**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 7**

Тема: Проектирование структуры классов

Студент: Хренникова Ангелина Сергеевна

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

Цель: Получить практические навыки в хороших практиках проектирования структуры классов приложения.

Задача: Спроектировать простейший «графический» векторный редактор.

Требования к функционалу редактора:

* Создание нового документ.
* 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 в std::cout).
* 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.

Фигуры: пятиугольник, шестиугольник, восьмиугольник.

1. **Описание программы**

Программа позволяет создавать графические примитивы фигур пятиугольник, шестиугольник, восьмиугольник, записывать и считывать их из файла, удалять, а также отменять действие последней операции с помощью команды undo. Запись производится в указанный файл. Все операции с файлами производит редактор. В программе есть специальное меню, позволяющее выбрать определенное действие.

Пятиугольник, шестиугольник и восьмиугольник задаются координатами центральной точкой и одной из точек самой фигуры, остальные точки вычисляются программой. Для каждой фигуры печатаются координаты, центр и площадь. Площадь вычисляется по формуле: , где R - радиус описанной окружности(разность координат центра и одной из точек многоугольника), n - количество углов многоугольника.

1. **Набор тестов**

Тест1: 1 file 4 pentagon 0 0 0 6 4 hexagon 0 0 0 6 4 octagon 0 0 0 6 6 5 1 2 1 file1 4 hexagon 0 0 0 1 6 3 file 6 4 octagon 0 0 0 1 6 7 6 8

1. Создается файл с именем: file.
2. Добавляются фигуры: пятиугольник, шестиугольник, восьмиугольник с центром в точке С(0; 0) и точкой А(0; 6).
3. Печатаются все фигуры с их характеристиками.
4. Удаляется фигура с индексом 1.
5. Все фигуры сохраняются в файл.
6. Создается файл с именем: file1. В него добавляется фигура: шестиугольник с центром в точке С(0; 0) и точкой А(0; 1).
7. Загружается файл с именем: file. В него добавляется фигура: восьмиугольник с центром в точке С(0; 0) и точкой А(0; 1).
8. Отменяется добавление последней команды, то есть добавление восьмиугольника.
9. Выход из программы.

Тест2: 3 file 6 4 hexagon 0 0 0 2 6 5 2 6 7 6 8

1. Загружает файл с именем: file.
2. Добавляет фигуру: шестиугольник с центром в точке С(0; 0) и точкой А(0; 2).
3. Удаление фигуры с индексом 2 и отмена этого действия.
4. Выход из программы.
5. **Результаты выполнения тестов**

Тест1:

1. Create a new file

2. Save data to a file

3. Load data from a file

4. Add a figure

pentagon

hexagon

octagon

5. Remove a figure by index

6. Print all the current figures

7. Undo the last addition / removal action

8. Exit

Select menu item: 1

file

OK

Select menu item: 4

Select figure and enter cords: pentagon

0 0 0 6

OK

Select menu item: 4

Select figure and enter cords: hexagon

0 0 0 6

OK

Select menu item: 4

Select figure and enter cords: octagon

0 0 0 6

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Hexagon:

(0; 6) (-5.19615; 3) (-5.19615; -3) (0; -6) (5.19615; -3) (5.19615; 3)

Center: 0 0

Area: 93.5307

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 5

Enter index: 1

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 2

OK

Select menu item: 1

file1

OK

Select menu item: 4

Select figure and enter cords: hexagon

0 0 0 1

OK

Select menu item: 6

Hexagon:

(0; 1) (-0.866025; 0.5) (-0.866025; -0.5) (-1.11022e-16; -1) (0.866025; -0.5) (0.866025; 0.5)

Center: 0 0

Area: 2.59808

Select menu item: 3

Enter name file: file

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 4

Select figure and enter cords: octagon

0 0 0 1

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Octagon:

(0; 1) (-0.707107; 0.707107) (-1; 0) (-0.707107; -0.707107) (0; -1) (0.707107; -0.707107) (1; 0) (0.707107; 0.707107)

Center: 0 0

Area: 2.82843

Select menu item: 7

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 8

Тест2:

1. Create a new file

2. Save data to a file

3. Load data from a file

4. Add a figure

pentagon

hexagon

octagon

5. Remove a figure by index

6. Print all the current figures

7. Undo the last addition / removal action

8. Exit

Select menu item: 3

Enter name file: file

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 4

Select figure and enter cords: hexagon

0 0 0 2

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Hexagon:

