Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «ООП»**

**Тема:**

**Проектирование структуры классов.**

|  |  |
| --- | --- |
| Студент: | Рыженко И.А. |
| Группа: | М80-208Б-18 |
| Преподаватель: | Поповкин А.В. |
| Вариант: | 22 |
| Оценка: |  |
| Дата: | 23.03.2021 |

Москва

2021

1.**Код на C++:**

#pragma once

#include <iostream>

template <class T>

struct **Vertex**

{

using m\_Vertex = std::pair<T,T>;

m\_Vertex coord;

void **read\_file**(std::istream& is);

};

template <class T>

void Vertex<T>::**read\_file**(std::istream& is)

{

is >> this->coord.first >> this->coord.second;

}

template <class T>

std::istream& operator>> (std::istream& is, Vertex<T>& p){

std::cout << "Coordinate 'x': ";

is >> p.coord.first;

std::cout << "Coordinate 'y': ";

is >> p.coord.second;

return is;

}

template <class T>

std::ostream& operator<< (std::ostream& os, const Vertex<T>& p){

os << "[" << p.coord.first << "; " << p.coord.second << "]";

return os;

}

template<class T>

Vertex<T> operator+(const Vertex<T>& A, const Vertex<T>& B) {

Vertex<T> res;

res.coord.first = A.coord.first + B.coord.first;

res.coord.second = A.coord.second + B.coord.second;

return res;

}

template<class T>

Vertex<T> operator/=(Vertex<T>& res, const int& number) {

res.coord.first = res.coord.first / number;

res.coord.second = res.coord.second / number;

return res;

}

figure.h:

#ifndef D\_FIGURE\_H\_

#define D\_FIGURE\_H\_

#include <iostream>

#include "point.h"

struct **figure**

{

virtual Vertex<double> ***center***() const = 0;

virtual void ***print***(std::ostream &os) const = 0;

virtual void ***help\_print***(std::ostream &os) const = 0;

virtual void ***read\_file***(std::istream &is) = 0;

virtual double ***square***() const = 0;

virtual ~***figure***() {};

};

#endif

five\_angles.h:

#ifndef D\_FIVE\_ANGLES\_H\_

#define D\_FIVE\_ANGLES\_H\_

#include <iostream>

#include "figure.h"

struct **five\_angles** : figure

{

virtual ~***five\_angles***() override {};

**five\_angles**(std::istream &is);

**five\_angles**() = default;

Vertex<double> ***center***() const override;

void ***print***(std::ostream &os) const override;

void ***help\_print***(std::ostream &os) const override;

void ***read\_file***(std::istream &is) override;

double ***square***() const override;

private:

Vertex<double> points[5];

};

#endif

five\_angles.cpp:

#include <iostream>

#include "five\_angles.h"

five\_angles::**five\_angles**(std::istream &is)

{

for (size\_t i = 0; i < 5; i++)

{

std::cout << "Input the " << i+1 << " vertex: \n";

is >> this->points[i];

}

}

void five\_angles::***read\_file***(std::istream &is)//чтение вершин из файла

{

for (size\_t i = 0; i < 5; i++)

this->points[i].read\_file(*is*);

}

Vertex<double> five\_angles::***center***() const {

Vertex<double> p;

for (size\_t i = 0; i < 5; i++)

p = p + points[i];

p /= 5;

return p;

}

void five\_angles::***print***(std::ostream &os) const {

os << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " " << points[4] <<"\n";

}

void five\_angles::***help\_print***(std::ostream &os) const {

os << "1 " << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " " << points[4] <<"\n";

}

double five\_angles::***square***() const {

double s=0;

s = std::abs(points[0].coord.first\*points[1].coord.second+points[1].coord.first\*points[2].coord.second+points[2].coord.first\*points[3].coord.second+points[3].coord.first\*

points[4].coord.second+points[4].coord.first\*points[0].coord.second-points[1].coord.first\*points[0].coord.second-

points[2].coord.first\*points[1].coord.second-points[3].coord.first\*points[2].coord.second-

points[4].coord.first\*points[3].coord.second-points[0].coord.first\*points[4].coord.second)/2;

return s;

}

six\_angles.h:

#ifndef D\_SIX\_ANGLES\_H\_

#define D\_SIX\_ANGLES\_H\_

#include <iostream>

#include "figure.h"

struct **six\_angles** : figure

{

virtual ~***six\_angles***() override {};

**six\_angles**(std::istream &is);

**six\_angles**() = default;

Vertex<double> ***center***() const override;

void ***print***(std::ostream &os) const override;

void ***help\_print***(std::ostream &os) const override;

void ***read\_file***(std::istream &is) override;

double ***square***() const override;

private:

Vertex<double> points[6];

