**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: Воронов Кирилл Михайлович

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.

Требование 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 27 Прямоугольник, трапеция, ромб.*

**2. Описание программы**

Создадим абстрактный класс Figure, чтобы наследовать от него классы других фигур (Rectangle, Romb, Trapeze) и иметь возможность хранить их вместе. Также реализуем класс Factory, который будет читать и создавать фигуры, и сам класс Document, в котором и будут реализованы функции добавления, удаления (с помощью абстрактного класса Action и наследованных от него классов) , а также сохранения/загрузки в файл/с файла и undo. Взаимодействие с программой происходит через интерактивное меню, в котором указаны номера команд и их действие. Команды вызывают соответствующие функции.

**3. Набор тестов и результаты их выполнения**

**Тест 1**

Введите 1, чтобы создать документ

Введите 2, чтобы добавить фигуру

Введите 3, чтобы удалить фигуру

Введите 4, сохранить документ в файл

Введите 5, чтобы загрузить элемент из файла

Введите 6, чтобы вывести содержимое документа

Введите 7, чтобы отменить последнее действие(удаление, добавление)

Введите 8, чтобы вывести меню

1

Документ создан

2

Введите 1, чтобы добавить прямоугольник

Введите 2, чтобы добавить ромб

Введите 3, чтобы добавить трапецию

2

Введите позицию вставки

0

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной

0 0 1 1

2

Введите 1, чтобы добавить прямоугольник

Введите 2, чтобы добавить ромб

Введите 3, чтобы добавить трапецию

1

Введите позицию вставки

1

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной

0 0 6 7

6

------------------

Rectangle

(0,7) (6,7)

(0,0) (6,0)

------------------

Romb

(0,1)

(-0.5,0.5) (0.5,0.5)

(0,0)

3

Введите позицию удаления

0

6

------------------

Romb

(0,1)

(-0.5,0.5) (0.5,0.5)

(0,0)

7

6

------------------

Rectangle

(0,7) (6,7)

(0,0) (6,0)

------------------

Romb

(0,1)

(-0.5,0.5) (0.5,0.5)

(0,0)

**Тест 2**

Введите 1, чтобы создать документ

Введите 2, чтобы добавить фигуру

Введите 3, чтобы удалить фигуру

Введите 4, сохранить документ в файл

Введите 5, чтобы загрузить элемент из файла

Введите 6, чтобы вывести содержимое документа

Введите 7, чтобы отменить последнее действие(удаление, добавление)

Введите 8, чтобы вывести меню

1

Документ создан

2

Введите 1, чтобы добавить прямоугольник

Введите 2, чтобы добавить ромб

Введите 3, чтобы добавить трапецию

3

Введите позицию вставки

0

Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину высоты (трапеция равнобокая)

0 -1 5 6 2

2

Введите 1, чтобы добавить прямоугольник

Введите 2, чтобы добавить ромб

Введите 3, чтобы добавить трапецию

1

Введите позицию вставки

0

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной

0 -1 9 5

2

Введите 1, чтобы добавить прямоугольник

Введите 2, чтобы добавить ромб

Введите 3, чтобы добавить трапецию

2

Введите позицию вставки

0

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной

0 2 3 3

6

------------------

Romb

(0,5)

(-1.5,3.5) (1.5,3.5)

(0,2)

------------------

Rectangle

(0,4) (9,4)

(0,-1) (9,-1)

------------------

Trapeze

(-0.5,1) (5.5,1)

(0,-1) (0,4)

4

Введите расположение файла

a.txt

Документ сохранён

3

Введите позицию удаления

0

3

Введите позицию удаления

0

3

Введите позицию удаления

0

6

Документ пуст

5

Введите расположение файла

a.txt

Документ загружен

6

------------------

Romb

(0,5)

(-1.5,3.5) (1.5,3.5)

(0,2)

------------------

Rectangle

(0,4) (9,4)

(0,-1) (9,-1)

------------------

Trapeze

(-0.5,1) (5.5,1)

(0,-1) (0,4)

**4. Листинг программы**

/\*Воронов К.М.

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.

Требование 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 27 Прямоугольник, трапеция, ромб.