(0; 2) (-1.73205; 1) (-1.73205; -1) (-2.22045e-16; -2) (1.73205; -1) (1.73205; 1)

Center: 0 0

Area: 10.3923

Select menu item: 5

Enter index: 2

OK

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Select menu item: 7

Select menu item: 6

Pentagon:

(0; 6) (-5.70634; 1.8541) (-3.52671; -4.8541) (3.52671; -4.8541) (5.70634; 1.8541)

Center: 0 0

Area: 85.5951

Octagon:

(0; 6) (-4.24264; 4.24264) (-6; 0) (-4.24264; -4.24264) (0; -6) (4.24264; -4.24264) (6; 0) (4.24264; 4.24264)

Center: 0 0

Area: 101.823

Hexagon:

(0; 2) (-1.73205; 1) (-1.73205; -1) (-2.22045e-16; -2) (1.73205; -1) (1.73205; 1)

Center: 0 0

Area: 10.3923

Select menu item: 8

1. **Листинг программы**

#include <iostream>

#include <fstream>

#include <memory>

#include <string>

#include <vector>

#include <algorithm>

#include <cstdint>

#include <stack>

#include <cmath>

struct point { // Координаты точки

double x, y;

point(double a, double b) {

x = a, y = b;

};

point() = default;

};

std::istream& operator >> (std::istream& is, point& p) { // Ввод точки

return is >> p.x >> p.y;

}

std::ostream& operator << (std::ostream& os, const point& p) { // Вывод точки

return os << p.x << ' ' << p.y;

}

struct figure { //Фигура

virtual point center() const = 0;

virtual void print(std::ostream&) const = 0;

virtual void printFile(std::ofstream&) const = 0;

virtual double area() const = 0;

virtual ~figure() = default;

};

struct pentagon : figure { // Пятиугольник

private:

point a0, a1; // Центр и точка

public:

point center() const override { // Центр пятиугольника

return a0;

};

void print(std::ostream& os) const override { // Вывод пятиугольника и его характеристик

double c = (sqrt(5) - 1) / 4;

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

double a2x = a0.x + (a1.x - a0.x) \* c - (a1.y - a0.y) \* s;

double a2y = a0.y + (a1.x - a0.x) \* s + (a1.y - a0.y) \* c;

double a3x = a0.x + (a2x - a0.x) \* c - (a2y - a0.y) \* s;

double a3y = a0.y + (a2x - a0.x) \* s + (a2y - a0.y) \* c;

double a4x = a0.x + (a3x - a0.x) \* c - (a3y - a0.y) \* s;

double a4y = a0.y + (a3x - a0.x) \* s + (a3y - a0.y) \* c;

double a5x = a0.x + (a4x - a0.x) \* c - (a4y - a0.y) \* s;

double a5y = a0.y + (a4x - a0.x) \* s + (a4y - a0.y) \* c;

os << "Pentagon:\n" << "(" << a1.x << "; " << a1.y << ") " <<

"(" << a2x << "; " << a2y << ") " << "(" << a3x << "; " << a3y << ") " <<

"(" << a4x << "; " << a4y << ") " << "(" << a5x << "; " << a5y << ")" << '\n';

os << "Center: " << center() << '\n' << "Area: " << area() << '\n';

};

void printFile(std::ofstream& of) const override { // Запись в файл

of << "pentagon\n" << a0 << '\n' << a1;

};

double area() const override { // Площадь пятиугольника

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

double r = sqrt(pow(a1.x - a0.x, 2) + pow(a1.y - a0.y, 2));

double area = 0.5 \* r \* r \* 5 \* s;

return area;

};

pentagon() = default;

pentagon(std::istream& is) { // Ввод пятиугольника

is >> a0 >> a1;

};

pentagon(std::ifstream& is) { // Чтение из файла

is >> a0 >> a1;

};

};

struct hexagon : figure { // Шестиугольник

private:

point a0, a1;

public:

point center() const override { //Центр шестиугольника

return a0;

