**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: Чернобаев Андрей Александрович

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

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 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;

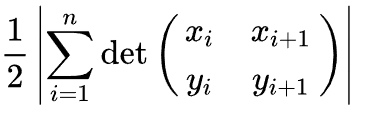
· Взаимодействие с пользователем (ввод команд) реализовать в функции main;

**Варианты заданий (выпуклые равносторонние фигуры вращения):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант** | **Фигура №1** | **Фигура №2** | **Фигура №3** |
| 9. | Треугольник | Квадрат | Прямоугольник |

1. Описание программы

Площадь фигур вычисляется с помощью формулы площади Гаусса:



Программа состоит из 14 файлов:

Command.cpp - реализация классов из Command.h

Command.h - реализация классов команд

Состоит из:

* 3-х классов.

Document.cpp - реализация класса из Document.h

Document.h - заголовочный файл для реализации документа

Состоит из:

* класса Document

Editor.cpp - реализация класса Editor из Editor.h

Editor.h - заголовочный файл для реализации самого графического редактора.

Состоит из:

* класса Editor

Factory.cpp - реализация класса Factory из Factory.h

Factory.h - заголовочный файл для реализации паттерна фабрика.

Состоит из:

* класса Factory со статическими функциями

Figure.h - класс с абстрактным классом фигуры

Состоит из:

* класса Figure
* функции вычисления детерминанта
* функции вычисления площади многоугольника с помощью формулы Гаусса

Point.h - заголовочный файл с классом Point, отвечающим за точку

Состоит из:

* класса Point

Rectangle.h - заголовочный файл с классом Rectangle, отвечающим за треугольник

Состоит из:

* класса Rectangle

Square.h - заголовочный файл с классом Square, отвечающим за квадрат

Состоит из:

* класса Square

Triangle.h - заголовочный файл с классом Square, отвечающим за квадрат

Состоит из:

* класса Triangle

1. Набор тестов

из файла **test\_01.txt**:

create\_doc

first.txt

add\_figure

square 1 1 1 1 1 1 1 1

add\_figure

rectangle 0 0 1 1 0 1 1 0

remove\_figure

0

undo

save\_doc

print\_doc

print\_doc

add\_figure

square 0 0 2 2 2 0 0 2

undo

exit

Из файла **test\_02.txt**

create\_doc

second.txt

add\_figure

square 0 0 2 2 0 2 2 0

add\_figure

rectangle 0 0 5 6 5 0 0 6

add\_figure

triangle 0 0 2 2 6 7

print\_doc

save\_doc

remove\_figure

1

undo

undo

add\_figure

square 0 0 3 3 0 3 3 0

undo

undo

print\_doc

remove\_figure

0

undo

exit

1. Результаты выполнения тестов

|  |  |
| --- | --- |
| input | output |
| create\_doc  first.txt  add\_figure  square 1 1 1 1 1 1 1 1  add\_figure  rectangle 0 0 1 1 0 1 1 0  remove\_figure  0  undo  save\_doc  print\_doc  print\_doc  add\_figure  square 0 0 2 2 2 0 0 2  undo  exit | enter name of new document  document create  was added  was added  enter index  was removed  save document  square  1 1  1 1  1 1  1 1  rectangle  0 0  1 1  0 1  1 0  square  1 1  1 1  1 1  1 1  rectangle  0 0  1 1  0 1  1 0  was added |
| create\_doc  second.txt  add\_figure  square 0 0 2 2 0 2 2 0  add\_figure  rectangle 0 0 5 6 5 0 0 6  add\_figure  triangle 0 0 2 2 6 7  print\_doc  save\_doc  remove\_figure  1  undo  undo  add\_figure  square 0 0 3 3 0 3 3 0  undo  undo  print\_doc  remove\_figure  0  undo  exit | enter name of new document  document create  was added  was added  was added  square  0 0  2 2  0 2  2 0  rectangle  0 0  5 6  5 0  0 6  triangle  0 0  2 2  6 7  save document  enter index  was removed  was added  square  0 0  2 2  0 2  2 0  enter index  was removed |

1. Листинг программы

из файла **Command.cpp**

void InsertCommand::unexecute() {

doc\_->remove\_last();

}

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

doc\_ = doc;

}

RemoveCommand::RemoveCommand(std::shared\_ptr<Figure> &newFigure, size\_t newIndex, std::shared\_ptr<Document> &doc) {

doc\_ = doc;

figure\_ = newFigure;

index\_ = newIndex;

}

void RemoveCommand::unexecute() {

doc\_->insert\_index(figure\_,index\_);

}

из файла **Command.h**:

#ifndef OOP7\_COMMAND\_H

#define OOP7\_COMMAND\_H

#include "Document.h"

class ICommand {

public:

virtual void unexecute() = 0;

protected:

