**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: Тимофеев Алексей Владимирович

Группа: 80-207

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;

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | 5-угольник | 6-угольник | 8-угольник |

1. **Описание программы**

В программе реализован класс Figure и его наследники (фигуры вращения) Pentagone, Hexagon, Octagon. Данные конструкции были взяты из 3 лабораторной работы и адаптированы под нужды 7 работы, а именно: был переписан конструктор и определение названия фигуры.Также реализован класс Initiate, в котором инициализируются фигуры. Класс Backup хранит в себе стек динамических массивов, для того чтобы можно было вернуть программу на шаг назад. Save() - добавляет вектор на вершину стека, а метод Undo() - выдает элемент, который лежит на вершине стека и удаляет его с вершины.В программе перед каждым добавлением или удалением вектор фигур подается методу Save(), тем самым обеспечивая сохранения состояние программы.Также реализовано удаление фигуры по индексу.

1. **Набор  тестов**

координаты центральной точки и длина радиуса 10 10 10

координаты центральной точки и длина радиуса 8 9 10

координаты центральной точки и длина радиуса 6 6 6 -100 -50 20

координаты центральной точки и длина радиуса -100 -50 20

1. **Результаты выполнения тестов**

dude@DESKTOP-9IO9OQQ:/mnt/d/Ycheba2kurs/OOp/OOPlab/laba7$ ./lab7

Enter:

1 - creating a new document

2 - import document from file

3 - export document to file

4 - add pentagon

5 - add hexagon

6 - add octagon

7 - delete shape by index

8 - print shapes to the screen

9 - undo the last action

0 - end the program

Enter the command: 5

Enter the coordinates of the center point and the length of the radius of the circumscribed circle, separated by a space: 10 10 10

Enter the command: 4

Enter the coordinates of the center point and the length of the radius of the circumscribed circle, separated by a space: 8 9 10

Enter the command: 6

Enter the coordinates of the center point and the length of the radius of the circumscribed circle, separated by a space: 6 6 6

Enter the command: 8

Hexagon

(0.47587,6.95189) (18.1418,15.8061) (4.0154,1.98847) (13.2578,19.4545) (9.77903,0.0024416) (7.16309,19.5892)

Pentagon

(-1.67251,11.5382) (16.7115,4.08978) (0.820149,15.9606) (13.178,0.444956) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Enter the command: 5

Enter the coordinates of the center point and the length of the radius of the circumscribed circle, separated by a space: -100 -50 20

Enter the command: 8

Hexagon

(0.47587,6.95189) (18.1418,15.8061) (4.0154,1.98847) (13.2578,19.4545) (9.77903,0.0024416) (7.16309,19.5892)

Pentagon

(-1.67251,11.5382) (16.7115,4.08978) (0.820149,15.9606) (13.178,0.444956) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Hexagon

(-119.048,-56.0962) (-83.7164,-38.3878) (-111.969,-66.0231) (-93.4844,-31.0911) (-100.442,-69.9951) (-105.674,-30.8217)

Enter the command: 9

Enter the command: 8

Hexagon

(0.47587,6.95189) (18.1418,15.8061) (4.0154,1.98847) (13.2578,19.4545) (9.77903,0.0024416) (7.16309,19.5892)

Pentagon

(-1.67251,11.5382) (16.7115,4.08978) (0.820149,15.9606) (13.178,0.444956) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Enter the command: 3

Enter file:save

Enter the command: 7

Enter index:1

Enter the command: 7

Enter index:2

Enter the command: 8

Pentagon

(-1.67251,11.5382) (16.7115,4.08978) (0.820149,15.9606) (13.178,0.444956) (5.16309,18.5892)

Enter the command: 2

Enter file: save

Enter the command: 8

Hexagon

(-9.04826,3.90379) (8.61768,12.758) (-5.50873,-1.05963) (3.73368,16.4063) (0.254904,-3.04566) (-2.36104,16.5411)

Pentagon

(-11.345,14.0765) (7.03897,6.62802) (-8.85236,18.4988) (3.50545,2.98319) (-4.50942,21.1274)