};

void print(std::ostream& os) const override { //вывод шестиугольника

double c = 0.5;

double s = sqrt(3) / 2;

double a2x = a0.x + (a1.x - a0.x) \* c - (a1.y - a0.y) \* s;

double a2y = a0.y + (a1.x - a0.x) \* s + (a1.y - a0.y) \* c;

double a3x = a0.x + (a2x - a0.x) \* c - (a2y - a0.y) \* s;

double a3y = a0.y + (a2x - a0.x) \* s + (a2y - a0.y) \* c;

double a4x = a0.x + (a3x - a0.x) \* c - (a3y - a0.y) \* s;

double a4y = a0.y + (a3x - a0.x) \* s + (a3y - a0.y) \* c;

double a5x = a0.x + (a4x - a0.x) \* c - (a4y - a0.y) \* s;

double a5y = a0.y + (a4x - a0.x) \* s + (a4y - a0.y) \* c;

double a6x = a0.x + (a5x - a0.x) \* c - (a5y - a0.y) \* s;

double a6y = a0.y + (a5x - a0.x) \* s + (a5y - a0.y) \* c;

os << "Hexagon:\n" << "(" << a1.x << "; " << a1.y << ") " <<

"(" << a2x << "; " << a2y << ") " << "(" << a3x << "; " << a3y << ") " <<

"(" << a4x << "; " << a4y << ") " << "(" << a5x << "; " << a5y << ") " <<

"(" << a6x << "; " << a6y << ")" << '\n';

os << "Center: " << center() << '\n' << "Area: " << area() << '\n';

};

void printFile(std::ofstream& of) const override { //вывод в файл

of << "hexagon\n" << a0 << '\n' << a1;

};

double area() const override { //площадь

double area = pow(sqrt(pow(a1.x - a0.x, 2) + pow(a1.y - a0.y, 2)), 2) \* 3 \* sqrt(3) \* 0.5;

return area;

};

hexagon() = default;

hexagon(std::istream& is) { //чтение

is >> a0 >> a1;

};

hexagon(std::ifstream& is) { //чтение из файла

is >> a0 >> a1;

};

};

struct octagon : figure { //восьмиугольник

private:

point a0, a1;

public:

point center() const override { //центр

return a0;

};

void print(std::ostream& os) const override { //вывод

double c = sqrt(2) / 2;

double s = sqrt(2) / 2;

double a2x = a0.x + (a1.x - a0.x) \* c - (a1.y - a0.y) \* s;

double a2y = a0.y + (a1.x - a0.x) \* s + (a1.y - a0.y) \* c;

double a3x = a0.x + (a2x - a0.x) \* c - (a2y - a0.y) \* s;

double a3y = a0.y + (a2x - a0.x) \* s + (a2y - a0.y) \* c;

double a4x = a0.x + (a3x - a0.x) \* c - (a3y - a0.y) \* s;

double a4y = a0.y + (a3x - a0.x) \* s + (a3y - a0.y) \* c;

double a5x = a0.x + (a4x - a0.x) \* c - (a4y - a0.y) \* s;

double a5y = a0.y + (a4x - a0.x) \* s + (a4y - a0.y) \* c;

double a6x = a0.x + (a5x - a0.x) \* c - (a5y - a0.y) \* s;

double a6y = a0.y + (a5x - a0.x) \* s + (a5y - a0.y) \* c;

double a7x = a0.x + (a6x - a0.x) \* c - (a6y - a0.y) \* s;

double a7y = a0.y + (a6x - a0.x) \* s + (a6y - a0.y) \* c;

double a8x = a0.x + (a7x - a0.x) \* c - (a7y - a0.y) \* s;

double a8y = a0.y + (a7x - a0.x) \* s + (a7y - a0.y) \* c;

os << "Octagon:\n" << "(" << a1.x << "; " << a1.y << ") " <<

"(" << a2x << "; " << a2y << ") " << "(" << a3x << "; " << a3y << ") " <<

"(" << a4x << "; " << a4y << ") " << "(" << a5x << "; " << a5y << ") " <<

"(" << a6x << "; " << a6y << ") " << "(" << a7x << "; " << a7y << ") " <<

"(" << a8x << "; " << a8y << ") " << '\n';

os << "Center: " << center() << '\n' << "Area: " << area() << '\n';

};

void printFile(std::ofstream& of) const override { //вывод в файл

of << "octagon\n" << a0 << '\n' << a1;

};

double area() const override { //площадь

double r = sqrt(pow(a1.x - a0.x, 2) + pow(a1.y - a0.y, 2));

double area = 4 \* r \* r \* (sqrt(2) / 2);

return area;

};

octagon() = default;

octagon(std::istream& is) { //чтение

is >> a0 >> a1;

};

octagon(std::ifstream& is) { //чтение из файла

is >> a0 >> a1;

};

