**Федеральное государственное автономное образовательное учреждение высшего образования**

**"Московский Авиационный Институт (Национальный Исследовательский Университет"**

Институт “Информационные технологии и прикладная математика”

**Лабораторная работа № 8**

по курсу «ООП»

Тема: Асинхронное программирование

Студент: Николаева Елизавета

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата: 15.01.2021

Оценка:

Москва, 2021

1. Постановка задачи

Вариант 5

Фигуры: ромб, пятиугольник, шестиугольник.

Создать простейший “графический” редактор. Требования к функционалу редактора:

1) создание нового документа

2) импорт документа в файл

3) экспорт документа из файла

4) создание графического примитива

5) удаление графического примитива

6) отображение документа на экране

7) реализовать операцию undo, отменяющую последнее сделанное действие

Требования к реализации:

1) создание графических примитивов необходимо вынести в отдельный класс factory.

2) сделать упор на использование полиморфизма при работе с фигурами

3) взаимодействие с пользователем реализовать в функции main

2. Описание программы

Репозиторий: <https://github.com/esnikolaeva/oop/tree/main/oop_exercise_07>

Класс figure – абстрактный базовый класс для остальных фигур. Содержит в себе чисто виртуальные функции square() для вычисления площади, print() для печати фигуры, write\_to\_file() для записи в файл. Единственный атрибут – координаты центра фигуры.

Классы rhombus, pentagon, hexagon – классы наследники от figure, в которых описаны ромб, пятиугольник и шестиугольник соответственно.

Класс document описывает функционал для работы с “графическим” документом. Включает методы добавления фигуры в документ, удаления фигур, печати всех фигур, записи данных в файл, чтения всех данных из файла и отмены последнего действия.

Класс factory. В нем реализован шаблон factory для упрощения создания новых объектов.

В функции main описано взаимодействие с пользователем. Реализовано меню.

3. Тестирование

**Тест 01:**

6

Open document first

1

Enter document’s name:

doc1

Created new document

4

Enter figure id (1 – rhombus, 2 – pentagon, 3 – hexagon)

1

Enter coords of the center and lengths of diagonals

2 2 4 8

Figure was successfully added

4

Enter figure id (1 – rhombus, 2 – pentagon, 3 – hexagon)

2

Enter coords of the center and lengths of radius

1 1 10

Figure was successfully added

4

Enter figure id (1 – rhombus, 2 – pentagon, 3 – hexagon)

3

Enter coords of the center and length of side

-1 -3 5

Figure was successfully added

6

Rhombus { (0; 2), (2; 6), (4; 2), (2; -2) }

Square: 16

Center: (2; 2)

Pentagon { (11; 4.1), (1; 11), (-8.5; 4.1), (-4.9; -7.1), (6.9; -7.1) }

Square: 2.4e+002

Center: (1; 1)

Hexagon { (4; -3), (1.5; 1.3), (-3.5; 1.3), (-6; -3), (-5.5; -7.3), (1.5; -7.3)}

Square: 65

Center: (-1; -3)

5

Enter

Enter id of the figure

1 Figure was successfully removed

6 Rhombus {(0; 2), (

2; 6), (

4; 2), (

2; -

2)}

Square: 16

Center: (2; 2)

Hexagon {(4; -3), (1.5; 1.3), (-3.5; 1.3), (-6; -3), (-3.5; -7.3), (1.5;

-7.3)}

Square: 65

Center: (-1; -3)

7 Done

6 Rhombus {(0; 2), (

2; 6), (

4; 2), (

2; -

2)}

Square: 16

Center: (2; 2)

Pentagon {(11; 4.1), (1; 11), (-8.5; 4.1), (-4.9; -7.1), (6.9; -7.1)}

Square: 2.4e+002

Center: (1; 1)

Hexagon {(4; -3), (1.5; 1.3), (-3.5; 1.3), (-6; -3), (-3.5; -7.3), (1.5;

-7.3)}

Square: 65

Center: (-1; -3)

5 Enter id of the figure

1000

Invalid position

5 Enter id of the figure

-10

Invalid position

0

Process finished with exit code 0

**Тест 02:**

1

Enter document's name:

doc2

Created new document

4 Enter figure id (

1

-

rhombus, 2

-

pentagon, 3

-

hexagon)

