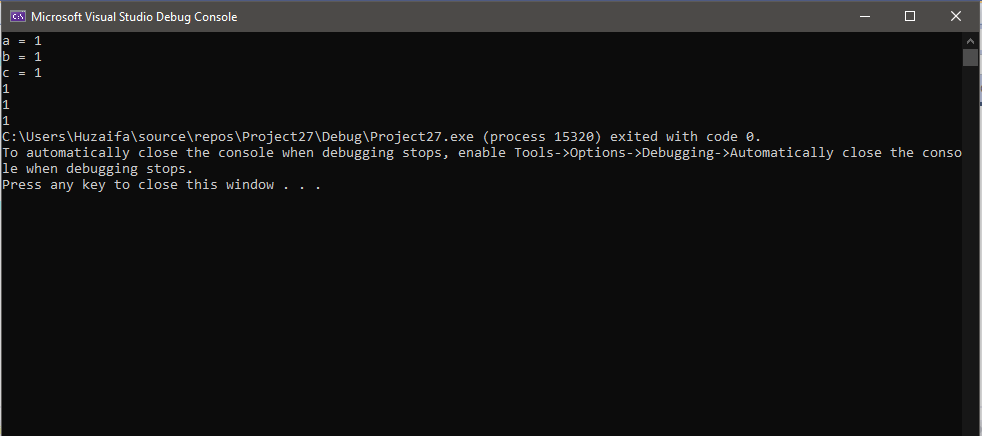
int main() {

Friend f;

fun(f);

fun1(f);

}



# TASK 6:

#include<iostream>

#include<string>

using namespace std;

class publication

{

private:

string name;

float price;

public:

virtual void GetData() {

cout << "Enter Name of the Publication :";

cin >> name;

cout << "Enter price of Publication :";

cin >> price;

}

virtual void OutData()

{

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

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

}

};

class book :public publication

{

int NoOfPages;

public:

void GetData() {

cout << "Enter number of pages of Book:";

cin >> NoOfPages;

}

void OutData() {

cout << "Number of pages in Book is:" << NoOfPages << endl;

}

};

class tape :public publication {

float time;

public:

void GetData() {

cout << "Enter playing time in minutes : ";

cin >> time;

}

void OutData() {

cout << "Playing time in minutes :" << time << endl;

}

};

int main()

{

int step;

publication\* a[100];

for (int i = 0; i < 100; i++) {

cin >> step;

if (step == 1) {

a[i] = new publication;

a[i]->publication::GetData();

a[i]->publication::OutData();

a[i] = new book;

a[i]->GetData();

a[i]->OutData();

a[i] = new tape;

a[i]->GetData();

a[i]->OutData();

cout << "Press 1 to continue or any other to end program" << endl;

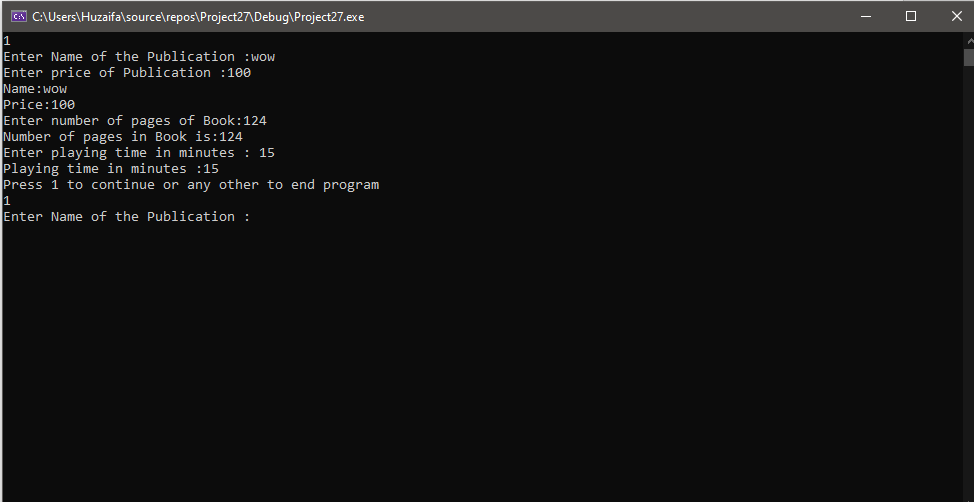
}

else

break;

}

}



# TASK 8:

#include<iostream>

using namespace std;

class Polygon {

protected:

float length, width, height, ar, peri;

public:

Polygon() {

length = 2;

width = 3;

height = 1;

ar = 3;

peri = 2;

}

virtual float area() = 0;

virtual float perimeter() = 0;

virtual void display() = 0 {

}

};

class square :public Polygon {

public:

square() {

length = 5;

}

square(float a) {

length = a;

}

float area() {

ar = length \* length;

return ar;

}

float perimeter() {

peri = 4 \* length;

return peri;

}

void display() {

cout << "area:" << ar << " " << endl;

cout << "perimeter:" << peri << " " << endl;

}

};

class triangle :public Polygon {

public:

triangle() {

length = 3;

width = 4;

}

triangle(float a, float b) {

length = a;

width = b;

}

float area() {

ar = 2 \* (length \* width);

return ar;

}

float perimeter() {

peri = 2 \* (length + width);

return peri;

}

void display() {

cout << "area:" << ar << " " << endl;

cout << "perimeter:" << peri << " " << endl;

}

};

class rectangle : public Polygon {

public:

rectangle() {

length = 5;

width = 1;

height = 2;

}

rectangle(float a, float b, float c) {

length = a;

width = b;

height = c;

}

float area() {

ar = (length \* width) / 2;

return ar;

}

float perimeter() {

peri = length + width + height;

return peri;

}

void display() {

cout << "area:" << ar << " " << endl;

cout << "perimeter:" << peri << " " << endl;

}

};

int main() {

float a, b, c;

cout << "Enter length:";

cin >> a;

cout << "Enter width:";

cin >> b;

cout << "Enter height:";

cin >> c;

cout << "\nData of square:\n";

Polygon\* p = new square(a);

p->area();

p->perimeter();

p->display();

cout << "\nData of triangle:\n";

delete p;

p = new triangle(a, b);

p->area();

p->perimeter();

p->display();

delete p;

cout << "\nData of rectangle:\n";

p = new rectangle(a, b, c);

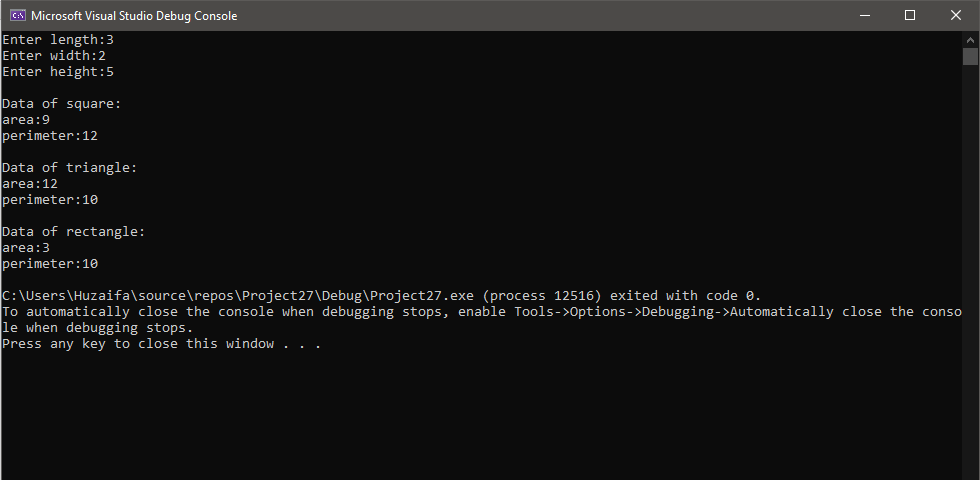
p->area();

p->perimeter();

p->display();

delete p;

}



# TASK 9:

#include<iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

virtual void set\_values(int a, int b)

{

width = a; height = b;

}

virtual int area() {

return 0;

}

};

class Rectangle : public Polygon {

public:

int area()

{

return width\*height;

}

};

class Triangle : public Polygon {

public:

int area()

{

return width\*height / 2;

}

};

int main() {

Rectangle rect;

Triangle trgl;

Polygon \* ppoly1 = &rect;

Polygon \* ppoly2 = &trgl;

ppoly1->set\_values(4, 5);

ppoly2->set\_values(4, 5);

cout<<ppoly2->area()<<endl;

system("pause");

return 0;

}

**­ Reason:**

* In class polygon ‘\_’ was missing between set\_value();
* area() was not declared in the parent class. We should write the definition of the area in the base class and make it virtual, so we can access it from ppoly2 pointer.