\*/

#include <iostream>

#include <string>

#include <algorithm>

#include <memory>

#include <list>

#include <stack>

#include <cmath>

#include <string>

using namespace std;

class Figure {

public:

virtual void Print() = 0;

virtual void Write(FILE\* out) = 0;

virtual ~Figure() {}

};

const unsigned long long Rec\_Id = 1;

const unsigned long long Romb\_Id = 2;

const unsigned long long Tr\_Id = 3;

template <typename T>

class Rectangle: public Figure{

public:

using type = T;

pair<T,T> t;

T a;

T b;

Rectangle(){};

Rectangle(T x,T y,T a1, T b1): t(x,y), a(a1), b(b1){};

void Print() override {

pair<double,double> t2(t.first + a, t.second );

pair<double,double> t3(t.first + a, t.second + b);

pair<double,double> t4(t.first, t.second + b);

cout << "------------------" << endl;

cout << "Rectangle" << endl;

cout << "(" << t4.first << "," << t4.second << ") "

<< "(" << t3.first << "," << t3.second << ")"

<< endl << endl << "(" << t.first << ","

<< t.second << ") " << "(" << t2.first << ","

<< t2.second << ")" << endl;

}

void Write(FILE\* out) override {

fwrite(&Rec\_Id, sizeof(unsigned long long), 1, out);

fwrite(&t.first, sizeof(T), 1, out);

fwrite(&t.second, sizeof(T), 1, out);

fwrite(&a, sizeof(T), 1, out);

fwrite(&b, sizeof(T), 1, out);

}

};

template <typename T>

class Romb: public Figure{

public:

using type = T;

pair<T,T> t;

T a;

T b;

Romb(T x,T y,T a1, T b1): t(x,y), a(a1), b(b1){};

void Print() override {

pair<T,T> t2(t.first,t.second+b);

pair<T,T> t3(t.first + a/2,t.second+b/2);

pair<T,T> t4(t.first - a/2,t.second+b/2);

cout << "------------------" << endl;

cout << "Romb" << endl;

cout << " (" << t2.first << "," << t2.second << ") " << endl

<< endl << "(" << t4.first << "," << t4.second << ") " << "("

<< t3.first << "," << t3.second << ")" << endl << endl << " ("

<< t.first << "," << t.second << ")" << endl;

}

void Write(FILE\* out) override {

fwrite(&Romb\_Id, sizeof(unsigned long long), 1, out);

fwrite(&t.first, sizeof(T), 1, out);

fwrite(&t.second, sizeof(T), 1, out);

fwrite(&a, sizeof(T), 1, out);

fwrite(&b, sizeof(T), 1, out);

}

};

template <typename T>

class Trapeze: public Figure{

public:

using type = T;

pair<T,T> t;

T a;

T b;

T c;

Trapeze(T x,T y,T a1, T b1, T s1): t(x,y), a(a1), b(b1), c(s1){};

void Print() override {

pair<T,T> t2(t.first,t.second + a);

pair<T,T> t3(t.first + (a - b)/2, t.second + c);

pair<T,T> t4(t2.first + a - (a - b)/2, t.second + c);

cout << "------------------" << endl;

cout << "Trapeze" << endl;

cout << " (" << t3.first << "," << t3.second << ") ("

<< t4.first << "," << t4.second << ")" << endl << endl

<< "(" << t.first << "," << t.second << ") ("

<< t2.first << "," << t2.second << ")" << endl;

}

void Write(FILE\* out) override {

fwrite(&Tr\_Id, sizeof(unsigned long long), 1, out);

fwrite(&t.first, sizeof(T), 1, out);

fwrite(&t.second, sizeof(T), 1, out);

fwrite(&a, sizeof(T), 1, out);

fwrite(&b, sizeof(T), 1, out);

fwrite(&c, sizeof(T), 1, out);

}

};

template<class T, class FIGURE>

class Factory;

template<class T>

class Factory<T, Rectangle<T> > {

public:

static shared\_ptr<Figure> Create() {

pair <T,T> c;

T a,b;

printf("Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной\n");

cin >> c.first >> c.second >> a >> b;

if((!cin)||(a<=0)||(b<=0)){

printf("Введены неверные данные, взяты значения по умолчанию: 0 0 1 1\n");

c.first = 0;

c.second = 0;

a = 1;

b = 1;

}

Rectangle<T> \* rect = new Rectangle<T>(c.first,c.second, a, b);

return shared\_ptr<Figure>(rect);

}

static shared\_ptr<Figure> Read(FILE\* in) {

pair <T,T> c;

T a,b;

fread(&c.first, sizeof(T), 1, in);

fread(&c.second, sizeof(T), 1, in);

fread(&a, sizeof(T), 1, in);

fread(&b, sizeof(T), 1, in);

Rectangle<T> \* rect = new Rectangle<T>(c.first,c.second, a, b);

return shared\_ptr<Figure>(rect);

}

};

template<class T>

class Factory<T, Romb<T> > {

public:

static shared\_ptr<Figure> Create() {

pair <T,T> c;