};

struct factory { //фигуры

std::shared\_ptr<figure> FigureCreate(std::istream& is) {

std::string n;

is >> n;

if (n == "pentagon") {

return std::shared\_ptr<figure>(new pentagon(is));

}

else if (n == "hexagon") {

return std::shared\_ptr<figure>(new hexagon(is));

}

else if (n == "octagon") {

return std::shared\_ptr<figure>(new octagon(is));

}

else {

throw std::logic\_error("There is no such figure\n");

}

};

std::shared\_ptr<figure> FigureCreateFile(std::ifstream& is) {

std::string n;

is >> n;

if (n == "pentagon") {

return std::shared\_ptr<figure>(new pentagon(is));

}

else if (n == "hexagon") {

return std::shared\_ptr<figure>(new hexagon(is));

}

else if (n == "octagon") {

return std::shared\_ptr<figure>(new octagon(is));

}

else {

throw std::logic\_error("There is no such figure\n");

}

};

};

struct document {

public:

void Print() const {

if (buffer\_.empty()) {

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

}

for (auto elem : buffer\_) {

elem->print(std::cout);

}

};

document(std::string& newName) : name\_(newName), factory\_(), buffer\_(0) {};

void Insert(std::shared\_ptr<figure>& ptr) {

buffer\_.push\_back(ptr);

};

void Rename(const std::string& newName) {

name\_ = newName;

};

void Save(const std::string& filename) const {

std::ofstream fout;

fout.open(filename);

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

throw std::runtime\_error("File is not opened\n");

}

fout << buffer\_.size() << '\n';

for (auto elem : buffer\_) {

elem->printFile(fout);

}

};

void Load(const std::string& filename) {

std::ifstream fin;

fin.open(filename);

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

throw std::runtime\_error("File is not opened\n");

}

size\_t size;

fin >> size;

buffer\_.clear();

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

buffer\_.push\_back(factory\_.FigureCreateFile(fin));

}

name\_ = filename;

};

std::shared\_ptr<figure> GetFigure(uint32\_t index) {

return buffer\_[index];

};

void Erase(uint32\_t index) {

if (index >= buffer\_.size()) {

throw std::logic\_error("Out of bounds\n");

}

buffer\_[index] = nullptr;

for (; index < buffer\_.size() - 1; ++index) {

buffer\_[index] = buffer\_[index + 1];

}

buffer\_.pop\_back();

};

std::string GetName() {

return this->name\_;

};

size\_t Size() {

return buffer\_.size();

};

private:

friend class InsertCommand;

friend class DeleteCommand;

factory factory\_;

std::string name\_;

std::vector<std::shared\_ptr<figure>> buffer\_;

void RemoveLast() {

if (buffer\_.empty()) {

throw std::logic\_error("Document is empty");

}

buffer\_.pop\_back();

};

void InsertIndex(std::shared\_ptr<figure>& newFigure, uint32\_t index) {

buffer\_.insert(buffer\_.begin() + index, newFigure);

};

};

struct Acommand {

public:

virtual ~Acommand() = default;

virtual void UnExecute() = 0;

protected:

std::shared\_ptr<document> doc\_;

};

struct InsertCommand : public Acommand {

public:

void UnExecute() override {

doc\_->RemoveLast();

};

InsertCommand(std::shared\_ptr<document>& doc) {

doc\_ = doc;

};

};

struct DeleteCommand : public Acommand {

public:

DeleteCommand(std::shared\_ptr<figure>& newFigure, uint32\_t newIndex, std::shared\_ptr<document>& doc) {

doc\_ = doc;

figure\_ = newFigure;

index\_ = newIndex;

};

void UnExecute() override {

doc\_->InsertIndex(figure\_, index\_);

};

private:

std::shared\_ptr<figure> figure\_;

uint32\_t index\_;

};

struct editor {

private:

std::shared\_ptr<document> doc\_;

std::stack<std::shared\_ptr<Acommand>> history\_;

public:

~editor() = default;

void PrintDocument() {

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

doc\_->Print();

};

void CreateDocument(std::string& newName) {

doc\_ = std::make\_shared<document>(newName);

};

bool DocumentExist() {

return doc\_ != nullptr;

};

editor() : doc\_(nullptr), history\_() {}

void InsertInDocument(std::shared\_ptr<figure>& newFigure) { //Вставка

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

std::shared\_ptr<Acommand> command = std::shared\_ptr<Acommand>(new InsertCommand(doc\_));

doc\_->Insert(newFigure);

history\_.push(command);

