**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 7**

Тема: Проектирование структуры классов

Студент: Савров Никита

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

Спроектировать простейший графический векторный редактор.

Требование к функционалу редактора:

• создание нового документа

• импорт документа из файла

• экспорт документа в файл

• создание графического примитива (согласно варианту задания)

• удаление графического примитива

• отображение документа на экране (печать перечня графических объектов и их характеристик)

• реализовать операцию undo, отменяющую последнее сделанное действие. Должно действовать для операций добавления/удаления фигур.

Требования к реализации:

• Создание графических примитивов необходимо вынести в отдельный класс – Factory.

• Сделать упор на использовании полиморфизма при работе с фигурами;

• Взаимодействие с пользователем (ввод команд) реализовать в функции main;

1. Описание программы

Для реализации задачи реализуем класс figure от которого наследуем классы rectangle rhomb и trapeze. Для обработки фигур реализуем класс factory. Он будет осуществлять ввод фигур и обработку команд, а также с помощью структуры данных стек осуществлять операцию undo, путем запоминания всех команд в стек и отмены последней.

1. Набор testcases

test\_01

1

doc file

2

0 0

0 10

10 10

20 0

3

0 0

20 20

4

20 20

5

7

6

test\_02

1

doc file1

3

0 0

10 10

4

10 10

4

15 15

4

20 20

4

30 30

6

5

5

6

test\_03

1

doc file2

3

0 0

20 20

3

0 0

10 10

3

10 10

30 30

3

5 10

20 20

7

7

6

1. Результаты выполнения тестов.

test\_01

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1

doc file

2

0 0

0 10

10 10

20 0

3

0 0

20 20

4

20 20

5

7

6

input name doc and file

Succesfull

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinats for 4 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input angle and side

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

trapeze

0 0

0 10

10 10

20 0

rectangle

0 0

0 20

20 20

20 0

rhomb

0 0

11.4715 16.383

27.8546 27.8546

16.383 11.4715

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

test\_02

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1

doc file1

3

0 0

10 10

4

10 10

4

15 15

4

20 20

4

30 30

6

5

5

6

input name doc and file

Succesfull

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input angle and side

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input angle and side

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input angle and side

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input angle and side

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

rectangle

0 0

0 10

10 10

10 0

rhomb

0 0

6.42788 7.66044

14.0883 14.0883

7.66044 6.42788

rhomb

0 0

9.13142 11.9003

21.0317 21.0317

11.9003 9.13142

rhomb

0 0

11.4715 16.383

27.8546 27.8546

16.383 11.4715

rhomb

0 0

15 25.9808

40.9808 40.9808

25.9808 15

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

rectangle

0 0

0 10

10 10

10 0

rhomb

0 0

6.42788 7.66044

14.0883 14.0883

7.66044 6.42788

rhomb

0 0

9.13142 11.9003

21.0317 21.0317

11.9003 9.13142

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

test\_03

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1

doc file2

3

0 0

20 20

3

0 0

10 10

3

10 10

30 30

3

5 10

20 20

7

7

6

input name doc and file

Succesfull

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

input 2 coordinates for 2 vertex

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

rectangle

0 0

0 20

20 20

20 0

rectangle

0 0

0 10

10 10

10 0

1 - create new document

2 - add trapeze

3 - add rectangle

4 - add rhomb

5 - remove last figure

6 - print document

7 - undo

8 - save

9 - load

0 - show

5.Листинг программы

rectangle.h

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include <algorithm>

#include <utility>

#include <iostream>

#include <vector>

#include <cmath>

#include "figure.h"

class Rectangle : public Figure{

public:

std::pair <double, double> a,b,c,d;

int number\_vertex = 4;

std::string name;

Rectangle(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (std::vector <std::pair<double,double>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

name = "rectangle";

}

std::string GetName(){

return name;

}

std::vector <std::pair <double, double>> GetCoord(){

std::vector <std::pair <double, double>> v(4);

v[0].first = a.first;

v[0].second = a.second;

v[1].first = b.first;

v[1].second = b.second;

v[2].first = c.first;

v[2].second = c.second;

v[3].first = d.first;

v[3].second = d.second;

return v;

