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: Будникова Валерия Павловна
Группа: 80-207
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

ВАРИАНТ 31

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.)

Фигуры по варианту: треугольник, 6-угольник, 7-угольник.

2. Описание программы

В программе реализован класс Figure и его наследники - Triangle, Hexagon, Octagon. В каждом классе есть конструкторы(определения координат вершин для каждой фигуры), также метод получения вектора координат, получения сторону фигуры и ее имени. Все эти методы реализованы аналогично 3-й лабораторной работе. Также реализован класс Factory, в котором есть методы создания фигур. Класс Action хранит в себе стек векторов, для возможности сделать шаг назад после удаления или добавления фигуры. Метод этого класса - 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 “до” совершения действия. Перед записью. В моей реализации состояние вектора запоминается также перед импортом документа из файла. При создании нового документа вектор сохраняется методом Save(), а происходит очищения вектора. Также реализовано удаление фигуры по индексу.

3. Руководство по использованию программы

Взаимодействие с пользователем происходит с помощью меню:

Введите:

1 - создание нового документа
2 - импорт документа из файла
3 - экспорт документа в файл
4 - добавить треугольник
5 - добавить шестиугольник

6 - добавить восьмиугольник
7 - удалить фигуру по индексу
8 - печать на экран фигур
9 - отменить последнее действие
10 - завершить программу

4. Набор тестов и результаты работы программы

Описание ввода: При вводе неправильных значений пользователю выдается соответствующее сообщение на экран.

```
Lera:lab7 valeriabudnikova$ make run
./lab7
Введите:
 1  - создание нового документа
 2  - импорт документа из файла
 3  - экспорт документа в файл
 4  - добавить треугольник
 5  - добавить шестиугольник
 6  - добавить восьмиугольник
 7  - удалить фигуру по индексу
 8  - печать на экран фигур
 9  - отменить последнее действие
 10 - завершить программу
Введите команду:2
Введите файла:1.txt
Ошибка открытия файла
Введите команду:4
Введите координаты верхней точки и длину стороны через пробел: 1 2 3

Введите команду:5
Введите координаты верхней точки и длину стороны через пробел: 1 5 6

Введите команду:6
Введите координаты верхней точки и длину стороны через пробел: 3 4 5

Введите команду:8
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)
Octagon
 (3,4) (7.6194,2.08658) (-1.6194,2.08658) (9.53281,-2.53281) (-3.53281,-2.53281)
(7.6194,-5.2388) (-1.6194,-5.2388) (3,-9.06563)

Введите команду:9

Введите команду:8
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)

Введите команду:3
```

```
Введите файла:1.txt

Введите команду:4
Введите координаты верхней точки и длину стороны через пробел: 1 2 3

Введите команду:6
Введите координаты верхней точки и длину стороны через пробел: 2 3 4

Введите команду:8
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Octagon
 (2,3) (5.69552,1.46927) (-1.69552,1.46927) (7.22625,-2.22625) (-3.22625,-2.22625)
(5.69552,-4.39104) (-1.69552,-4.39104) (2,-7.4525)

Введите команду:7
Введите индекс:1

Введите команду:8
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Octagon
 (2,3) (5.69552,1.46927) (-1.69552,1.46927) (7.22625,-2.22625) (-3.22625,-2.22625)
(5.69552,-4.39104) (-1.69552,-4.39104) (2,-7.4525)

Введите команду:9

Введите команду:8
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Octagon
 (2,3) (5.69552,1.46927) (-1.69552,1.46927) (7.22625,-2.22625) (-3.22625,-2.22625)
(5.69552,-4.39104) (-1.69552,-4.39104) (2,-7.4525)

Введите команду:2
Введите файла:1.txt

Введите команду:8
Triangle
 (1,2) (-0.5,-0.598076) (2.5,-0.598076)
Hexagon
 (1,5) (-4.19615,2) (6.19615,2) (-4.19615,-4) (6.19615,-4) (1,-7)
Введите команду:10
Lera:lab7 valeriabudnikova$
```