Octagon

(12.3039,16.2108) (6.46349,16.4694) (3.1754,11.6356) (5.56117,6.29851) (11.3558,5.52485) (15.0582,10.0491) (13.1534,15.5762) (7.44979,16.8589)

Enter the command: 0

1. **Листинг программы**

**figure.hpp**

#pragma once

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <memory>

struct Pair {

double x;

double y;

Pair() {

x = 0;

y = 0;

}

Pair(double a, double b) {

x = a;

y = b;

}

};

class Figure {

protected:

std::vector<Pair> points;

public:

Figure() {}

Figure(double x1, double y1, int c) {

points.emplace\_back(Pair(x1,y1));

}

virtual std::string Name() {

return "Point";

}

virtual int Get() {

return 0;

}

virtual std::vector<Pair> Coord() {

return points;

}

};

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

os << '(' << p.x << ',' << p.y << ')';

return os;

}

std::ostream& operator<<(std::ostream &os, std::vector<Pair> p) {

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

os << " " << p[i];

}

return os;

}

std::ostream& operator<<(std::ostream &os, std::vector<std::shared\_ptr<Figure>> p) {

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

os << p[i]->Name() << std::endl;

os << p[i]->Coord() << std::endl;

}

return os;

}

**hexagon.hpp**

#pragma once

#include <cmath>

#include "figure.hpp"

class Hexagon: public Figure {

private:

int length;

public:

Hexagon() : Figure() {length = 0;}

Hexagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 6; i++) {// расчет точек

angle = 360 / 6;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Hexagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

**octagone.hpp**

#pragma once

#include <cmath>

#include "figure.hpp"

class Octagon: public Figure {

private:

int length;

public:

Octagon() : Figure() { length = 0; }

Octagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 8; i++) {// расчет точек

angle = 360 / 8;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Octagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

**pentagone.hpp**

#pragma once

#include <cmath>

#include "figure.hpp"

class Pentagon: public Figure {

private:

int length;

public:

Pentagon() : Figure() { length = 0; }

Pentagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 5; i++) {// расчет точек

angle = 360 / 5;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Pentagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

**main.cpp**

#include <stack>

#include <execution>

#include <iostream>

#include <algorithm>

#include <memory>

#include "pentagon.hpp"

#include "hexagon.hpp"

#include "octagon.hpp"

#include "figure.hpp"

struct Memento {

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

Memento() {}

Memento(const std::vector<std::shared\_ptr<Figure>> &other) : state({other}) {}

~Memento() {}

};

struct Backup {

std::stack<Memento> temp;

void Save(std::vector<std::shared\_ptr<Figure>> fignext) {

temp.emplace(fignext);

}

std::vector<std::shared\_ptr<Figure>> Undo() {

if (!temp.empty()) {

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

temp.pop();

return res;

} else {

throw std::logic\_error("impossible to return, vector is empty");

}

}

};

template<class A, class B, class C>

class Initiate {

public:

Initiate() {}

~Initiate() {}

std::shared\_ptr<Figure> Pentagon(double x1, double y1, int R) {

return std::make\_shared<A>(x1, y1, R);

}

std::shared\_ptr<Figure> Hexagon(double x1, double y1, int R) {

return std::make\_shared<B>(x1, y1, R);

}

std::shared\_ptr<Figure> Octagon(double x1, double y1, int R) {

return std::make\_shared<C>(x1, y1, R);

}

};

void Print(){

std::cout << "Enter the coordinates of the center point and the length of the radius of the circumscribed circle, separated by a space: ";

}

int main() {

Backup doo;

double x1, y1;

int R, m;

std::vector<std::shared\_ptr<Figure>> fig;

Initiate<Pentagon, Hexagon, Octagon> addfigure;

std::cout << "Enter:\n 1 - creating a new document\n 2 - import document from file\n 3 - export document to file\n";

std::cout << " 4 - add pentagon\n 5 - add hexagon\n 6 - add octagon\n";

std::cout << " 7 - delete shape by index \n 8 - print shapes to the screen \n 9 - undo the last action \n 0 - end the program\n";