};

#endif

six\_angles.cpp:

#include <iostream>

#include "six\_angles.h"

six\_angles::**six\_angles**(std::istream &is)

{

for (size\_t i = 0; i < 6; i++)

{

std::cout << "Input the " << i+1 << " vertex: \n";

is >> this->points[i];

}

}

void six\_angles::***read\_file***(std::istream &is)//чтение вершин из файла

{

for (size\_t i = 0; i < 6; i++)

this->points[i].read\_file(*is*);

}

Vertex<double> six\_angles::***center***() const {

Vertex<double> p;

for (size\_t i = 0; i < 6; i++)

p = p + points[i];

p/=6;

return p;

}

void six\_angles::***print***(std::ostream &os) const {

os << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " " << points[4] << " " << points[5] <<"\n";

}

void six\_angles::***help\_print***(std::ostream &os) const {

os <<"2 " << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " " << points[4] << " " << points[5] <<"\n";

}

double six\_angles::***square***() const {

double s=0;

s = std::abs(points[0].coord.first\*points[1].coord.second+points[1].coord.first\*points[2].coord.second+points[2].coord.first

\*points[3].coord.second+points[3].coord.first\*points[4].coord.second+points[4].coord.first\*points[5].coord.second

+points[5].coord.first\*points[0].coord.second-points[1].coord.first\*points[0].coord.second-

points[2].coord.first\*points[1].coord.second-points[3].coord.first\*points[2].coord.second

-points[4].coord.first\*points[3].coord.second-points[5].coord.first\*points[4].coord.second-points[0].coord.first\*points[5].coord.second)/2;

return s;

}

eight\_angles.h:

#ifndef D\_EIGHT\_ANGLES\_H\_

#define D\_EIGHT\_ANGLES\_H\_

#include <iostream>

#include "figure.h"

struct **eight\_angles** : figure

{

virtual ~***eight\_angles***() override {};

**eight\_angles**(std::istream &is);

**eight\_angles**() = default;

Vertex<double> ***center***() const override;

void ***print***(std::ostream &os) const override;

void ***help\_print***(std::ostream &os) const override;

void ***read\_file***(std::istream &is) override;

double ***square***() const override;

private:

Vertex<double> points[8];

};

#endif

eight\_angles.cpp:

#include <iostream>

#include "eight\_angles.h"

eight\_angles::**eight\_angles**(std::istream &is)//метод ввода вершин восьмиугольника

{

for (size\_t i = 0; i < 8; i++)

{

std::cout << "Input the " << i+1 << " vertex: \n";

is >> this->points[i];

}

}

void eight\_angles::***read\_file***(std::istream &is)//чтение вершин из файла

{

for (size\_t i = 0; i < 8; i++)

this->points[i].read\_file(*is*);

}

Vertex<double> eight\_angles::***center***() const //метод подсчёта центра фигуры

{

Vertex<double> p;

for (size\_t i = 0; i < 8; i++)

p = p + points[i];

p/=8;

return p;

}

void eight\_angles::***print***(std::ostream &os) const {//метод вывода в консоль

os << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " " << points[4] << " " << points[5] << " " << points[6]

<< " " << points[7]<<"\n";

}

void eight\_angles::***help\_print***(std::ostream &os) const {//метод вывода в файл

os <<"3 " << points[0] << " " << points[1] << " " << points[2] << " " << points[3] << " "

<< points[4] << " " << points[5] << " " << points[6]

<< " " << points[7]<<"\n";

}

double eight\_angles::***square***() const {//метод подсчёта площади фигуры

double s=0;

s = std::abs(points[0].coord.first\*points[1].coord.second+points[1].coord.first\*points[2].coord.second+points[2].coord.first\*

points[3].coord.second+points[3].coord.first\*points[4].coord.second+points[4].coord.first\*points[5].coord.second+

points[5].coord.first\*points[6].coord.second+points[6].coord.first\*points[7].coord.second+

points[7].coord.first\*points[0].coord.second-points[1].coord.first\*points[0].coord.second-points[2].coord.first\*

points[1].coord.second-points[3].coord.first\*points[2].coord.second-points[4].coord.first\*points[3].coord.second-

points[5].coord.first\*points[4].coord.second-points[6].coord.first\*points[5].coord.second

-points[7].coord.first\*points[6].coord.second-points[0].coord.first\*points[7].coord.second)/2;

return s;

}

document.h:

#ifndef D\_DOCUMENT\_H\_

#define D\_DOCUMENT\_H\_

#include "figure.h"

#include "five\_angles.h"

#include "six\_angles.h"

#include "eight\_angles.h"