};

void DeleteInDocument(uint32\_t index) { //Удаление

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

if (index >= doc\_->Size()) {

std::cout << "Out of bounds\n";

return;

}

std::shared\_ptr<figure> tmp = doc\_->GetFigure(index);

std::shared\_ptr<Acommand> command = std::shared\_ptr<Acommand>(new DeleteCommand(tmp, index, doc\_));

doc\_->Erase(index);

history\_.push(command);

};

void SaveDocument() { //запись документа

if (doc\_ == nullptr) {

std::cout << "No document!\nNot ";

return;

}

std::string saveName = doc\_->GetName();

doc\_->Save(saveName);

};

void LoadDocument(std::string& name) { //чтение из документа

try {

doc\_ = std::make\_shared<document>(name);

doc\_->Load(name);

while (!history\_.empty()) {

history\_.pop();

}

}

catch (std::logic\_error& e) {

std::cout << e.what();

}

};

void Undo() { //отмена последнего действия

if (history\_.empty()) {

throw std::logic\_error("History is empty\n");

}

std::shared\_ptr<Acommand> lastCommand = history\_.top();

lastCommand->UnExecute();

history\_.pop();

};

};

void help() {

std::cout <<

"1. Create a new file\n"

"2. Save data to a file\n"

"3. Load data from a file\n"

"4. Add a figure\n"

"\tpentagon\n"

"\thexagon\n"

"\toctagon\n"

"5. Remove a figure by index\n"

"6. Print all the current figures\n"

"7. Undo the last addition / removal action\n"

"8. Exit\n";

}

void create(editor& edit, std::string& tmp) {

edit.CreateDocument(tmp);

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

}

void load(editor& edit) {

std::string tmp;

std::cin >> tmp;

try {

edit.LoadDocument(tmp);

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

}

catch (std::runtime\_error& e) {

std::cout << e.what();

}

}

void save(editor& edit) {

std::string tmp;

try {

edit.SaveDocument();

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

}

catch (std::runtime\_error& e) {

std::cout << e.what();

}

}

void add(editor& edit) {

factory fac;

try {

std::shared\_ptr<figure> newElem = fac.FigureCreate(std::cin);

edit.InsertInDocument(newElem);

}

catch (std::logic\_error& e) {

std::cout << e.what() << '\n';

}

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

}

void remove(editor& edit, int& index) {

try {

edit.DeleteInDocument(index);

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

}

catch (std::logic\_error& err) {

std::cout << err.what() << "\n";

}

}

int main() {

help();

editor edit;

std::string namefile;

int index;

int command;

std::cout << "Select menu item: ";

std::cin >> command;

while (command > 0 && command < 8) {

if (command == 1) {

std::cin >> namefile;

create(edit, namefile);

}

if (command == 2) {

save(edit);

}

if (command == 3) {

std::cout << "Enter name file: ";

load(edit);

}

if (command == 4) {

std::cout << "Select figure snd enter cords: ";

add(edit);

}

if (command == 5) {

std::cout << "Enter index: ";

std::cin >> index;

remove(edit, index);

}

if (command == 6) {

edit.PrintDocument();

}

if (command == 7) {

try {

edit.Undo();

}

catch (std::logic\_error& e) {

std::cout << e.what();

}

}

std::cout << "Select menu item: ";

std::cin >> command;

}

return 0;

}

1. **Вывод**

В данной лабораторной работе я получила практические навыки в проектировании структуры классов приложения. Класс проектирования представляет абстракцию одного или нескольких классов в реализации системы. Классы определяют объекты, которые, в свою очередь, реализуют варианты использования. Класс основан на требованиях, предъявляемых реализациями вариантов использования объектам системы, а также на всех ранее разработанных моделях объектов. В проектировании это очень удобный инструмент, который позволяет структурировать код и избежать повторений.

**Список литературы**

1. Справочник по языку С++/Электронный ресурс/Режим доступа: <https://en.cppreference.com/w/> (дата обращения: 19.12.2020).
2. Паттерн проектирования/Электронный ресурс/Режим доступа:<http://cpp-reference.ru/patterns/creational-patterns/factory-method/> (дата обращения: 20.12.2020).
3. Проектирование и наследование/Электронный ресурс/Режим доступа: <http://www.williamspublishing.com/PDF/5-8459-0859-0/part.pdf> (дата обращения: 20.12.2020).