T a,b;

printf("Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной\n");

cin >> c.first >> c.second >> a >> b;

if((!cin)||(a<=0)||(b<=0)){

printf("Введены неверные данные, взяты значения по умолчанию: 0 0 1 1\n");

c.first = 0;

c.second = 0;

a = 1;

b = 1;

}

Romb<T> \* romb = new Romb<T>(c.first,c.second, a, b);

return shared\_ptr<Figure>(romb);

}

static shared\_ptr<Figure> Read(FILE\* in) {

pair <T,T> c;

T a,b;

fread(&c.first, sizeof(T), 1, in);

fread(&c.second, sizeof(T), 1, in);

fread(&a, sizeof(T), 1, in);

fread(&b, sizeof(T), 1, in);

Romb<T> \* romb = new Romb<T>(c.first, c.second, a, b);

return shared\_ptr<Figure>(romb);

}

};

template<class T>

class Factory<T, Trapeze<T> > {

public:

static shared\_ptr<Figure> Create() {

pair <T,T> c;

T a,b,d;

printf("Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину высоты (трапеция равнобокая)\n");

cin >> c.first >> c.second >> a >> b >> d;

if((a<=0)||(b<=0)||(d<=0)){

printf("Введены неверные данные, взяты значения по умолчанию 0 0 5 2 4\n");

c.first = 0;

c.second = 0;

a = 10;

b = 5;

d = 5;

}

Trapeze<T> \* tr = new Trapeze<T>(c.first,c.second, a, b, d);

return shared\_ptr<Figure>(tr);

}

static shared\_ptr<Figure> Read(FILE\* in) {

pair <T,T> c;

T a,b,d;

fread(&c.first, sizeof(T), 1, in);

fread(&c.second, sizeof(T), 1, in);

fread(&a, sizeof(T), 1, in);

fread(&b, sizeof(T), 1, in);

fread(&d, sizeof(T), 1, in);

Trapeze<T> \* tr = new Trapeze<T>(c.first,c.second, a, b, d);

return shared\_ptr<Figure>(tr);

}

};

template<class T>

class Document {

public:

class Action {

public:

virtual void PerformAction(Document & fact) = 0;

virtual ~Action() {}

};

stack<shared\_ptr <Action> > st\_ac;

list<shared\_ptr <Figure> > ls\_fig;

class AddAction: public Action {

public:

unsigned long long Position;

shared\_ptr<Figure> AddFigure;

AddAction(unsigned long long & pos, shared\_ptr<Figure> & fig) : Position(pos), AddFigure(fig) {}

void PerformAction(Document & fact) override {

fact.AddFigure(Position,AddFigure);

}

};

class DeleteAction: public Action {

public:

unsigned long long Position;

DeleteAction(unsigned long long pos): Position(pos){}

void PerformAction(Document & fact) override {

fact.Delete(Position);

}

};

void Create() {

while (!st\_ac.empty()) {

st\_ac.pop();

}

ls\_fig.clear();

}

void Add (unsigned long long & pos, unsigned long long & id) {

if (id == Rec\_Id) {

AddFigure(pos, Factory<T, Rectangle<T>>:: Create());

} else {

if (id == Romb\_Id) {

AddFigure(pos, Factory<T, Romb<T>>:: Create());

} else {

if (id == Tr\_Id) {

AddFigure(pos, Factory<T, Trapeze<T>>:: Create());

}

}

}

}

void AddFigure(unsigned long long & pos, shared\_ptr<Figure> f) {

list< shared\_ptr <Figure> >::iterator it = ls\_fig.begin();

unsigned long long cur\_pos = 0;

while (cur\_pos > pos && pos < ls\_fig.size() + 1) {

++it;

++cur\_pos;

}

DeleteAction\* delAct = new DeleteAction(pos + 1);

st\_ac.push(shared\_ptr<Action>(delAct));

ls\_fig.insert(it, f);

}

void Delete(unsigned long long & pos) {

if (ls\_fig.empty()) {

printf("Нечего удалять\n");

return;

}

list< shared\_ptr <Figure> >::iterator it = ls\_fig.begin();

unsigned long long cur\_pos = 0;

while (cur\_pos > pos && pos < ls\_fig.size() + 1) {

++it;

++cur\_pos;

}

AddAction\* addAct = new AddAction(cur\_pos, \*it);

st\_ac.push(shared\_ptr<Action>(addAct));

ls\_fig.erase(it);

}

void Save(FILE\* out) {

unsigned long long size = ls\_fig.size();

fwrite(&size, sizeof(unsigned long long), 1, out);

for (list< shared\_ptr <Figure> >::iterator it = ls\_fig.begin(); it != ls\_fig.end(); ++it) {