}

friend std::istream& operator>>(std::istream& in, Rectangle &obj){

std::vector <std::pair <double,double>> v(4);

in >> v[0].first >> v[0].second >> v[2].first >> v[2].second;

v[1].first = v[0].first;

v[1].second = v[2].second;

v[3].first = v[2].first;

v[3].second = v[0].second;

obj.Set(v);

return in;

}

friend std::ostream& operator<<(std::ostream& out, Rectangle &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

trapeze.h

#ifndef TRAPEZE\_H

#define TRAPEZE\_H

#include <algorithm>

#include <utility>

#include <iostream>

#include <vector>

#include <cmath>

#include "figure.h"

class Trapeze : public Figure{

public:

std::pair <double, double> a,b,c,d;

int number\_vertex = 4;

std::string name;

Trapeze(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (std::vector <std::pair<double,double>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

name = "trapeze";

}

std::string GetName(){

return name;

}

std::vector <std::pair <double, double>> GetCoord(){

std::vector <std::pair <double, double>> v(4);

v[0].first = a.first;

v[0].second = a.second;

v[1].first = b.first;

v[1].second = b.second;

v[2].first = c.first;

v[2].second = c.second;

v[3].first = d.first;

v[3].second = d.second;

return v;

}

friend std::istream& operator>>(std::istream& in, Trapeze &obj){

std::vector <std::pair <double,double>> v(4);

for (int i = 0; i < 4; i++){

in >> v[i].first >> v[i].second;

}

obj.Set(v);

return in;

}

friend std::ostream& operator<<(std::ostream& out, Trapeze &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

#endif

rhomb.h

#ifndef RHOMB\_H

#define RHOMB\_H

#include <algorithm>

#include <utility>

#include <iostream>

#include <vector>

#include <cmath>

#include "figure.h"

class Rhomb : public Figure{

public:

std::pair <double, double> a,b,c,d;

int number\_vertex = 4;

std::string name;

Rhomb(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (std::vector <std::pair<double,double>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

name = "rhomb";

}

std::string GetName(){

return name;

}

std::vector <std::pair <double, double>> GetCoord(){

std::vector <std::pair <double, double>> v(4);

v[0].first = a.first;

v[0].second = a.second;

v[1].first = b.first;

v[1].second = b.second;

v[2].first = c.first;

v[2].second = c.second;

v[3].first = d.first;

v[3].second = d.second;

return v;

}

friend std::istream& operator>>(std::istream& in, Rhomb &obj){

std::vector <std::pair <double,double>> v(4);

int angle;

double side, pi = M\_PI;

std::cin >> angle >> side;

v[0].first = 0;

v[0].second = 0;

if (angle <= 90){

double alpha = (90 - angle) / 2;

v[3].second = sin(alpha \* pi / 180) \* side;

v[3].first = cos(alpha \* pi / 180) \* side;

v[1].first = sin(alpha \* pi / 180) \* side;

v[1].second = cos(alpha \* pi / 180) \* side;

v[2].second = sin(alpha \* pi / 180) \* side + v[1].second;

v[2].first = cos(alpha \* pi / 180) \* side + v[1].first;

} else if (angle < 180) {

double alpha = (270 - angle) / 2;

v[3].second = cos(alpha \* pi / 180) \* side;

v[3].first = sin(alpha \* pi / 180) \* side;

v[1].first = cos(alpha \* pi / 180) \* side;

v[1].second = sin(alpha \* pi / 180) \* side;

double d = sqrt(2 \* side \* side - 2 \* side \* side \* (cos((180 - alpha) \* pi / 180)));

v[2].first = cos(45 \* pi / 180) \* side;

v[2].second = cos(45 \* pi / 180) \* side;

} else {

for (int i = 0; i < 4; i++){

v[i].first = 0;

v[i].second = 0;

}

}

obj.Set(v);

return in;

