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: Савченко Илья Владимирович
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;

Вариант 1:

Треугольник, квадрат, прямоугольник

2. **Описание программы**

main.cpp	
struct FileWork	Класс для работы с файлами
sruct LastOp	Структура для хранения последнего действия (для команды undo)
int main()	Драйвер код
point.h	
struct Point	Структура для хранения и обработки 2D точек/ векторов.

figures.h	

struct Figure
  virtual void print();
  virtual void serialize(ofstream&)
	Абстрактный класс фигур

struct Triangle : Figure struct Square : Figure struct Rect : Figure	Конкретные классы
factory.h	
struct Factory	Структура “фабрики”
Factory:: Figure* createTri(Point&, Point&, Point&); Figure* createSq(Point&, Point&); Figure* createRct(Point&, Point&, float)	Функции создания фигур

3. Набор тестов

Программа принимает на вход запрос на создание фигур, через текстовый интерфейс задаются их координаты, после чего их можно удалять, записывать и читать из файла.

Пример работы в пункте 4

4. Результаты выполнения тестов

```
     point is 2 numbers
t          - triangle
q          - square
r          - rectangle
l          - list figures
d <index>  - delete
u          - undo last action
s <name>   - save to file
o <name>   - open from file
x          - exit
> t
Triangle
  a: 0 0
  b: 0 4
  c: 3 2

> q
Square
```

```
  a: 0 0
  c: 5 5

> r
Rect
  a: 0 0
  b: 3 4
  ratio: 2

> l
0    Tri:     a{0, 0}   b{0, 4}   c{3, 2}
1    Sqr:     a{0, 0}   b{0, 5}   c{5, 5}   d{5, 0}
2    Rct:     a{0, 0}   b{3, 4}   c{11, -2}   d{8, -6}

> s figs
Saved as figs

> d 1

> l
0    Tri:     a{0, 0}   b{0, 4}   c{3, 2}
1    Rct:     a{0, 0}   b{3, 4}   c{11, -2}   d{8, -6}

> d 0

> l
0    Rct:     a{0, 0}   b{3, 4}   c{11, -2}   d{8, -6}

> u
Deletion canceled

> q
Square
  a: 1 1
  c: 2 2

> u
Addition canceled

> l
0    Rct:     a{0, 0}   b{3, 4}   c{11, -2}   d{8, -6}
1    Tri:     a{0, 0}   b{0, 4}   c{3, 2}

> o figs
Read successful (3 figures)

> l
0    Tri:     a{0, 0}   b{0, 4}   c{3, 2}
```

```
1    Sqr:     a{0, 0}   b{0, 5}   c{5, 5}   d{5, 0}
2    Rct:     a{0, 0}   b{3, 4}   c{11, -2}   d{8, -6}

> x
exit
```

