Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

з комп’ютерного практикуму № 2

на тему :

„ НАСЛЕДОВАНИЕ С++ ”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *ІП-63 Дроздович В.О.* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *ас. Головченко М.М.* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

ЗМІСТ

[1 Мета роботи 3](#_Toc501592188)

[2 Постановка задачі 4](#_Toc501592189)

[3 Аналітичні викладки 5](#_Toc501592190)

[4 UML діаграмма класу 6](#_Toc501592191)

[5 Тексти програмного коду 7](#_Toc501592192)

[6 Результат роботи програми 9](#_Toc501592193)

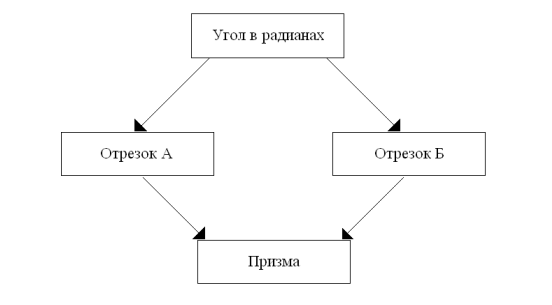
[Висновки 10](#_Toc501592194)

# Мета роботи

*Мета роботи* – придбати навички складання елементарних програм із використанням поліморфізму.

# Постановка задачі

Определить следующую иерархию классов (рисунок 3.9).



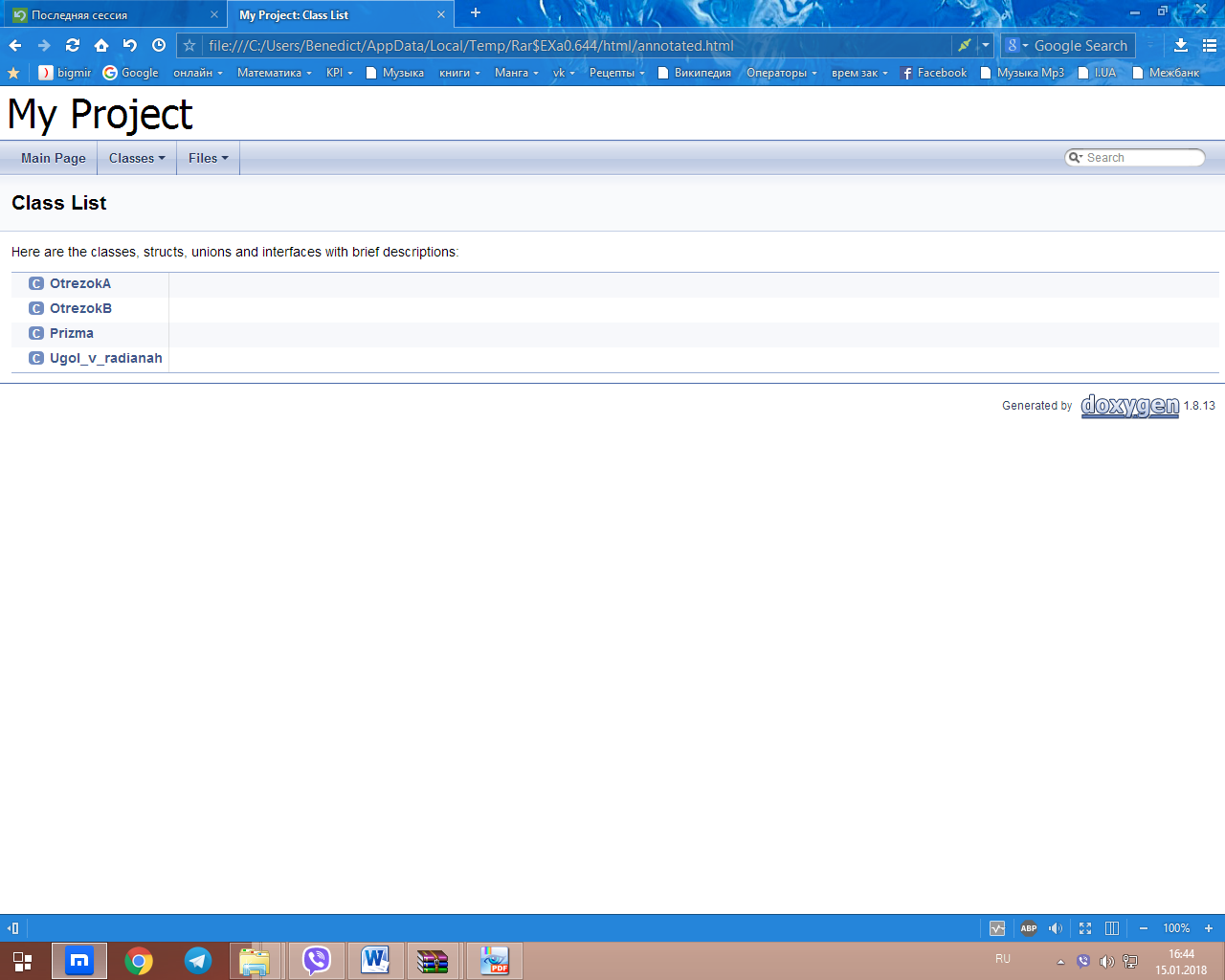
Определить площадь боковой поверхности и объём призмы.