1 Enter coords of the center and lengths of diagonals

0 0 5 10

Figure was successfully added

6 Rhombus {(-2.5; 0), (

0; 5), (

2.5; 0), (

0; -

5)}

Square: 25

Center: (0; 0)

2 Successfully saved

4 Enter figure id (

1

-

rhombus, 2

-

pentagon, 3

-

hexagon)

2 Enter coords of the center and length of radius

10 10 10

Figure was successfully added

6 Rhombus {(-2.5; 0), (

0; 5), (

2.5; 0), (

0; -

5)}

Square: 25

Center: (0; 0)

Pentagon {(20; 13), (10; 20), (0.49; 13), (4.1; 1.9), (16; 1.9)}

Square: 2.4e+002

Center: (10; 10)

3 Enter file's name:

doc123

No such file

3 Enter file's name:

doc2

File was successfully loaded

6 Rhombus {(-2.5; 0), (

0; 5), (

2.5; 0), (

0; -

5)}

Square: 25

Center: (0; 0)

0

Process finished with exit code 0

4. Листинг

**figure.h**

#include <cmath>

class figure {

public:

figure() = default;

figure(std::pair<double, double> &center\_) : center(center\_) {}

virtual double square() = 0;

virtual void print() = 0;

virtual void write\_to\_file(std::ofstream &out) = 0;

std::pair<double, double> get\_center() {

return center;

}

protected:

std::pair<double, double> center;

};

**rhombus.h**

#include "figure.h"

class rhombus : public figure {

public:

rhombus() = default;

rhombus(std::pair<double, double> &center, double d1, double d2) :

figure(center), diag1(d1), diag2(d2) {}

double square() override {

return diag1 \* diag2 \* 0.5;

}

void print() override {

std::cout << \*this;

}

void write\_to\_file(std::ofstream &out) override {

int id = 1;

out.write((char \*) &id, sizeof(int));

out.write((char \*) &center.first, sizeof(double));

out.write((char \*) &center.second, sizeof(double));

out.write((char \*) &diag1, sizeof(double));

out.write((char \*) &diag2, sizeof(double));

}

friend std::ostream &operator<<(std::ostream &out, rhombus &r);

private:

double diag1 = 0;

double diag2 = 0;

};

std::ostream &operator<<(std::ostream &out, rhombus &r) {

out << "Rhombus {(" << r.center.first - r.diag1 \* 0.5 << "; " <<

r.center.second << "), (";

out << r.center.first << "; " << r.center.second + r.diag2 \* 0.5 << "), (";

out << r.center.first + r.diag1 \* 0.5 << "; " << r.center.second << "), (";

out << r.center.first << "; " << r.center.second - r.diag2 \* 0.5 << ")}";

return out;

}

**pentagon.h**

#include "figure.h"

class pentagon : public figure {

public:

pentagon() = default;

pentagon(std::pair<double, double> &center, double rad) : figure(center),

radius(rad) {}

double square() override {

double pi = acos(-1);

double side = radius \* cos(13 \* pi / 10) - radius \* cos(17 \* pi / 10);

return sqrt(25 + 10 \* sqrt(5)) \* pow(side, 2) \* 0.25;

}

void print() override {

std::cout << \*this;

}

void write\_to\_file(std::ofstream &out) override {

int id = 2;

out.write((char \*) &id, sizeof(int));

out.write((char \*) &center.first, sizeof(double));

out.write((char \*) &center.second, sizeof(double));

out.write((char \*) &radius, sizeof(double));

}

friend std::ostream &operator<<(std::ostream &out, pentagon &p);

private:

double radius = 0;

};

std::ostream &operator<<(std::ostream &out, pentagon &p) {

std::cout << "Pentagon {";

double pi = acos(-1);

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

double angle = 2 \* pi \* i / 5;

std::cout.precision(2);

std::cout << "(" << p.center.first + p.radius \* cos(angle + pi / 10) << ";

"

<< p.center.second + p.radius \* sin(angle + pi / 10) << ")";

if (i != 4) {

std::cout << ", ";

}

}

std::cout << "}";

return out;

}

**hexagon.h**

#include "figure.h"

class hexagon : public figure {

public:

hexagon() = default;

hexagon(std::pair<double, double> &center, double rad) : figure(center),

radius(rad) {}

double square() override {

return pow(radius, 2) \* 3 \* sqrt(3) \* 0.5;

}

void print() override {

std::cout << \*this;