5. Листинг программы

```
////////// main.cpp
/**
 * Савченко И.В.
 * М8О-208Б-19
 * https://github.com/ShyFly46/oop_exercise_07
 *
 * Вариант 1:
 * "Графический векторный редактор"
 * Треугольник, квадрат, прямоугольник
 **/

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

#include "point.h"
#include "figures.h"
#include "factory.h"

using namespace std;

struct FileWork {
    vector<Figure*> &vec;

    FileWork(vector<Figure*> &v)
       : vec(v)
    {}

    void Write(const string& fName){
       ofstream myFile(fName);
       for(Figure* i : vec){
             i->serialize(myFile);
       }
       myFile.close();
       cout
             << "Saved as "
             << fName
             << '\n';
    }

    void Read(const string& fName){
       ifstream myFile(fName);
       if(!myFile.is_open()){
             cout << "File not found\n";
             return;
       }
```

```
        vec.clear();

        char figType;
        Point a, b, c;
        Factory fac;

        while(myFile.get(figType)){
              switch(figType){
                    case 't':
                          myFile >> a >> b >> c;
                          vec.push_back(fac.createTri(a, b, c));
                          break;
                    case 'q':
                          myFile >> a >> c;
                          vec.push_back(fac.createSq(a, c));
                          break;
                    case 'r':
                          myFile >> a >> b >> c;
                          vec.push_back(fac.createRct(a, b, c));
                          break;
                    case ' ':
                    case '\t':
                    case '\n':
                          break;
                    default:
                          cout << "File not supported\n";
                          cout << "Read "
                               << vec.size()
                                 << " figures so far\n";
                                 myFile.close();
                                 return;
              }
        }

        myFile.close();
        cout
              << "Read successful ("
              << vec.size()
              << " figures)\n";
    }
};

struct LastOp{
    enum oper{NUL, ADD, DEL};
    oper op = NUL;
    Figure* ptr = nullptr;

    void Update(Figure* newPtr = nullptr){
        delete ptr;
        ptr = newPtr;
        if(!newPtr){
              op = ADD;
              return;
        }
        op = DEL;
    }
};
```

```
void help(){
    cout
       << " point is 2 numbers\n"
       << "t       - triangle\n"
       << "q       - square\n"
       << "r       - rectangle\n"
       << "l       - list figures\n"
       << "d <index> - delete\n"
       << "u       - undo last action\n"
       << "s <name>  - save to file\n"
       << "o <name>  - open from file\n"
       << "x       - exit\n";
}

int main(){
    Factory myFac;
    vector<Figure*> figVec;
    FileWork fileW(figVec);
    Figure* fig;

    LastOp lOp;

    char cmd;

    help();

    cout << "> ";
    while(cin >> cmd){
       switch(cmd){
             case 'h':
                   help();
                   break;
             case 't':
                   {
                         cout << "Triangle\n";
                         Point a, b, c;
                         cout << "  a: ";
                         cin >> a;
                         cout << "  b: ";
                         cin >> b;
                         cout << "  c: ";
                         cin >> c;

                         fig = myFac.createTri(a, b, c);
                         figVec.push_back(fig);

                         lOp.Update(nullptr);
                   }
                   break;
             case 'q':
                   {
                         cout << "Square\n";
                         Point a, c;
                         cout << "  a: ";
                         cin >> a;
                         cout << "  c: ";
                         cin >> c;
```

```
			fig = myFac.createSq(a, c);
			figVec.push_back(fig);

			lOp.Update();
		}
		break;
	case 'r':
		{
			cout << "Rect\n";
			Point a, b;
			cout << "  a: ";
			cin >> a;
			cout << "  b: ";
			cin >> b;
			cout << "  ratio: ";
			float r;
			cin >> r;

			fig = myFac.createRct(a, b, r);
			figVec.push_back(fig);

			lOp.Update();
		}
		break;
	case 'd':
		{
			unsigned int index;
			cin >> index;
			try{
				Figure *ptr = figVec.at(index);
				lOp.Update(ptr);
				figVec.erase(figVec.begin() + index);
			}
			catch(out_of_range& e){
				cout << "Out of range\n";
			}
		}
		break;
	case 'u':
		{
			if(lOp.op == LastOp::ADD){
				delete figVec.back();
				figVec.pop_back();
				cout << "Addition canceled\n";
			} else if(lOp.op == LastOp::DEL){
				figVec.push_back(lOp.ptr);
				cout << "Deletion canceled\n";
			} else {
				cout << "Nothing to undo\n";
			}
			lOp.op = LastOp::NUL;
			lOp.ptr = nullptr;
		}
		break;
	case 'l':
		{
			int ind = 0;
			for(Figure* i : figVec){
```

