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:	Суворова С. А.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	22
Оценка:	
Дата:	

Москва
2019

1.**Код на C++:**

point.h:
```
#ifndef D_POINT_H_
#define D_POINT_H_

#include <iostream>

struct point{
    double x,y;
};

std::istream& operator>>(std::istream& is, point& p);
std::ostream& operator<<(std::ostream& os,const point& p);
point operator+(point x1,point x2);
point& operator/= (point& x1, int number);

#endif
```

point.cpp:
```
#include <iostream>

#include "point.h"

std::istream& operator>> (std::istream& is, point& p){
    is >> p.x >>p.y;
    return is;
}

std::ostream& operator<< (std::ostream& os, const point& p){
    os << p.x << " " << p.y;
    return os;
}

point operator+(point x1,point x2){
    point x3;
    x3.x=x1.x+x2.x;
    x3.y=x1.y+x2.y;
    return x3;
}
point& operator/= (point& x1, int number){
    x1.x=x1.x/number;
    x1.y=x1.y/number;
    return x1;
}
```

figure.h:
```
#ifndef D_FIGURE_H_
#define D_FIGURE_H_

#include <iostream>

#include "point.h"

struct figure{
    virtual point center() const = 0;
    virtual void print(std::ostream &os) const = 0;
    virtual void help_print(std::ostream &os) const = 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{

    five_angles(std::istream &is);

    point center() const override;
    void print(std::ostream &os) const override;
    void help_print(std::ostream &os) const override;
    double square() const override;

private:
point one,two,three,four,five;

};

#endif
```

five_angles.cpp:

```
#include <iostream>

#include "five_angles.h"

five_angles::five_angles(std::istream &is){
    is >> one >> two >> three >> four >> five;
}

point five_angles::center() const {
    point p;
    p=one+two+three+four+five;
    p/=5;
    return p;
}

void five_angles::print(std::ostream &os) const {
    os << one << " " << two << " " << three << " " << four << " " << five
<<"\n";
}
void five_angles::help_print(std::ostream &os) const {
    os << "1 " << one << " " << two << " " << three << " " << four << " " <<
five <<"\n";
}

double five_angles::square() const {
    double s=0;
    s=(one.x*two.y+two.x*three.y+three.x*four.y+four.x*five.y+five.x*one.y-
two.x*one.y-
            three.x*two.y-four.x*three.y-five.x*four.y-one.x*five.y)/2;
    if(s<0){
        return -s;
    }else {
        return s;
    }
}
```

six_angles.h:
```
#ifndef D_SIX_ANGLES_H_
#define D_SIX_ANGLES_H_

#include <iostream>
#include "figure.h"


struct six_angles : figure{

    six_angles(std::istream &is);

    point center() const override;
    void print(std::ostream &os) const override;
    void help_print(std::ostream &os) const override;
    double square() const override;
private:
    point one,two,three,four,five,six;
};

#endif
```

six_angles.cpp:
```
#include <iostream>

#include "six_angles.h"

six_angles::six_angles(std::istream &is){
    is >> one >> two >> three >> four >> five >>six;
}

point six_angles::center() const {
    point p;
    p=one+two+three+four+five+six;
    p/=6;
    return p;
}

void six_angles::print(std::ostream &os) const {
    os << one << " " << two << " " << three << " " << four << " " << five <<
" " << six <<"\n";
}
void six_angles::help_print(std::ostream &os) const {
    os <<"2 " << one << " " << two << " " << three << " " << four << " " <<
five << " " << six <<"\n";
}

double six_angles::square() const {
    double s=0;

s=(one.x*two.y+two.x*three.y+three.x*four.y+four.x*five.y+five.x*six.y+six.x*
one.y-two.x*one.y-
       three.x*two.y-four.x*three.y-five.x*four.y-six.x*five.y-
one.x*six.y)/2;
    if(s<0){
        return -s;
    }else {
        return s;
    }
}
```