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

};

class InsertCommand : public ICommand {

public:

void unexecute() override;

explicit InsertCommand(std::shared\_ptr<Document>& doc);

};

class RemoveCommand : public ICommand {

public:

RemoveCommand(std::shared\_ptr<Figure>& newFigure, size\_t newIndex, std::shared\_ptr<Document>& doc);

void unexecute() override;

private:

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

size\_t index\_;

};

#include "Command.cpp"

#endif //OOP7\_COMMAND\_H

из файла **Document.cpp**:

void Document::print() const {

if (buffer\_.empty()) {

std::cout << "buffer is empty" << std::endl;

}

for (const auto& elem : buffer\_) {

elem->print(std::cout);

}

}

Document::Document(std::string& newName) {

name\_ = newName;

}

void Document::insert(std::shared\_ptr<Figure>& ptr) {

buffer\_.push\_back(ptr);

}

void Document::save(const std::string& filename) const {

std::ofstream fout;

fout.open(filename);

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

throw std::runtime\_error("Can't save to file. No file");

}

fout << buffer\_.size() << std::endl;

for (const auto& elem : buffer\_) {

elem->print\_file(fout);

}

}

void Document::load(const std::string& filename) {

std::ifstream fin;

fin.open(filename);

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

throw std::runtime\_error("Cant't load file. No file");

}

size\_t size;

fin >> size;

buffer\_.clear();

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

buffer\_.push\_back(Factory::figure\_create\_file(fin));

}

name\_ = filename;

}

std::shared\_ptr<Figure> Document::get\_figure(size\_t index) {

return buffer\_[index];

}

void Document::remove\_at(size\_t index) {

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

throw std::out\_of\_range("out of range");

}

buffer\_.erase(buffer\_.begin() + index);

}

std::string Document::get\_name() {

return this->name\_;

}

size\_t Document::size() {

return buffer\_.size();

}

void Document::remove\_last() {

if (buffer\_.empty()) {

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

}

buffer\_.pop\_back();

}

void Document::insert\_index(std::shared\_ptr<Figure>& newFigure, size\_t index) {

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

}

из файла **Document.h**:

#ifndef OOP7\_DOCUMENT\_H

#define OOP7\_DOCUMENT\_H

#include <fstream>

#include <cstdint>

#include <memory>

#include <string>

#include <algorithm>

#include <vector>

#include "Factory.h"

#include "Figure.h"

class Document {

public:

void print() const ;

explicit Document(std::string& newName);

void insert(std::shared\_ptr<Figure>& ptr);

void save(const std::string& filename) const;

void load(const std::string& filename);

std::shared\_ptr<Figure> get\_figure(size\_t index);

void remove\_at(size\_t index);

std::string get\_name();

size\_t size();

void remove\_last();

void insert\_index(std::shared\_ptr<Figure>& newFigure, size\_t index);

private:

friend class InsertCommand;

friend class RemoveCommand;

std::string name\_;

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

};

#include "Document.cpp"

#endif //OOP7\_DOCUMENT\_H

из файла **Editor.cpp**:

Editor::Editor() {

doc\_ = nullptr;

}

void Editor::print\_document() {

if (doc\_ == nullptr) {

std::cout << "no document" << std::endl;

return;

}

doc\_->print();

}

void Editor::create\_document(std::string& newName) {

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

}

void Editor::insert\_in\_document(std::shared\_ptr<Figure>& newFigure) {

if (doc\_ == nullptr) {

std::cout << "no document" << std::endl;

return;

}

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

doc\_->insert(newFigure);

history\_.push(command);

}

void Editor::delete\_in\_document(size\_t index) {

if (doc\_ == nullptr) {

std::cout << "no document" << std::endl;

return;

}

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

std::cout << "out of range" << std::endl;

return;

}

std::shared\_ptr<Figure> tmp = doc\_->get\_figure(index);

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

doc\_->remove\_at(index);

history\_.push(command);

}

void Editor::save\_document() {

if (doc\_ == nullptr) {

std::cout << "no document" << std::endl;

return;

}

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

doc\_ ->save(saveName);

}

void Editor::load\_document(std::string& name) {

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

doc\_->load(name);

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

history\_.pop();

}

}

void Editor::undo() {

if (history\_.empty()) {

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

}

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

lastCommand->unexecute();

history\_.pop();

}

из файла **Editor.h**:

#ifndef OOP7\_EDITOR\_H

#define OOP7\_EDITOR\_H

#include <stack>

#include "Figure.h"

#include "Document.h"

#include "Command.h"

class Editor {

public:

Editor();

void print\_document();

void create\_document(std::string& newName);

void insert\_in\_document(std::shared\_ptr<Figure>& newFigure);

void delete\_in\_document(size\_t index);

void save\_document();

void load\_document(std::string& name);

void undo();

private:

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

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

};

#include "Editor.cpp"

#endif //OOP7\_EDITOR\_H

из файла **Factory.cpp**:

std::shared\_ptr<Figure> Factory::figure\_create(std::istream& is) {

std::string name;

is >> name;

if ( name == "rectangle" ) {

return std::shared\_ptr<Figure> (new Rectangle(is));

} else if ( name == "triangle") {

return std::shared\_ptr<Figure> (new Triangle(is));

} else if ( name == "square") {

return std::shared\_ptr<Figure> (new Square(is));

} else {

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

}

}

std::shared\_ptr<Figure> Factory::figure\_create\_file(std::ifstream& is) {

std::string name;

is >> name;

if ( name == "rectangle" ) {

return std::shared\_ptr<Figure> (new Rectangle(is));

} else if ( name == "triangle") {

return std::shared\_ptr<Figure> (new Triangle(is));

} else if ( name == "square") {

return std::shared\_ptr<Figure> (new Square(is));

} else {

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

}

}

из файла **Factory.h**:

#ifndef OOP7\_FACTORY\_H

#define OOP7\_FACTORY\_H

#include <memory>

#include <iostream>

#include <fstream>

#include <string>

#include "Square.h"

#include "Rectangle.h"

#include "Triangle.h"

class Factory {

public:

static std::shared\_ptr<Figure> figure\_create(std::istream& is);

static std::shared\_ptr<Figure> figure\_create\_file(std::ifstream& is);

};

#include "Factory.cpp"

#endif //OOP7\_FACTORY\_H

из файла **Figure.h**:

#ifndef OOP7\_FIGURE\_H

#define OOP7\_FIGURE\_H

#include <iostream>

#include <vector>

#include <fstream>

#include "Point.h"

double get\_det(const double x1, const double y1,

const double x2, const double y2){

return x1 \* y2 - x2 \* y1;

}

double gauss\_area(const std::vector<Point>& nodes){

double out = 0.0;

auto count = nodes.size();

for(size\_t i = 0; i < (count - 1); i++){

out += get\_det(static\_cast<double>(nodes[i].x),static\_cast<double>(nodes[i].y),

static\_cast<double>(nodes[i+1].x),static\_cast<double>(nodes[i+1].y));

}

out += get\_det(static\_cast<double>(nodes[count-1].x),static\_cast<double>(nodes[count-1].y),

static\_cast<double>(nodes[0].x),static\_cast<double>(nodes[0].y));

return fabs(0.5 \* out);

}

class Figure {

public:

virtual Point center() const = 0;

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

virtual void print\_file(std::ofstream&) const = 0 ;

virtual double area() const = 0;

};

#endif //OOP7\_FIGURE\_H

Из файла **main.cpp**:

#include <iostream>

#include "Factory.h"

#include "Editor.h"

Editor graphic\_editor;

void menu() {

std::cout << "menu" << std::endl;

std::cout << "create\_doc" << std::endl;

std::cout << "load\_doc" << std::endl;

std::cout << "save\_doc" << std::endl;

std::cout << "add\_figure" << std::endl;

std::cout << "remove\_figure" << std::endl;

std::cout << "print\_doc" << std::endl;

std::cout << "undo" << std::endl;

std::cout << "exit" << std::endl;

}

void create() {

std::string tmp;

std::cout << "enter name of new document" << std::endl;

std::cin >> tmp;

graphic\_editor.create\_document(tmp);

std::cout << "document create" << std::endl;

}

void load() {

std::string tmp;

std::cout << "enter path to the file" << std::endl;

std::cin >> tmp;

graphic\_editor.load\_document(tmp);

std::cout << "document loaded" << std::endl;

}

void save() {

std::string tmp;

graphic\_editor.save\_document();

std::cout << "save document" << std::endl;

}

void add() {

std::shared\_ptr<Figure> newElem = Factory::figure\_create(std::cin);

graphic\_editor.insert\_in\_document(newElem);

std::cout << "was added" << std::endl;

}

void remove() {

size\_t index;

std::cout << "enter index" << std::endl;

std::cin >> index;

graphic\_editor.delete\_in\_document(index);

std::cout << "was removed" << std::endl;

}

int main() {

std::string cmd;

while (1) {

std::cin >> cmd;

if (cmd == "menu") {

menu();

}

else if (cmd == "create\_doc") {

create();

}

else if (cmd == "load\_doc") {

load();

}

else if (cmd == "save\_doc") {

save();

}

else if (cmd == "exit") {

break;

}

else if (cmd == "add\_figure") {

add();

}

else if (cmd == "remove\_figure") {

remove();