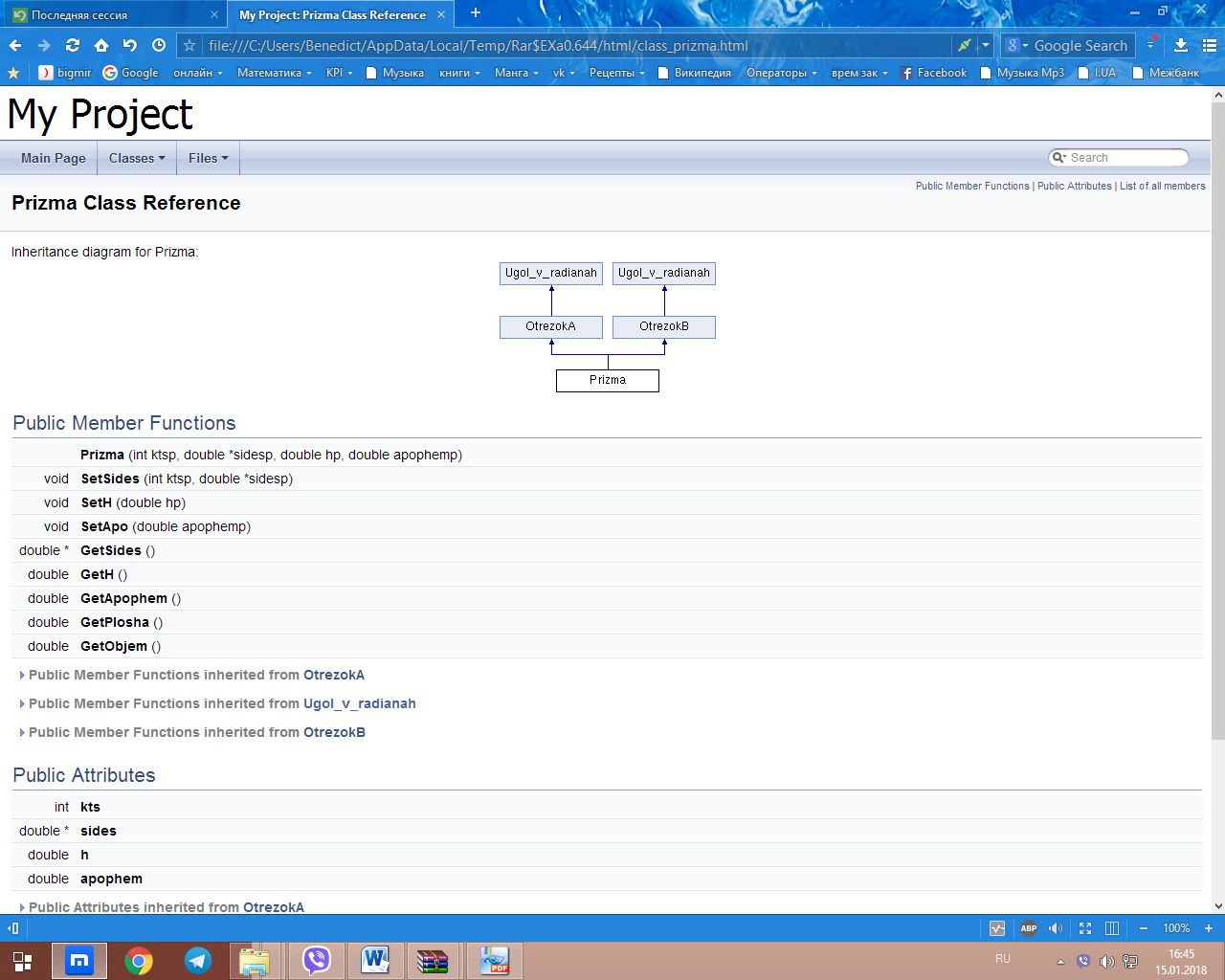
Инициализация данных производится пользователем.