6. Листинг программы

main.cpp

```
#include <iostream>
#include <algorithm>
#include <stack>
#include <execution>
#include "figure.hpp"
#include "triangle.hpp"
#include "hexagon.hpp"
#include "octagon.hpp"

template<class A, class B, class C>
class Factory {
    public:
    Factory() {}
    ~Factory() {}
    std::shared_ptr<Figure> Triangle(double x1, double y1, int side) {
        return std::make_shared<A>(x1, y1, side);
    }
    std::shared_ptr<Figure> Hexagon(double x1, double y1, int side) {
        return std::make_shared<B>(x1, y1, side);
    }
    std::shared_ptr<Figure> Octagon(double x1, double y1, int side) {
        return std::make_shared<C>(x1, y1, side);
    }
};



void Menu(){
     std::cout << "Введите:\n 1  - создание нового документа\n 2  - импорт документа из файла\n 3  - экспорт документа в файл\n";
     std::cout << " 4  - добавить треугольник\n 5  - добавить шестиугольник\n 6  - добавить восьмиугольник\n";
     std::cout << " 7  - удалить фигуру по индексу\n 8  - печать на экран фигур \n 9 - отменить последнее действие\n 10 - завершить программу\n";
}

void Print(){
    std::cout << "Введите координаты верхней точки и длину стороны через пробел: ";
}

struct Memento {
    std::vector<std::shared_ptr<Figure>> state;
    Memento() {}
    Memento(const std::vector<std::shared_ptr<Figure>> &other) : state({other}) {}
    ~Memento() {}
};

struct Action {
    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("err");
        }
    }
};

int main() {
    Action doo;
    double x1, y1;
    int side, m;
    std::vector<std::shared_ptr<Figure>> fig;
    Factory<Triangle, Hexagon, Octagon> addfigure;
    Menu();
    FILE * f;
    std::cout << "Введите команду:";
    while (std::cin >> m && m < 10 && m > 0) {
        switch (m) {
            case 1: {
                doo.Save(fig);
                fig.clear();
                //создание нового документа(очистка)
                break;
            }
            case 2: {
                //импорт документа из файла
                doo.Save(fig);
                fig.clear();
                int a;
                std::string name;
                std::cout << "Введите файла:";
                std::cin >> name;
                f = fopen(name.c_str(), "r");
                if (f == NULL) {
                    std::cout << "Ошибка открытия файла";
                    break;
                }
                bool fl = true;
                while (!feof(f)) {
                    fread(&a, sizeof(int), 1, f);
                    fread(&x1, sizeof(double), 1, f);
                    fread(&y1, sizeof(double), 1, f);
                    fread(&side, sizeof(int), 1, f);
                    switch (a) {
                    case 1: {
                        fig.push_back(addfigure.Triangle(x1, y1, side));
                        break;
                    }
                    case 2: {
                        fig.push_back(addfigure.Hexagon(x1, y1, side));
```

```
                    break;
                }
                case 3: {
                    fig.push_back(addfigure.Octagon(x1, y1, side));
                    break;
                }
                default:
                    break;
                }
                a = -1;
            }
            fclose(f);
            break;
        }
        case 3: {
            //экспорт документа в файл
            std::string name;
            std::cout << "Введите файла:";
            std::cin >> name;
            f = fopen(name.c_str(), "w");
            if (f == NULL) {
                std::cout << "Ошибка открытия файла"<< std::endl;
                break;
            }
            for (int i = 0; i < fig.size(); ++i) {
                int a;
                if (fig[i]->Name() == "Triangle" ) 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 >> side;
            if (side < 0) {
                 std::cout << "Введенные значения не верные! Длина не может быть
отрицательной."<< std::endl;
                break;
            }
            doo.Save(fig);
            fig.push_back(addfigure.Triangle(x1, y1, side));
            break;
        }
        case 5: {
            Print();
            std::cin >> x1 >> y1 >> side;
            if (side < 0) {
```

```
                    std::cout << "Введенные значения не верные! Длина не может быть отрицательной."<< std::endl;
                    break;
                }
                doo.Save(fig);
                fig.push_back(addfigure.Hexagon(x1, y1, side));
                break;
            }
            case 6: {
                Print();
                std::cin >> x1 >> y1 >> side;
                if (side < 0) {
                    std::cout << "Введенные значения не верные! Длина не может быть отрицательной." << std::endl;
                    break;
                }

                doo.Save(fig);
                fig.push_back(addfigure.Octagon(x1, y1, side));
                break;
            }
            case 7: {
                int ind;
                std::cout << "Введите индекс:";
                std::cin >> ind;
                doo.Save(fig);
                if (ind <= 0 || ind >= fig.size() + 1) {
                    std::cout << "Введенные значения не верные" << 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 << "Введите команду:";
    }
}
```