}

else if (cmd == "print\_doc") {

graphic\_editor.print\_document();

}

else if (cmd == "undo") {

graphic\_editor.undo();

}

else {

std::cout << "unknown cmd" << std::endl;

}

}

return 0;

}

из файла **Point.h**:

#ifndef OOP7\_POINT\_H

#define OOP7\_POINT\_H

#include <iostream>

class Point {

public:

double x, y;

Point() = default;

Point (double f,double s) {

x = f;

y = s;

}

};

std::istream& operator >> (std::istream& is,Point& p ) {

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

}

std::ostream& operator << (std::ostream& os,const Point& p) {

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

}

#endif //OOP7\_POINT\_H

из файла **Rectangle.h**:

#ifndef OOP7\_RECTANGLE\_H

#define OOP7\_RECTANGLE\_H

#include <cmath>

#include "Point.h"

#include "Figure.h"

class Rectangle : public Figure {

public:

Point a1, a2, a3, a4;

Point center() const override{

double x, y;

x = (a1.x + a2.x + a3.x + a4.x) / 4;

y = (a1.y + a2.y + a3.y + a4.y) / 4;

Point p(x, y);

return p;

}

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

os << "rectangle" << std::endl;

os << a1 << std::endl;

os << a2 << std::endl;

os << a3 << std::endl;

os << a4 << std::endl;

}

void print\_file(std::ofstream& of) const override{

of << "rectangle" << std::endl;

of << a1 << std::endl;

of << a2 << std::endl;

of << a3 << std::endl;

of << a4 << std::endl;

}

double area() const override{

std::vector<Point> points = {a1,a2,a3,a4};

return gauss\_area(points);

}

Rectangle(std::istream& is) {

is >> a1 >> a2 >> a3 >> a4;

}

};

#endif //OOP7\_RECTANGLE\_H

из файла **Square.h**:

#ifndef OOP7\_SQUARE\_H

#define OOP7\_SQUARE\_H

#include <cmath>

#include "Point.h"

#include "Figure.h"

class Square : public Figure {

public:

Point a1, a2, a3, a4;

Point center() const override{

double x, y;

x = (a1.x + a2.x + a3.x + a4.x) / 4;

y = (a1.y + a2.y + a3.y + a4.y) / 4;

Point p(x, y);

return p;

}

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

os << "square" << std::endl;

os << a1 << std::endl;

os << a2 << std::endl;

os << a3 << std::endl;

os << a4 << std::endl;

}

void print\_file(std::ofstream &of) const override{

of << "square" << std::endl;

of << a1 << std::endl;

of << a2 << std::endl;

of << a3 << std::endl;

of << a4 << std::endl;

}

double area() const override{

std::vector<Point> points = {a1,a2,a3,a4};

return gauss\_area(points);

}

Square(std::istream &is) {

is >> a1 >> a2 >> a3 >> a4;

}

};

#endif //OOP7\_SQUARE\_H

из файла **Triangle.h**:

#ifndef OOP7\_TRIANGLE\_H

#define OOP7\_TRIANGLE\_H

#include <cmath>

#include <iostream>

#include "Point.h"

#include "Figure.h"

class Triangle : public Figure{

public:

Point a1, a2, a3;

Point center() const override{

double x,y;

x = (a1.x + a2.x + a3.x) / 3;

y = (a1.y + a2.y + a3.y) / 3;

Point p(x,y);

return p;

}

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

os << "triangle " << std::endl;

os << a1 << std::endl;

os << a2 << std::endl;

os << a3 << std::endl;

}

void print\_file(std::ofstream &of) const override{

of << "triangle " << std::endl;

of << a1 << std::endl;

of << a2 << std::endl;

of << a3 << std::endl;

}

double area() const override{

std::vector<Point> points = {a1,a2,a3};

return gauss\_area(points);

}

Triangle(std::istream& is) {

is >> a1 >> a2 >> a3;

}

};

#endif //OOP7\_TRIANGLE\_H

1. Выводы

Я научился на практике пользоваться шаблонами проектирования (фабрикой).

1. Список использованных источников.

1 std::shared\_ptr [Электронный ресурс].

URL: https://en.cppreference.com/w/cpp/memory/shared\_ptr

2 Templates [Электронный ресурс].

URL: https://www.cplusplus.com/doc/oldtutorial/templates/

3 Фабричный метод на C++ [Электронный ресурс].

URL: https://refactoring.guru/ru/design-patterns/factory-method/cpp/example

4 std::logic\_error [Электронный ресурс].

URL: https://en.cppreference.com/w/cpp/error/logic\_error

5 operator new, operator new[] [Электронный ресурс].

URL: https://en.cppreference.com/w/cpp/memory/new/operator\_new