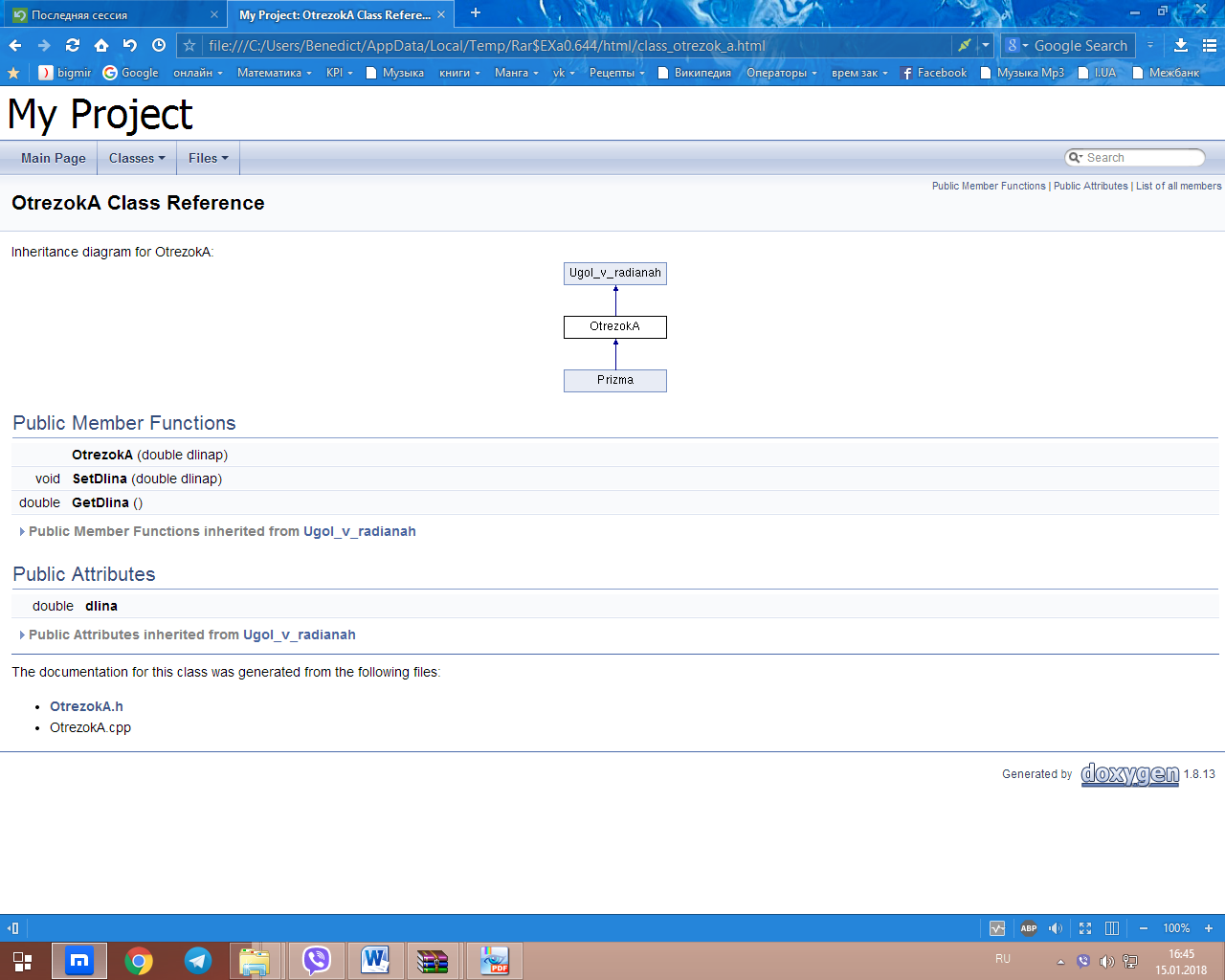
# Аналітичні викладки

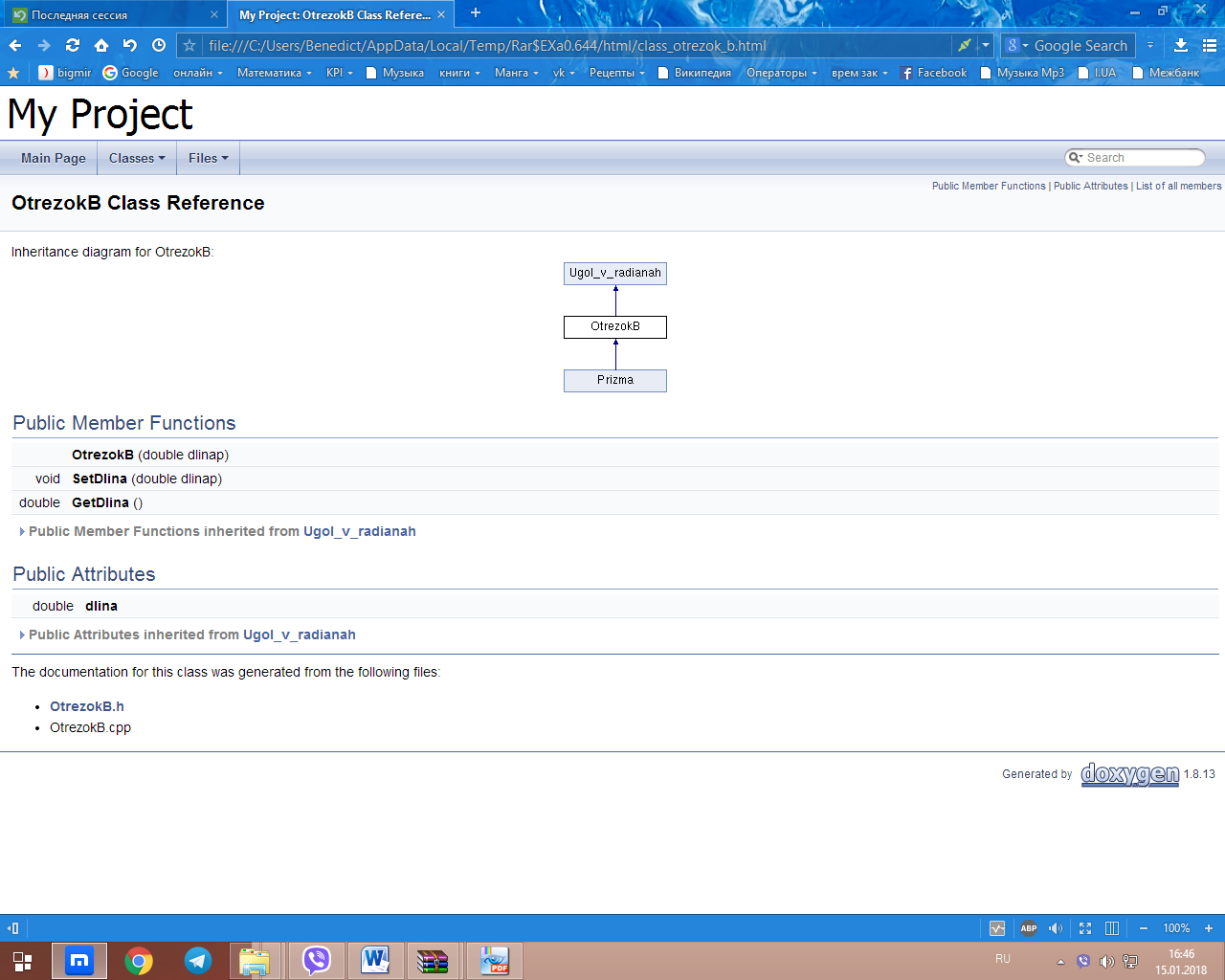
Наследование – это способность брать существующий (базовый) класс и порождать из него новый класс – потомок, с наследованием всех его атрибутов и поведения. Синтаксис определения производного класса: class имя\_класса: спецификатор\_доступа1 базовыйо\_класс1, спецификатор\_доступа2 базовый\_класс2,{ //тело класса }; При описании нового класса, производного от какого-то одного или нескольких базовых классов, можно добавлять новые функции-элементы и элементы-данные, сохраняя при этом все элементы родителей, а можно родительские элементы переопределить или перегрузить. В производном классе доступны открытые и защищённые элементы базового класса (прямого или косвенного). Закрытые элементы базового класса в производном классе недоступны. Производный класс может наследоваться от базового класса как public, protected или private. Защищённое и закрытое наследование встречаются редко и каждое из них нужно использовать с большой осторожностью.