}

friend std::ostream& operator<<(std::ostream& out, Rhomb &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

#endif

figure.h

#ifndef FIGURE\_H

#define FIGURE\_H

class Figure{

public:

virtual void Set (std::vector <std::pair<double,double>> cord) = 0;

virtual std::string GetName() = 0;

virtual std::vector<std::pair<double, double>> GetCoord() = 0;

};

#endif

factory.h

#ifndef FACTORY\_H

#define FACTORY\_H

#include <algorithm>

#include <utility>

#include <iostream>

#include <vector>

#include <stack>

#include <string>

#include <cmath>

#include "figure.h"

#include <fstream>

#include "trapeze.h"

#include "rectangle.h"

#include "rhomb.h"

#include <sstream>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

class Factory {

private:

std::string file\_name;

std::string doc\_name;

std::ofstream file;

std::stack<std::pair <std::string, Figure\*>> comand;

bool check\_empty;

std::vector <Figure\*> figures;

public:

Factory(){

file\_name = "";

doc\_name = "";

check\_empty = true;

}

bool CreateDocument(std::string name\_f, std::string name\_d){

file\_name = name\_f;

doc\_name = name\_d;

if (file.is\_open()){

file.close();

}

file.open(file\_name);

if (!file.is\_open()){

std::cout << "cant open file\n";

return false;

}

if (comand.size() != 0){

while (!comand.empty()){

comand.pop();

}

}

figures.clear();

check\_empty = false;

std::cout << "Succesfull\n";

return true;

}

bool Save(){

if (check\_empty){

std::cout << "document is empty\n";

return false;

}

if (file.is\_open()){

file.close();

file.open(file\_name);

}

file << doc\_name << " " << "count figures = " << figures.size() << '\n';

for (auto fig : figures){

file << fig->GetName() << '\n';

std::vector <std::pair<double, double>> v = fig->GetCoord();

for (int i = 0; i < 4; i++){

file << v[i].first << " " << v[i].second << '\n';

}

}

file.close();

return true;

}

bool IsEmpty(){

return check\_empty;

}

bool IsCorrect(std::string check){

for (int i = 0; i < check.size(); i++){

if (check[i] < '0' || check[i] > '9'){

return false;

}

}

return true;

}

bool AddRectangle(){

if (check\_empty){

std::cout << "document is empty\n";

return false;

}

std::cout << "input 2 coordinates for 2 vertex\n";

std::vector<std::pair<double, double>> tmp(4);

std::string tmp1;

std::cin >> tmp1;

if (!IsCorrect(tmp1)){

std::cout << "not correct input\n";

return false;

}

tmp[0].first = atof(tmp1.c\_str());

std::cin >> tmp1;

if (!IsCorrect(tmp1)){

std::cout << "not correct input\n";

return false;

}

tmp[0].second = atof(tmp1.c\_str());

std::cin >> tmp1;

if (!IsCorrect(tmp1)){

std::cout << "not correct input\n";

return false;

}

tmp[2].first = atof(tmp1.c\_str());

std::cin >> tmp1;

if (!IsCorrect(tmp1)){

std::cout << "not correct input\n";

return false;

}

tmp[2].second = atof(tmp1.c\_str());

tmp[1].first = tmp[0].first;

tmp[1].second = tmp[2].second;

tmp[3].first = tmp[2].first;

tmp[3].second = tmp[0].second;

Rectangle temp;

temp.Set(tmp);

Figure \*fig1 = new Rectangle(temp);

Figure \*fig2 = new Rectangle(temp);

figures.push\_back(fig1);

comand.push(std::make\_pair("add", fig2));

return true;

}

bool AddTrapeze(){

if (check\_empty){

std::cout << "document is empty\n";

return false;

}

std::cout << "input 2 coordinats for 4 vertex\n";

std::vector <std::pair<double, double>> tmp(4);

std::string tmp1,tmp2;

for (int i = 0; i < 4; i++){

std::cin >> tmp1 >> tmp2;

if (!IsCorrect(tmp1) || !IsCorrect(tmp2)){

std::cout << "not correct input\n";

return false;

