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:	Журавлев А.А.
Вариант:	22
Оценка:	11.01.2020
Дата:	

Москва
2020

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].co-
ord.first*points[2].coord.second+points[2].coord.first*points[3].coord.sec-
ond+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].co-
ord.first*points[2].coord.second+points[2].coord.first
            *points[3].coord.second+points[3].coord.first*points[4].co-
ord.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 восьмиуголь-
ника
```

```
{
    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].co-
ord.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, храня-
щиеся 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.