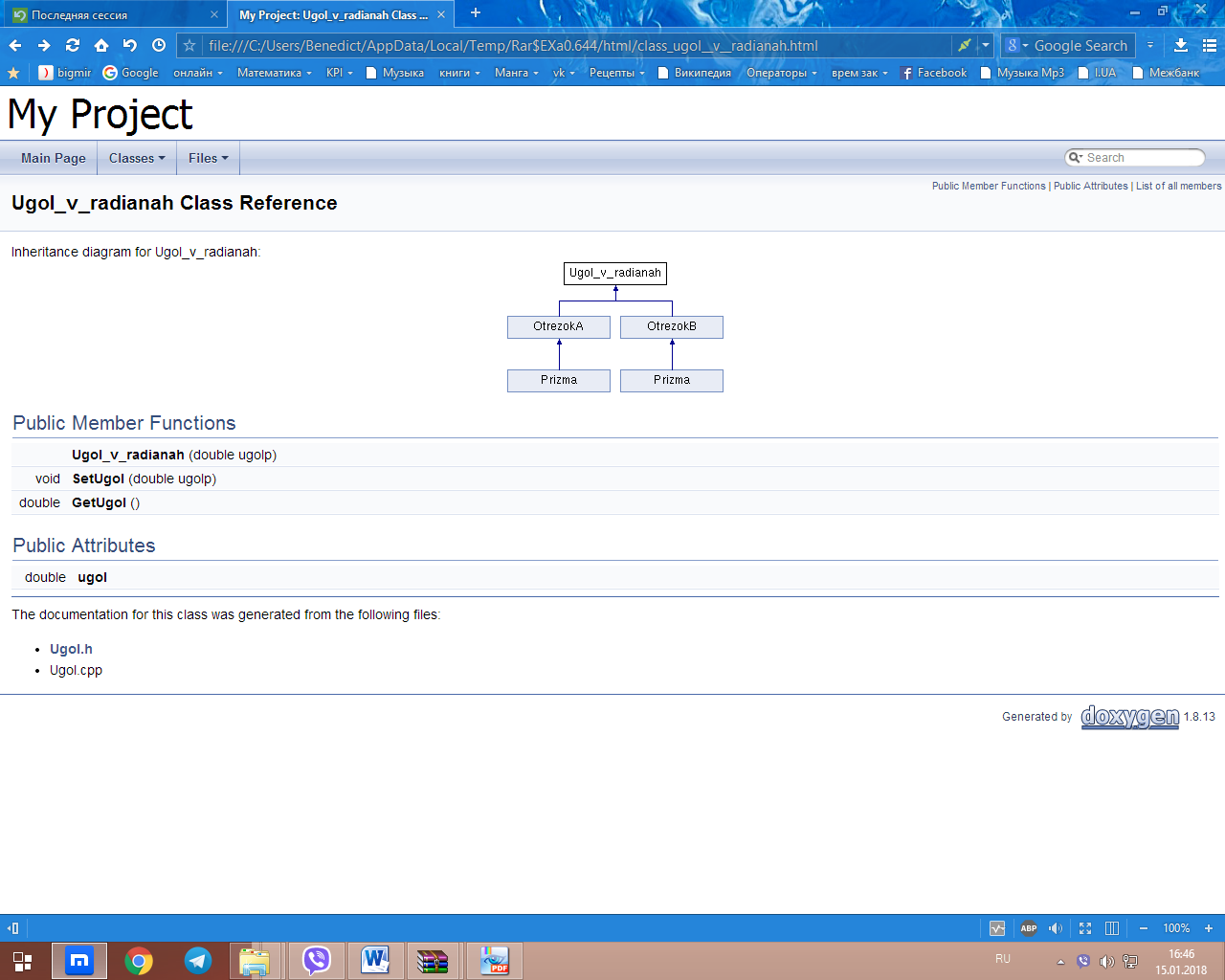
# UML діаграмма класу











# Тексти програмного коду

#include"OtrezokB.h"

class Prizma: public OtrezokA, public OtrezokB

{

public:

int kts;

double \*sides;

double h;

double apophem;