eight_angles.h:
```
#ifndef D_EIGHT_ANGLES_H_
#define D_EIGHT_ANGLES_H_

#include <iostream>
#include "figure.h"

struct eight_angles : figure{

    eight_angles(std::istream &is);

    point center() const override;
    void print(std::ostream &os) const override;
    void help_print(std::ostream &os) const override;
    double square() const override;
private:
    point one,two,three,four,five,six,seven,eight;
};

#endif
```

eight_angles.cpp:
```
#include <iostream>

#include "eight_angles.h"

eight_angles::eight_angles(std::istream &is){
    is >> one >> two >> three >> four >> five >>six >>seven >>eight;
}

point eight_angles::center() const {
    point p;
    p=one+two+three+four+five+six+seven+eight;
    p/=8;
    return p;
}

void eight_angles::print(std::ostream &os) const {
    os << one << " " << two << " " << three << " " << four << " " << five <<
" " << six << " " << seven
    << " " << eight<<"\n";
}
void eight_angles::help_print(std::ostream &os) const {
    os <<"3 " << one << " " << two << " " << three << " " << four << " " <<
five << " " << six << " " << seven
        << " " << eight<<"\n";
}

double eight_angles::square() const {
    double s=0;

s=(one.x*two.y+two.x*three.y+three.x*four.y+four.x*five.y+five.x*six.y+six.x*
seven.y+seven.x*eight.y+
            eight.x*one.y-two.x*one.y-three.x*two.y-four.x*three.y-
five.x*four.y-six.x*five.y-seven.x*six.y
            -eight.x*seven.y-one.x*eight.y)/2;
    if(s<0){
        return -s;
    }else {
        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>



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 {
    for (size_t i = 0; i < figures_.size(); ++i) {
        figures_[i]->help_print(os);
    }
}
```

```
void document::load(std::istream& is){
    int help;
    while(is >> help){
        if(help ==1){
            five_angles fig(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(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(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){
    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){
    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 {
    if(figures_.size()>0) {
        for (size_t i = 0; i < figures_.size(); ++i) {
            os << "figure " << i << "\n";
            os << "cordinats ";
            figures_[i]->print(os);
            os << "center " << figures_[i]->center() << "\n";
            os << "square " << figures_[i]->square() << "\n";
        }
    }
}

void document::undo() {
    if(operations_.size()>0) {
        operations_[operations_.size()-1]->undo(*this);
        operations_.erase(operations_.begin()+operations_.size()-1);
    }
}

void document::add_command::undo(document &doc) {
    doc.figures_.erase(doc.figures_.begin() + id);
}

void document::remove_command::undo(document &doc) {
    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::cin >>figures;
   size_t id;
   std::cin >> id;
   if(figures == "five_angles"){
       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 == "six_angles"){
       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 == "eight_angles"){
       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>
```

```cpp
#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"

int main() {
    std::string command;
    factory fact;
    std::unique_ptr<document> doc1;
    while(std::cin >> command){
        if(command=="new"){
            doc1=std::make_unique<document>();
        }else if(command=="save"){
            std::string path;
            std::cin >> path;
            std::ofstream os(path);
            doc1->save(os);
            os.close();
        }else if(command=="load"){
            std::string path;
            std::cin >> path;
            std::ifstream is(path);
            if(is) {
                doc1->load(is);
            }else {
                std::cout << "No such file\n";
            }
            is.close();
        }else if(command=="add"){
            fact.construct(doc1);
        }else if(command=="remove"){
            size_t id;
            std::cin >> id;
            doc1->remove_figure(id);
        }else if(command=="show"){
            doc1->show(std::cout);
        }else if(command == "undo"){
            doc1->undo();
        }
    }
    return 0;
}
```

2. **Ссылка на репозиторий в GitHub:**
https://github.com/Suvorova-Sofya/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 команду , 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.