FILE \* f;

std::cout << "Enter the command: ";

while (std::cin >> m && m < 10 && m > 0) {

switch (m) {

case 1: {//создание чистого документа(очистка)

doo.Save(fig);

fig.clear();

break;

}

case 2: { //импорт документа из файла

doo.Save(fig);

fig.clear();

int a;

std::string name;

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

std::cin >> name;

f = fopen(name.c\_str(), "r");

if (f == NULL) {

std::cout << "File open error";

break;

}

while (!feof(f)) {

fread(&a, sizeof(int), 1, f);

fread(&x1, sizeof(double), 1, f);

fread(&y1, sizeof(double), 1, f);

fread(&R, sizeof(int), 1, f);

switch (a) {

case 1: {

fig.push\_back(addfigure.Pentagon(x1, y1, R));

break;

}

case 2: {

fig.push\_back(addfigure.Hexagon(x1, y1, R));

break;

}

case 3: {

fig.push\_back(addfigure.Octagon(x1, y1, R));

break;

}

default:

break;

}

a = -1;

}

fclose(f);

break;

}

case 3: {//экспорт документа в файл

std::string name;

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

std::cin >> name;

f = fopen(name.c\_str(), "w");

if (f == NULL) {

std::cout << "File open error"<< std::endl;

break;

}

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

int a;

if (fig[i]->Name() == "Pentagon" ) a = 1;

if (fig[i]->Name() == "Hexagon" ) a = 2;

if (fig[i]->Name() == "Octagon" ) a = 3;

fwrite(&a, sizeof(int), 1, f);

std::vector<Pair> temp = fig[i]->Coord();

fwrite(&temp[0].x, sizeof(double), 1, f);

fwrite(&temp[0].y, sizeof(double), 1, f);

int b = fig[i]->Get();

fwrite(&b, sizeof(int), 1, f);

}

fclose(f);

break;

}

case 4: {

Print();

std::cin >> x1 >> y1 >> R;

if (R < 0) {

std::cout << "The entered values are incorrect! Length cannot be negative."<< std::endl;

break;

}

doo.Save(fig);

fig.push\_back(addfigure.Pentagon(x1, y1, R));

break;

}

case 5: {

Print();

std::cin >> x1 >> y1 >> R;

if (R < 0) {

std::cout << "The entered values are incorrect! Length cannot be negative."<< std::endl;

break;

}

doo.Save(fig);

fig.push\_back(addfigure.Hexagon(x1, y1, R));

break;

}

case 6: {

Print();

std::cin >> x1 >> y1 >> R;

if (R < 0) {

std::cout << "The entered values are incorrect! Length cannot be negative." << std::endl;

break;

}

doo.Save(fig);

fig.push\_back(addfigure.Octagon(x1, y1, R));

break;

}

case 7: {

int ind;

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

std::cin >> ind;

doo.Save(fig);

if (ind <= 0 || ind >= fig.size() + 1) {

std::cout << "The entered values are not correct" << std::endl;

break;

}

ind--;

fig.erase(fig.begin() + ind);

break;

}

case 8: {

std::cout << fig;

break;

}

case 9: {

fig = doo.Undo();

std::cout << std::endl;

break;

}

default:

break;

}

std::cout << "Enter the command: ";

}

}

1. **Вывод**

В данной лабораторной работе я научился создавать простейший “графический” векторный редактор. Также я научился делать возвращение в предыдущее состояние программы при добавлении и удалении фигур, попрактиковалась в работе с библиотекой STL, очень удобно что в ней есть умные указатели, которые сами собирают и удаляют за собой мусор.

Список литературы

1. cppreference.com [Электронный ресурс]. URL: https://en.cppreference.com/w/cpp/container/stack (дата обращения: 10.12.20)
2. Презентация “Проектируем структуру классов. ЛЕКЦИЯ 12” - Дзюба Д.В. (дата обращения: 10.12.20).