(\*it)->Write(out);

}

}

void Load(FILE\* in) {

Create();

unsigned long long size;

fread(&size, sizeof(unsigned long long), 1, in);

for (unsigned long long i = 0; i < size; ++i){

unsigned long long id;

fread(&id, sizeof(unsigned long long), 1, in);

if (id == Rec\_Id) {

ls\_fig.push\_back(Factory<T, Rectangle<T>>:: Read(in));

} else {

if (id == Romb\_Id) {

ls\_fig.push\_back(Factory<T, Romb<T>>:: Read(in));

} else {

if (id == Tr\_Id) {

ls\_fig.push\_back(Factory<T,Trapeze<T>>:: Read(in));

}

}

}

}

}

void Undo() {

if(st\_ac.empty()) {

printf("Невозможно отменить последнее действие\n");

return;

}

st\_ac.top()->PerformAction(\*this);

st\_ac.pop();

st\_ac.pop();

}

};

int main() {

Document<double> doc;

int input;

printf("Введите 1, чтобы создать документ\n");

printf("Введите 2, чтобы добавить фигуру\n");

printf("Введите 3, чтобы удалить фигуру\n");

printf("Введите 4, сохранить документ в файл\n");

printf("Введите 5, чтобы загрузить элемент из файла\n");

printf("Введите 6, чтобы вывести содержимое документа\n");

printf("Введите 7, чтобы отменить последнее действие(удаление, добавление)\n");

printf("Введите 8, чтобы вывести меню\n");

while (cin >> input) {

if (!cin){

cout << "Введены неверные данные" << endl;

return -1;

}

switch(input){

case 1:

{

doc.Create();

printf("Документ создан\n");

break;

}

case 2:

{

printf("Введите 1, чтобы добавить прямоугольник\n");

printf("Введите 2, чтобы добавить ромб\n");

printf("Введите 3, чтобы добавить трапецию\n");

unsigned long long type;

cin >> type;

if (!cin && type != 1 && type != 2 && type != 3) {

printf("Вы ввели неверную команду\n");

return -1;

}

printf("Введите позицию вставки\n");

unsigned long long p;

cin >> p;

if (!cin) {

printf("Вы ввели неверную позицию\n");

return -1;

}

doc.Add(p,type);

break;

}

case 3:

{

unsigned long long poss;

printf("Введите позицию удаления\n");

cin >> poss;

if (!cin) {

printf("Вы ввели неверную позицию\n");

return -1;

}

doc.Delete(poss);

break;

}

case 4:

{

string s;

printf("Введите расположение файла\n");

cin >> s;

FILE \*file = fopen(s.c\_str(),"wb");

if (file == NULL) {

printf("Не удалось открыть файл\n");

} else {

doc.Save(file);

fclose(file);

printf("Документ сохранён\n");

}

break;

}

case 5:

{

string s;

printf("Введите расположение файла\n");

cin >> s;

FILE \*file = fopen(s.c\_str(),"rb");

if (file == NULL) {

printf("Не удалось открыть файл\n");

} else {

doc.Load(file);

fclose(file);

printf("Документ загружен\n");

}

break;

}

case 6:

{

if (doc.ls\_fig.size() == 0) {

printf("Документ пуст\n");

}

list< shared\_ptr <Figure> >::iterator it = doc.ls\_fig.begin();

for (int i = 0; i < doc.ls\_fig.size(); ++i) {

(\*it)->Print();

++it;

}

break;

}

case 7:

{

doc.Undo();

break;

}

case 8:

{

printf("Введите 1, чтобы создать документ\n");

printf("Введите 2, чтобы добавить фигуру\n");

printf("Введите 3, чтобы удалить фигуру\n");

printf("Введите 4, сохранить документ в файл\n");

printf("Введите 5, чтобы загрузить элемент из файла\n");

printf("Введите 6, чтобы вывести содержимое документа\n");

printf("Введите 7, чтобы отменить последнее действие(удаление, добавление)\n");

printf("Введите 8, чтобы вывести меню\n");

break;

}

default:

cout<<"Вы ввели неверную команду"<<std::endl;

return -1;

break;

}

}

}

**5. Выводы**

В ходе выполнения лабораторной работы я спроектировал систему классов для графического редактора и реализовал запись структур классов в файл.

**Список литературы**

1. Мануал std::list [Электронный ресурс]

URL: <https://en.cppreference.com/w/cpp/container/list> (дата обращения 12.12.2020)

1. Мануал std::stack [Электронный ресурс]

URL: https://en.cppreference.com/w/cpp/container/stack (дата обращения 12.12.2020)