figure.hpp

```
#pragma once
#include <iostream>
#include <vector>
```

```
#include <string>

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;
}
```

triangle.hpp

```
#pragma once
#include <cmath>
#include "figure.hpp"
```

```
class Triangle: public Figure {
    private:
        int side;
    public:
        Triangle() : Figure() { side = 0; }

        Triangle(double x1, double y1, int c) {
            side = c;
            points.emplace_back(Pair(x1, y1));
            points.emplace_back(Pair(x1 - (double)side / 2, y1 - (double)side *
sqrt(3) / 2));
            points.emplace_back(Pair(x1 + (double)side / 2, y1 - (double)side *
sqrt(3) / 2));
        }

        std::string Name() override {
            return "Triangle";
        }

        int Get() override {
            return side;
        }

        std::vector<Pair> Coord() override {
            return points;
        }
};
```

hexagon.hpp

```
#pragma once

#include <cmath>

#include "figure.hpp"


class Hexagon: public Figure {

    private:

        int side;

    public:

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

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

            points.push_back(Pair(x1, y1));

            points.push_back(Pair(x1 - (double)c * sqrt(3) / 2, y1 - (double) c /
2));

            points.push_back(Pair(x1 + (double)c * sqrt(3) / 2, y1 - (double) c /
2));

            points.push_back(Pair(x1 - (double)c * sqrt(3) / 2, y1 - (double) c / 2
- c));

            points.push_back(Pair(x1 + (double)c * sqrt(3) / 2, y1 - (double) c / 2
- c));

            points.push_back(Pair(x1, y1 - 2 * c));

            side = c;

        }

            std::string Name() override {
```

```
            return "Hexagon";
        }
        int Get() override {
            return side;
        }
        std::vector<Pair> Coord() override {
            return points;
        }
};
```

octagon.hpp

```
#pragma once
#include <cmath>
#include "figure.hpp"

class Octagon: public Figure {
    private:
        int side;
    public:
        Octagon() : Figure() { side = 0; }
        Octagon(double x1, double y1, int c) {
            points.push_back(Pair(x1, y1));
            points.push_back(Pair(x1 + (double)c * cos(M_PI / 8), y1 -  (double)c *
sin(M_PI / 8)));
            points.push_back(Pair(x1 - (double)c * cos(M_PI / 8), y1 -  (double)c *
sin(M_PI / 8)));
            points.push_back(Pair(x1 + (double)c * sqrt(1 / sqrt(2) + 1), y1 -
(double)c * sqrt(1 / sqrt(2) + 1)));
            points.push_back(Pair(x1 - (double)c * sqrt(1 / sqrt(2) + 1), y1 -
(double)c * sqrt(1 / sqrt(2) + 1)));
            points.push_back(Pair(x1 + (double)c * cos(M_PI / 8), y1 - 2 *
(double)c * cos(M_PI / 8)));
            points.push_back(Pair(x1 - (double)c * cos(M_PI / 8), y1 - 2 *
(double)c * cos(M_PI / 8)));
            points.push_back(Pair(x1, y1 - 2 * (double)c * sqrt(1 / sqrt(2) + 1)));
            side = c;
        }
        std::string Name() override {
            return "Octagon";
        }

        int Get() override {
            return side;
        }
```

```
        std::vector<Pair> Coord() override {
            return points;
        }
};
```

makefile

```
CC=g++
CFLAGS=-std=c++17
OUTPUT=lab7

all:
	$(CC) $(CFLAGS) main.cpp -o $(OUTPUT)
run:
	./$(OUTPUT)
```

7. Выводы

В данной лабораторной работе я реализовала простейший “графический” векторный редактор. Также научилась реализовывать “действие назад” при добавлении и удалении фигур, попрактиковалась в работе с умными указателями. При выполнении лабораторной работы очень удобным было не думать об освобождении памяти, при удалении фигур или всего вектора, так как использовались умные указатели.

Список литературы

1. Презентация “Проектируем структуру классов. ЛЕКЦИЯ 12” - Дзюба Д.В. (дата обращения: 7.12.20).
2. cppreference.com [Электронный ресурс]. URL: https://en.cppreference.com/w/ (дата обращения: 7.12.20).