#include <vector>

#include <memory>

#include <iostream>

//Структура "Документ", представляющая из себя вектор фигур и операций над ними

struct **document**

{

**document**()= default;

void **save**(std::ostream& os) const;

void **load**(std::istream& is);

void **add\_figure**(std::unique\_ptr<figure>&& ptr, size\_t id);

void **remove\_figure**(size\_t id);

void **show**(std::ostream &os) const;

void **undo**();

struct **command**

{

size\_t id;

std::unique\_ptr<figure> ptr\_;

virtual void ***undo***(document &doc) = 0;

};

struct **add\_command**:public command

{

void ***undo***(document &doc) override;

};

struct **remove\_command**:public command

{

void ***undo***(document &doc) override;

};

private:

std::vector<std::unique\_ptr<figure>> figures\_;

std::vector<std::unique\_ptr<command>> operations\_;

};

#endif

document.cpp:

#include <iostream>

#include "document.h"

void document::**save**(std::ostream& os) const//сохранение в файл элементов коллекции

{

for (size\_t i = 0; i < figures\_.size(); ++i)

figures\_[i]->*help\_print*(*os*);

}

void document::**load**(std::istream& is)//загрузка из файла элементов коллекции

{

int help;

while(is >> help){

if(help == 1)

{

five\_angles fig;

fig.*read\_file*(*is*);

std::unique\_ptr<figure> new\_figure;

new\_figure=std::make\_unique<five\_angles>(*fig*);

figures\_.push\_back(std::move(*new\_figure*));

}

else if(help == 2)

{

six\_angles fig;

fig.*read\_file*(*is*);

std::unique\_ptr<figure> new\_figure;

new\_figure=std::make\_unique<six\_angles>(*fig*);

figures\_.push\_back(std::move(*new\_figure*));

}

else if(help == 3)

{

eight\_angles fig;

fig.*read\_file*(*is*);

std::unique\_ptr<figure> new\_figure;

new\_figure=std::make\_unique<eight\_angles>(*fig*);

figures\_.push\_back(std::move(*new\_figure*));

}

}

}

void document::**add\_figure**(std::unique\_ptr<figure>&& ptr,size\_t id)//добавление фигуры в коллекцию; id - место вставки

{

if (id >= this->figures\_.size())

{

std::cout << "Input index is out of bounds\n";

return;

}

figures\_.insert(figures\_.begin() + id,std::move(*ptr*));

add\_command op1;

std::unique\_ptr<add\_command> op;

op=std::make\_unique<add\_command>(std::move(*op1*));

op->id=id;

op->ptr\_ = nullptr;

operations\_.push\_back(std::move(*op*));

}

void document::**remove\_figure**(size\_t id)//удаление фигуры из коллекции с заданным индексом

{

if (id >= this->figures\_.size())

{

std::cout << "Input index is out of bounds\n";

return;

}

remove\_command op1;

std::unique\_ptr<remove\_command> op;

op=std::make\_unique<remove\_command>(std::move(*op1*));

op->id = id;

op->ptr\_=std::move(*figures\_[id]*);

operations\_.push\_back(std::move(*op*));

figures\_.erase(figures\_.begin() + id);

}

void document::**show**(std::ostream &os) const//метод, выводящий объекты, хранящиеся в коллекции, а также их параметры

{

if(figures\_.size()>0)

{

for (size\_t i = 0; i < figures\_.size(); ++i)

{

os << "Figure number " << i+1 << "\n";

os << "Cordinates of figure:";

figures\_[i]->*print*(*os*);

os << "Center: " << figures\_[i]->*center*() << "\n";

os << "Square: " << figures\_[i]->*square*() << "\n";

}

}

}

void document::**undo**()//отмена последней операции

{

if(operations\_.size()>0)

{

operations\_[operations\_.size()-1]->*undo*(*\**this);

operations\_.erase(operations\_.begin()+operations\_.size()-1);

}

}

void document::add\_command::***undo***(document &doc)//отмена операции добавления

{

doc.figures\_.erase(doc.figures\_.begin() + id);

}

void document::remove\_command::***undo***(document &doc)//отмена операции удаления

{

doc.figures\_.insert(doc.figures\_.begin() + id,std::move(*ptr\_*));

}

factory.h:

#ifndef D\_FACTORY\_H\_

#define D\_FACTORY\_H\_

#include "document.h"

#include "figure.h"

#include "five\_angles.h"

#include "six\_angles.h"

#include "eight\_angles.h"

#include <vector>

#include <memory>

#include <iostream>

struct **factory**

{

void **construct**(std::unique\_ptr<document>& vec);

};

#endif

factory.cpp:

#include <iostream>

#include "factory.h"

void factory::**construct**(std::unique\_ptr<document> &doc1)