Prizma();

Prizma(int ktsp, double \*sidesp, double hp, double apophemp);

void SetSides(int ktsp, double \*sidesp);

void SetH(double hp);

void SetApo(double apophemp);

double\* GetSides();

double GetH();

double GetApophem();

double GetPlosha();

double GetObjem();

~Prizma();

};

#include"Prizma.h"

Prizma::Prizma()

{

kts=4;

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

{

sides[i]=10;

}

apophem=5;

h=100;

}

Prizma::Prizma(int ktsp, double \*sidesp, double hp,double apophemp)

{

kts=ktsp;

sides = new double[ktsp];

sides=sidesp;

h=hp;

apophem = apophemp;

}

void Prizma::SetSides(int ktsp, double \*sidesp)

{

sides = new double[ktsp];

kts=ktsp;

sides=sidesp;

}

void Prizma::SetApo(double apophemp)

{

apophem=apophemp;

}

void Prizma::SetH(double hp)

{

h=hp;

}

double\* Prizma::GetSides()

{

return sides;

}

double Prizma::GetApophem()

{

return apophem;

}

double Prizma::GetH()

{

return h;

}

double Prizma::GetPlosha()

{

double perimetr=0;

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

{

perimetr+=sides[i];

}

return perimetr\*h;

}

double Prizma::GetObjem()

{

return GetPlosha()\*apophem/2;

}

Prizma::~Prizma()

{

delete []sides;

apophem=NULL;

h=NULL;

}

#include"Ugol.h"

class OtrezokA : public Ugol\_v\_radianah

{

public:

double dlina;

OtrezokA();

OtrezokA(double dlinap);

void SetDlina(double dlinap);

double GetDlina();

~OtrezokA();

};

#include"OtrezokA.h"

OtrezokA::OtrezokA()

{

dlina=100;

}

OtrezokA::OtrezokA(double dlinap)

{

dlina=dlinap;

}

void OtrezokA::SetDlina(double dlinap)

{

dlina=dlinap;

}

double OtrezokA::GetDlina()

{

return dlina;

}

OtrezokA::~OtrezokA()

{

dlina=NULL;

}

#include"OtrezokA.h"

class OtrezokB : public Ugol\_v\_radianah

{

public:

double dlina;

OtrezokB();

OtrezokB(double dlinap);

void SetDlina(double dlinap);

double GetDlina();

~OtrezokB();

};

#include"OtrezokB.h"

OtrezokB::OtrezokB()

{

dlina=100;

}

OtrezokB::OtrezokB(double dlinap)

{

dlina=dlinap;

}

void OtrezokB::SetDlina(double dlinap)

{

dlina=dlinap;

}

double OtrezokB::GetDlina()

{

return dlina;

}

OtrezokB::~OtrezokB()

{

dlina=NULL;

}

#include<iostream>

class Ugol\_v\_radianah

{

public:

double ugol;

Ugol\_v\_radianah();

Ugol\_v\_radianah(double ugolp);

void SetUgol(double ugolp);

double GetUgol();

~Ugol\_v\_radianah();

};

#include"Ugol.h"

Ugol\_v\_radianah::Ugol\_v\_radianah()

{

ugol=90;

}

Ugol\_v\_radianah::Ugol\_v\_radianah(double ugolp)

{

ugol=ugolp;

}

void Ugol\_v\_radianah::SetUgol(double ugolp)

{

ugol=ugolp;

}

double Ugol\_v\_radianah::GetUgol()

{

return ugol;

}

Ugol\_v\_radianah::~Ugol\_v\_radianah()

{

ugol=NULL;

}

#include"Prizma.h"

using namespace std;

int main()

{ setlocale(LC\_ALL, "Russian");

int ktsp;

cout<<"Введите количество сторон : \n";

cin>>ktsp;

cout<<"Введите длину каждой из сторон призмы: \n";

double \*sidesp = new double[ktsp];

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

{

cin>>sidesp[i];

}

cout<<"Введите высоту призмы : \n";

double hp;

cin>>hp;

cout<<"Введите длину апофемы : \n";

double apo;

cin>>apo;