}

void write\_to\_file(std::ofstream &out) override {

int id = 3;

out.write((char \*) &id, sizeof(int));

out.write((char \*) &center.first, sizeof(double));

out.write((char \*) &center.second, sizeof(double));

out.write((char \*) &radius, sizeof(double));

}

friend std::ostream &operator<<(std::ostream &out, hexagon &h);

private:

double radius = 0;

};

std::ostream &operator<<(std::ostream &out, hexagon &h) {

std::cout << "Hexagon {";

double pi = acos(-1);

for (int i = 0; i < 6; ++i) {

double angle = pi \* i / 3;

std::cout.precision(2);

std::cout << "(" << h.center.first + h.radius \* cos(angle) << "; "

<< h.center.second + h.radius \* sin(angle) << ")";

if (i != 5) {

std::cout << ", ";

}

}

std::cout << "}";

return out;

}

**factory.h**

#include "rhombus.h"

#include "pentagon.h"

#include "hexagon.h"

enum class figure\_type {

rhombus = 1,

pentagon = 2,

hexagon = 3

};

struct factory {

static std::shared\_ptr<figure> create(figure\_type t) {

switch (t) {

case figure\_type::rhombus: {

std::pair<double, double> center;

double d1, d2;

std::cin >> center.first >> center.second >> d1 >> d2;

return std::make\_shared<rhombus>(center, d1, d2);

}

case figure\_type::pentagon: {

std::pair<double, double> center;

double r;

std::cin >> center.first >> center.second >> r;

return std::make\_shared<pentagon>(center, r);

}

case figure\_type::hexagon: {

std::pair<double, double> center;

double r;

std::cin >> center.first >> center.second >> r;

return std::make\_shared<hexagon>(center, r);

}

default:

throw std::logic\_error("Wrong figure id");

}

}

static std::shared\_ptr<figure> read\_from\_file(figure\_type t, std::ifstream &in)

{

switch (t) {

case figure\_type::rhombus: {

std::pair<double, double> center;

double d1, d2;

in.read((char \*) &center.first, sizeof(double));

in.read((char \*) &center.second, sizeof(double));

in.read((char \*) &d1, sizeof(double));

in.read((char \*) &d2, sizeof(double));

return std::make\_shared<rhombus>(center, d1, d2);

}

case figure\_type::pentagon: {

std::pair<double, double> center;

double r;

in.read((char \*) &center.first, sizeof(double));

in.read((char \*) &center.second, sizeof(double));

in.read((char \*) &r, sizeof(double));

return std::make\_shared<pentagon>(center, r);

}

case figure\_type::hexagon: {

std::pair<double, double> center;

double r;

in.read((char \*) &center.first, sizeof(double));

in.read((char \*) &center.second, sizeof(double));

in.read((char \*) &r, sizeof(double));

return std::make\_shared<hexagon>(center, r);

}

default:

throw std::logic\_error("Wrong figure id");

}

}

};

**document.h**

#include <stack>

#include <fstream>

#include "factory.h"

class document {

private:

struct memento {

std::vector<std::shared\_ptr<figure>> state;

memento() = default;

memento(std::vector<std::shared\_ptr<figure>> &other) : state(other) {}

};

struct originator {

std::stack<memento> mementos;

void create\_memento(std::vector<std::shared\_ptr<figure>> &state) {

mementos.emplace(state);

}

std::vector<std::shared\_ptr<figure>> restore() {

if (!mementos.empty()) {

std::vector<std::shared\_ptr<figure>> res = mementos.top().state;

mementos.pop();

return res;

}

throw std::logic\_error("Can't undo");

}

};

std::string name;

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

originator origin;

public:

document(std::string &name\_) : name(name\_) {}

void add(const std::shared\_ptr<figure> &figure) {

origin.create\_memento(buffer);

buffer.push\_back(figure);

}

void remove(int id) {

if (id >= 0 && id < buffer.size()) {

origin.create\_memento(buffer);

buffer.erase(buffer.begin() + id);

} else {

throw std::logic\_error("Invalid position");

}

}

void undo() {

buffer = origin.restore();

}

void print() {

for (auto &f : buffer) {

f->print();

std::cout << std::endl;

std::cout << "Square: " << f->square() << std::endl;

auto center = f->get\_center();

std::cout << "Center: (" << center.first << "; " << center.second <<

")" << std::endl << std::endl;