{

std::string figures;

std::cout << "Input the number of figure for adding: \n";

std::cout << "Pentagon: 5\n";

std::cout << "Hexagon: 6\n";

std::cout << "Octagon: 8\n";

std::cin >> figures;

if (figures != "5" && figures != "6" && figures != "8")

{

std::cout << "Wrong input!\n";

return;

}

size\_t id;

std::cout << "Input the index for input: ";

std::cin >> id;

if(figures == "5")

{

std::unique\_ptr<figure> new\_figure;

new\_figure=std::make\_unique<five\_angles>(five\_angles(std::*cin*));

doc1->add\_figure(std::move(*new\_figure*),id);

}

else if(figures == "6")

{

std::unique\_ptr<figure> new\_figure;

new\_figure=std::make\_unique<six\_angles>( six\_angles(std*::cin*));

doc1->add\_figure(std::move(*new\_figure*),id);

}

else if(figures == "8")

{

std::unique\_ptr<figure> new\_figure;

new\_figure = std::make\_unique<eight\_angles>( eight\_angles(std::*cin*));

doc1->add\_figure(std::move(*new\_figure*),id);

}

}

main.cpp:

#include <iostream>

#include <string>

#include <stdio.h>

#include <vector>

#include <memory>

#include <fstream>

#include "figure.h"

#include "five\_angles.h"

#include "six\_angles.h"

#include "eight\_angles.h"

#include "document.h"

#include "factory.h"

void **PrintMenu**()

{

std::cout << "Save elements in file: 1" << std::endl;

std::cout << "Load elements from file: 2" << std::endl;

std::cout << "Add element: 3" << std::endl;

std::cout << "Remove element: 4" << std::endl;

std::cout << "Output elements: 5" << std::endl;

std::cout << "Undo last operation: 6" << std::endl;

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

}

int **main**()

{

std::cout << "Laba 07\n";

std::string command;

factory fact;

std::unique\_ptr<document> doc1;

doc1 = std::make\_unique<document>();

while(command != "0")

{

PrintMenu();

std::cin >> command;

if(command=="1")

{

std::cout << "Input the path to file for saving: ";

std::string path;

std::cin >> path;

std::ofstream os(path);

doc1->save(os);

os.close();//закрытие файла

}

else if(command=="2")

{

std::cout << "Input the path to file for loading: ";

std::string path;

std::cin >> path;

std::ifstream is(path);

if(is)

doc1->load(is);

else

std::cout << "No such file\n";

is.close();//закрытие файла

}

else if(command=="3")

fact.construct(*doc1*);

else if(command=="4")

{

std::cout << "Input the index for deleating: ";

size\_t id;

std::cin >> id;

doc1->remove\_figure(id);

}

else if(command== "5")

doc1->show(std*::cout*);

else if(command == "6")

doc1->undo();

std::cout << "\n\n\n";

}

return 0;

}

2. **Ссылка на репозиторий в GitHub:**

https://github.com/THEproVANO/oop\_exercise\_07

3.**Набор testcases:**

test1:

new

add five\_angles 0

1 1 2 2 3 3 4 4 5 5

add six\_angles 1

1 1 2 2 3 3 4 4 5 5 6 6

add eight\_angles 2

1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

show

remove 1

show

remove 0

show

test2:

new

add five\_angles 0

1 1 2 2 3 3 4 4 5 5

add six\_angles 1

1 1 2 2 3 3 4 4 5 5 6 6

add eight\_angles 2

1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

save tfile.txt

show

remove 0

remove 0

remove 0

show

load tfile.txt

show

test3:

new

add five\_angles 0

1 1 2 2 3 3 4 4 5 5

add six\_angles 1

1 1 2 2 3 3 4 4 5 5 6 6

add eight\_angles 2

1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

show

undo

show

remove 0

show

undo

show

undo

show

4.**Результаты выполнения программы:**

test1:

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 2

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

test2:

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 2

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 2

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

test3:

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 2

cordinats 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8

center 4.5 4.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

remove 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

figure 1

cordinats 1 1 2 2 3 3 4 4 5 5 6 6

center 3.5 3.5

square 0

figure 0

cordinats 1 1 2 2 3 3 4 4 5 5

center 3 3

square 0

5**. Объяснение результатов работы программы:**

Пользователь вводит команд , и её дополнительные атрибуты (имя файла, имя фигуры, координаты, позицию). В зависимости от этого программы выполняет одно из семи команд: создание нового документа, загрузка документа в файл, выгрузка документа из файла, добавление фигуры, удаление фигуры, показ всех фигур с их характеристиками, отмена последнего действия.

6.**Вывод:**

В данной программе показывается, каким образом можно создать собственный очень примитивный векторный графический редактор, чтобы наиболее простым образом показать, как происходит проектирование структуры классов.