Prizma \*prizma = new Prizma(ktsp,sidesp,hp,apo);

cout<<"Площадь призмы = "<<prizma->GetPlosha()<<endl;

cout<<"Объем призмы = "<<prizma->GetObjem()<<endl;

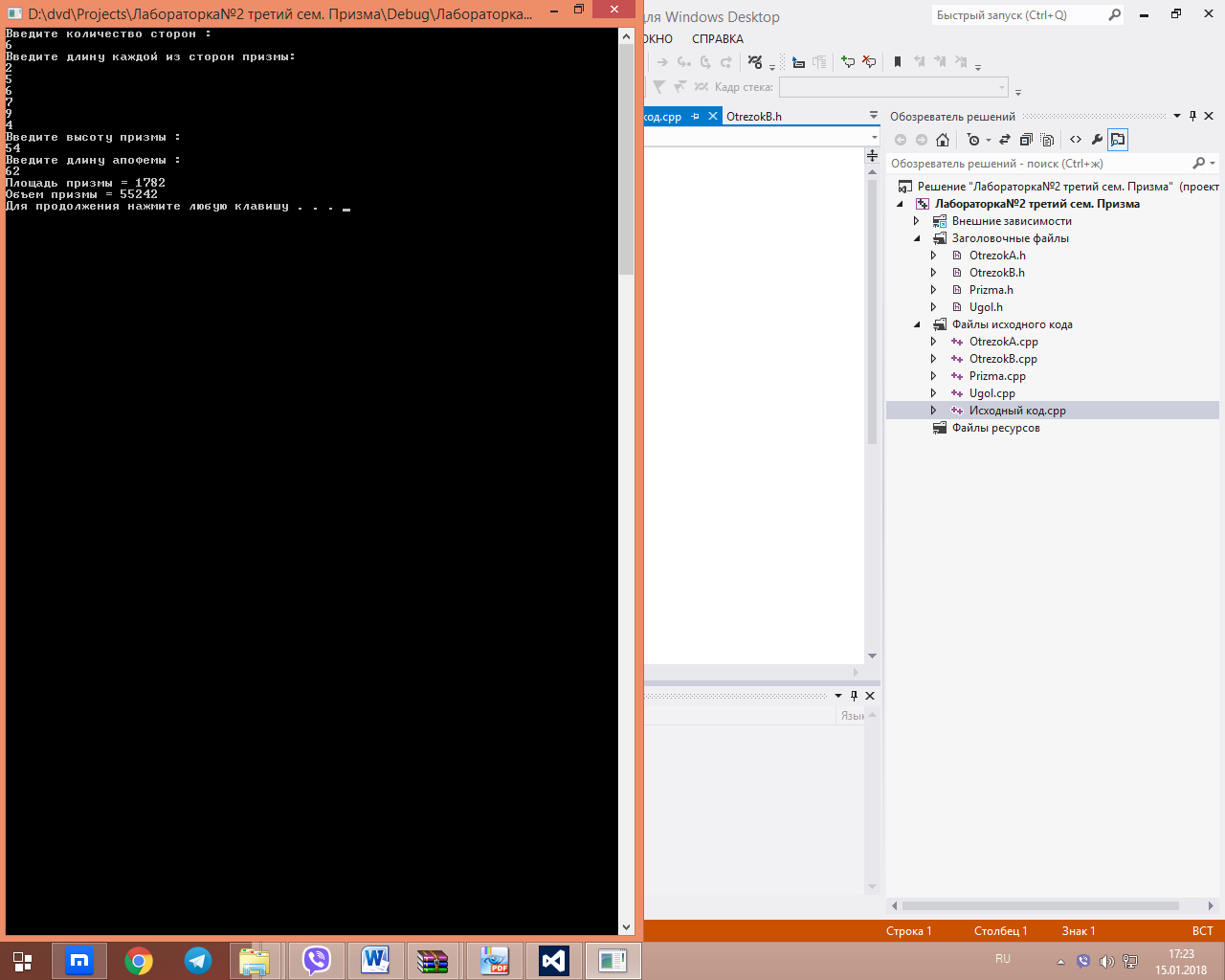
delete []sidesp;

system("pause");

return 0;

}

# Результат роботи програми



Висновки

В ході виконання лабораторної роботи я отримав навички скаладання та застосування поліморфізму в С++.