}

tmp[i].first = atof(tmp1.c\_str());

tmp[i].second = atof(tmp2.c\_str());

}

Trapeze temp;

temp.Set(tmp);

Figure \*fig1 = new Trapeze(temp);

Figure \*fig2 = new Trapeze(temp);

figures.push\_back(fig1);

comand.push(std::make\_pair("add", fig2));

return true;

}

bool AddRhomb(){

if (check\_empty){

std::cout << "document is empty\n";

return false;

}

std::cout << "input angle and side\n";

std::vector <std::pair<double, double>> v(4);

std::string tmp1,tmp2;

double pi = M\_PI, angle,side;

std::cin >> tmp1 >> tmp2;

if (!IsCorrect(tmp1) || !IsCorrect(tmp2)){

std::cout << "not correct input\n";

return false;

}

angle = atof(tmp1.c\_str());

side = atof(tmp2.c\_str());

v[0].first = 0;

v[0].second = 0;

if (angle <= 90 && side > 0){

double alpha = (90 - angle) / 2;

v[3].second = sin(alpha \* pi / 180) \* side;

v[3].first = cos(alpha \* pi / 180) \* side;

v[1].first = sin(alpha \* pi / 180) \* side;

v[1].second = cos(alpha \* pi / 180) \* side;

v[2].second = sin(alpha \* pi / 180) \* side + v[1].second;

v[2].first = cos(alpha \* pi / 180) \* side + v[1].first;

} else if (angle < 180 && side > 0) {

double alpha = (270 - angle) / 2;

v[3].second = cos(alpha \* pi / 180) \* side;

v[3].first = sin(alpha \* pi / 180) \* side;

v[1].first = cos(alpha \* pi / 180) \* side;

v[1].second = sin(alpha \* pi / 180) \* side;

double d = sqrt(2 \* side \* side - 2 \* side \* side \* (cos((180 - alpha) \* pi / 180)));

v[2].first = cos(45 \* pi / 180) \* side;

v[2].second = cos(45 \* pi / 180) \* side;

} else {

std::cout << "wrong input\n";

return false;

}

Rhomb temp;

temp.Set(v);

Figure \*fig1 = new Rhomb(temp);

Figure \*fig2 = new Rhomb(temp);

figures.push\_back(fig1);

comand.push(std::make\_pair("add", fig2));

return true;

}

bool RemoveFigure(){

if (check\_empty){

std::cout << "document is empty\n";

return false;

}

Figure \*fig = figures[figures.size() - 1];

figures.pop\_back();

comand.push(std::make\_pair("rem", fig));

return true;

}

void PrintDocument(){

if (check\_empty){

std::cout << "document is empty\n";

return;

}

for (auto fig : figures){

std::cout << fig->GetName() << '\n';

for (int i = 0; i < 4; i++){

std::cout << fig->GetCoord()[i].first << " " << fig->GetCoord()[i].second << '\n';

}

}

}

bool UnDo(){

if (comand.empty()){

std::cout << "if firsts comand\n";

return false;

}

if (comand.top().first == "add"){

comand.pop();

figures.pop\_back();

return true;

}

figures.push\_back(comand.top().second);

comand.pop();

return true;

}

bool Load(std::string load\_file){

std::string tmp1,tmp2,tmp3,tmp4;

std::ifstream ofile(load\_file);

figures.clear();

if (comand.size() != 0){

while (!comand.empty()){

comand.pop();

}

}

if (!ofile.is\_open()){

std::cout << "cant open file\n";

return false;

}

ofile >> tmp1 >> tmp2 >> tmp3 >> tmp4;

if (check\_empty){

doc\_name = tmp1;

file\_name = load\_file;

check\_empty = true;

}

std::vector <std::pair <double, double>> cord(4);

int n;

ofile >> n;

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

ofile >> tmp1;

for (int j = 0; j < 4; j++){

ofile >> cord[j].first >> cord[j].second;