}

}

void save() {

std::ofstream out;

out.open(name, std::ios::out | std::ios::binary | std::ios::trunc);

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

throw std::logic\_error("Can't open file");

} else {

int size = buffer.size();

out.write((char \*) &size, sizeof(int));

for (auto &f : buffer) {

f->write\_to\_file(out);

}

out.close();

}

}

void open(std::ifstream &in) {

int size;

in.read((char \*) &size, sizeof(int));

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

int type;

in.read((char \*) &type, sizeof(int));

buffer.push\_back(factory::read\_from\_file((figure\_type) type, in));

}

}

};

**main.cpp**

#include <iostream>

#include <memory>

#include <vector>

#include "document.h"

void print\_menu() {

std::cout << "1. Create new document" << std::endl;

std::cout << "2. Save document" << std::endl;

std::cout << "3. Open document" << std::endl;

std::cout << "4. Add figure" << std::endl;

std::cout << "5. Remove figure" << std::endl;

std::cout << "6. Print figures" << std::endl;

std::cout << "7. Undo" << std::endl;

std::cout << "0. Exit" << std::endl;

std::cout << std::endl;

}

int main() {

print\_menu();

std::shared\_ptr<document> doc;

int cmd;

while (true) {

std::cin >> cmd;

if (cmd == 1) {

std::string name;

std::cout << "Enter document's name:" << std::endl;

std::cin >> name;

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

std::cout << "Created new document" << std::endl;

} else if (cmd == 2) {

if (!doc) {

std::cout << "Open document first" << std::endl;

} else {

try {

doc->save();

std::cout << "Successfully saved" << std::endl;

}

catch (std::exception &ex) {

std::cout << ex.what() << std::endl;

}

}

} else if (cmd == 3) {

std::string file\_name;

std::cout << "Enter file's name: " << std::endl;

std::cin >> file\_name;

std::ifstream in;

in.open(file\_name, std::ios::in | std::ios::binary);

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

std::cout << "No such file" << std::endl;

} else {

doc = std::make\_shared<document>(file\_name);

try {

doc->open(in);

std::cout << "File was successfully loaded" << std::endl;

}

catch (std::exception &ex) {

std::cout << ex.what() << std::endl;

}

in.close();

}

} else if (cmd == 4) {

if (!doc) {

std::cout << "Open document first" << std::endl;

} else {

std::cout << "Enter figure id (1 - rhombus, 2 - pentagon, 3 -

hexagon)" << std::endl;

int type;

std::cin >> type;

if (type == 1) {

std::cout << "Enter coords of the center and lengths of

diagonals" << std::endl;

} else if (type == 2) {

std::cout << "Enter coords of the center and length of radius"

<< std::endl;

} else if (type == 3) {

std::cout << "Enter coords of the center and length of side" <<

std::endl;

}

std::shared\_ptr<figure> fig = factory::create((figure\_type) type);

doc->add(fig);

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

}

} else if (cmd == 5) {

if (!doc) {

std::cout << "Open document first" << std::endl;

} else {

int id;

std::cout << "Enter id of the figure" << std::endl;

std::cin >> id;

try {

doc->remove(id);

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

}

catch (std::exception &ex) {

std::cout << ex.what() << std::endl;

}

}

} else if (cmd == 6) {

if (!doc) {

std::cout << "Open document first" << std::endl;

} else {

doc->print();

}

} else if (cmd == 7) {

if (!doc) {

std::cout << "Open document first" << std::endl;

} else {

try {

doc->undo();

std::cout << "Done" << std::endl;

}

catch (std::exception &ex) {

std::cout << ex.what() << std::endl;

}

}

} else if (cmd == 0) {

break;

} else {

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

}

}

};

5. Список литературы

Справочник по языку [Электронный ресурс]. URL: <https://ravesli.com/uroki-cpp/> (дата обращения 15.01.2021).

Справочник по языку [Электронный ресурс]. URL:

<https://metanit.com/cpp/tutorial/> (дата обращения 15.01.2021).

Шаблон memento [Электронный ресурс]. URL:

http://cpp-reference.ru/patterns/behavioral-patterns/memento/ (дата обращения

17.12.2020).

Шаблон factory [Электронный ресурс]. URL:

http://cpp-reference.ru/patterns/creational-patterns/factory-method/ (дата обращения

17.12.2020).