```
                                cout << ind << '\t';
                                i->print();
                                ++ind;
                          }
                    }
                    break;
              case 's':
                    {
                          string name;
                          cin >> name;
                          fileW.Write(name);
                    }
                    break;
              case 'o':
                    {
                          string name;
                          cin >> name;
                          fileW.Read(name);
                    }
                    break;
              case 'x':
                    cout << "exit\n";
                    return 0;
              default:
                    cout << "???\n";
                    break;
        }
        cout << "\n> ";
    }
}
////////// point.h
#ifndef POINT_H
#define POINT_H

struct Point{
    float x, y;

    Point(float x = 0, float y = 0)
       : x(x), y(y)
    {}
};

Point operator+ (const Point& a, const Point& b){
    return Point(a.x + b.x , a.y + b.y);
}
Point operator- (const Point& a, const Point& b){
    return Point(a.x - b.x , a.y - b.y);
}
Point operator* (const Point& p, float f){
    return Point(p.x * f, p.y * f);
}

std::ostream& operator<<(std::ostream& out, const Point& p){
    out << "{"
      << p.x
      << ", "
      << p.y
      << "}";
```

```
    return out;
}
std::istream& operator>>(std::istream& in, Point& p){
    in >> p.x >> p.y;
    return in;
}

#endif
////////// figures.h
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>
#include <fstream>
#include "point.h"

using namespace std;

struct Figure {
    virtual void print() = 0;
    virtual void serialize(ofstream &file) = 0;
};

struct Triangle : public Figure{
    Point a, b, c;

    void print() override{
       std::cout
             << "Tri: \t"
             << "a"     << a
             << "   b" << b
             << "   c" << c
             << '\n';
    };

    void serialize(ofstream &file) override{
       file
             << "t "
             << a.x << " " << a.y
             << " "
             << b.x << " " << b.y
             << " "
             << c.x << " " << c.y
             << '\n';
    }
};

struct Square : public Figure{
    Point a, b, c, d;

    void print() override{
       std::cout
             << "Sqr: \t"
             << "a"     << a
             << "   b" << b
             << "   c" << c
             << "   d" << d
             << '\n';
```

```
    }

    void serialize(ofstream &file) override{
       file
             << "q "
             << a.x << " " << a.y
             << " "
             << c.x << " " << c.y
             << '\n';
    }
};

struct Rect : public Figure{
    Point a, b, c, d;

    void print() override{
       std::cout
             << "Rct: \t"
             << "a"     << a
             << "   b" << b
             << "   c" << c
             << "   d" << d
             << '\n';
    }

    void serialize(ofstream &file) override{
       file
             << "r "
             << a.x << " " << a.y
             << " "
             << b.x << " " << b.y
             << " "
             << c.x << " " << c.y
             << '\n';
    }
};

#endif
////////// factory.h
#ifndef FACTORY_H
#define FACTORY_H

#include"figures.h"
#include"point.h"

struct Factory{
    Figure* createTri(const Point& a, const Point& b, const Point& c){
       Triangle* newF = new Triangle;
       newF->a = a;
       newF->b = b;
       newF->c = c;
       return (Figure*) newF;
    }
    Figure* createSq(const Point& a, const Point& c){
       Square* newF = new Square;
       newF->a = a;
       newF->c = c;
```

```
        Point cent = (a+c) * 0.5;
        Point vec = c - cent;
        // rotate by 45 deg clockwise
        float tmp = -vec.x;
        vec.x = vec.y;
        vec.y = tmp;

        newF->b = cent - vec;
        newF->d = cent + vec;

        return (Figure*) newF;
    }
    Figure* createRct(const Point& a, const Point& b, float ratio){
        Rect* newF = new Rect;
        newF->a = a;
        newF->b = b;

        Point vec = b - a;
        Point v2 = vec;
        // rotate by 45 deg clockwise
        float tmp = -vec.x;
        vec.x = vec.y;
        vec.y = tmp;
        //length ratio'd
        vec = vec * ratio;

        newF->d = a + vec;
        newF->c = newF->d + v2;

        return (Figure*) newF;
    }
    Figure* createRct(const Point& a, const Point& b, const Point& c){
        Rect* newF = new Rect;
        newF->a = a;
        newF->b = b;
        newF->c = c;

        newF->d = a + (c - b);

        return (Figure*) newF;
    }
};
#endif
```

Список литературы

1. Руководство по языку C++ [Электронный ресурс].
   URL: https://www.cplusplus.com/
   (дата обращения 20.04.2021).
2. Design Pattern — Factory Pattern [Электронный ресурс].
   URL: https://www.tutorialspoint.com/design_pattern/factory_pattern.htm
   (дата обращения 20.04.2021).