}

if (tmp1 == "rectangle"){

Rectangle temp;

temp.Set(cord);

Figure \*fig = new Rectangle(temp);

figures.push\_back(fig);

} else if (tmp1 == "trapeze"){

Trapeze temp;

temp.Set(cord);

Figure \*fig = new Trapeze(temp);

figures.push\_back(fig);

} else {

Rhomb temp;

temp.Set(cord);

Figure \*fig = new Rhomb(temp);

figures.push\_back(fig);

}

}

ofile.close();

return true;

}

void Show(){

std::ofstream off;

off.open("temp.txt");

std::vector <std::pair < double, double>> tmp(4);

for (auto fig : figures){

tmp = fig->GetCoord();

for (int j = 0; j < 4; j++){

off << tmp[j].first << " " << tmp[j].second << " ";

}

}

off.close();

int status;

int id = fork();

if (id == 0){

execl("testp.py", "testp.py", NULL);

} else {

waitpid(id, &status, 0);

remove("temp.txt");

}

}

};

#endif

main.cpp

// Лабораторная работа №7 Савров Никита М80-207Б-18

// Спроектировать простейший графический векторный редактор.

// Требование к функционалу редактора:

// • создание нового документа

// • импорт документа из файла

// • экспорт документа в файл

// • создание графического примитива (согласно варианту задания)

// • удаление графического примитива

// • отображение документа на экране (печать перечня графических объектов и их характеристик)

// • реализовать операцию undo, отменяющую последнее сделанное действие. Должно действовать для

// операций добавления/удаления фигур.

// Требования к реализации:

// • Создание графических примитивов необходимо вынести в отдельный класс – Factory.

// • Сделать упор на использовании полиморфизма при работе с фигурами;

// • Взаимодействие с пользователем (ввод команд) реализовать в функции main;

#include "factory.h"

int main(){

std::string input;

Factory docs;

bool check;

std::cout << "1 - create new document\n2 - add trapeze\n3 - add rectangle\n4 - add rhomb\n5 - remove last figure\n6 - print document\n7 - undo\n8 - save\n9 - load\n0 - show\n";

while (std::cin >> input){

if (input == "1"){

std::string name\_d, name\_f;

std::cout << "input name doc and file\n";

std::cin >> name\_d >> name\_f;

docs.CreateDocument(name\_f,name\_d);

} else if (input == "2"){

check = docs.AddTrapeze();

} else if (input == "3"){

check = docs.AddRectangle();

} else if (input == "4"){

check = docs.AddRhomb();

} else if (input == "5"){

check = docs.RemoveFigure();

} else if (input == "6"){

docs.PrintDocument();

} else if(input == "7"){

check = docs.UnDo();

} else if(input == "8"){

check = docs.Save();

} else if(input == "9"){

std::string name;

std::cout << "input name file\n";

std::cin >> name;

check = docs.Load(name);

} else if(input == "0"){

docs.Show();

} else {

std::cout << "check menu\n";

}

std::cout << "1 - create new document\n2 - add trapeze\n3 - add rectangle\n4 - add rhomb\n5 - remove last figure\n6 - print document\n7 - undo\n8 - save\n9 - load\n0 - show\n";

}

}

https://github.com/trol53/oop\_exercise\_01/tree/master/oop\_exercise\_07

6.Вывод

Данная программа может быть полезна для работы с геометрическими фигурами. Так-же ввиду того что классы фигур наследуются от одного общего класса позволяет писать более гибкий код. Благодаря парадигме ООП мы можем представлять пользовательские типы данных в виде объектов что значительно упрощает программирование. Так-же эта работа показывает практическое применение наследования классов.

Литература

1.Онлайн библиотека c++ cppreference[электронный ресурс] URL:<https://ru.cppreference.com/w/cpp/container/stack>

(дата обращения 20.12.2019)

2.Информационный портал хабр [электронный ресурс] URL: [https://habr.com/ru/post/112997](https://habr.com/ru/post/112997/)/

(дата обращения 19.12.2019)