

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ)**

ЛАБОРАТОРНАЯ РАБОТА №4

**по курсу объектно-ориентированное программирование I семестр, 2021/22
уч. год**

Студент: Макаров Глеб Александрович, группа М8О-207Б-20

Преподаватель: Дорохов Евгений Павлович

Условие

Задание: Стэк (Пятиугольник). Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (колонка фигура 1), согласно вариантам задания. Классы должны удовлетворять следующим правилам:

1. Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.
2. Классы фигур должны содержать набор следующих методов:
 - Перегруженный оператор ввода координат вершин фигуры из потока `std::istream` (`>>`). Он должен заменить конструктор, принимающий координаты вершин из стандартного потока.
 - Перегруженный оператор вывода в поток `std::ostream` (`<<`), заменяющий метод `Print` из лабораторной работы 1.
 - Оператор копирования (`=`)
 - Оператор сравнения с такими же фигурами (`==`)
3. Класс-контейнер должен содеержать объекты фигур “по значению” (не по ссылке).

Нельзя использовать:

- Стандартные контейнеры `std`.
- Шаблоны (`template`).
- Различные варианты умных указателей (`shared_ptr`, `weak_ptr`).

Программа должна позволять:

- Вводить произвольное количество фигур и добавлять их в контейнер.
- Распечатывать содержимое контейнера.
- Удалять фигуры из контейнера.

Описание программы

Исходный код лежит в 8 файлах:

1. `main.cpp`: тестирование кода
2. `figure.h`: родительский класс-интерфейс для фигур

3. point.h: описание класса точки
4. point.cpp: реализация класса точки
5. pentagon.h: описание класса пятиугольника, наследующегося от figure
6. pentagon.cpp: реализация класса пятиугольника
7. tstack.h: структура стэка
8. tstack.cpp: реализация стэка

Дневник отладки

Ошибок не наблюдалось.

Недочёты

Недочётов не заметил.

Вывод

В данной лабораторной работе была написана классическая структура данных - стэк. Сложностей не возникло, так как много раз подобное писалось на языке Си на первом курсе. Разница только в том, что в стэке теперь хранится объект класса.

Исходный код

main.cpp

```
#include "pentagon.h"
#include "tstack.h"

void menu(){
    std::cout << "Select an action" << std::endl;
    std::cout << "1) Print the top element" << std::endl;
    std::cout << "2) Remove an item from the stack" << std::endl;
    std::cout << "3) Print items from the stack" << std::endl;
    std::cout << "4) Clear the stack" << std::endl;
    std::cout << "5) Add an item to the stack" << std::endl;
    std::cout << "6) Print the stack length" << std::endl;
    std::cout << "7) Is the stack empty?" << std::endl;
    std::cout << "8) End the program" << std::endl;
}

int main() {
    std::cout.setf(std::ios_base::boolalpha);
    TStack s = TStack();
    char k = 'y';
    menu();
    std::cin >> k;
    while (k != EOF) {
        switch (k) {
            case '1': {
                if(!s.Empty()) std::cout << s.Top() << std::endl;
                else std::cout << std::endl;
                break;
            }
            case '2': s.Pop();
                break;
            case '3': std::cout << s << std::endl;
                break;
            case '4': s.Clear();
                break;
            case '5': s.Push(Pentagon(std::cin));
                break;
            case '6':std::cout << s.Length() << std::endl;
                break;
            case '7':std::cout << (bool)s.Empty() << std::endl;
                break;
            case '8': {
```

```

        std::cout << "Have a nice day!" << std::endl;
        return 0;
    }
    default: std::cout << "Input error! Enter a number from the suggested menu!" <<
std::endl;
        break;
    }
    menu();
    std::cin >> k;
}
return 0;
}

```

figure.h

```

#ifndef MAI_OOP_FIGURE_H
#define MAI_OOP_FIGURE_H
#include "point.h"

class Figure {
public:
    virtual size_t VertexesNumber() = 0;
    virtual double Area() = 0;
    virtual void Print(std::ostream &os) = 0;

};

#endif //MAI_OOP_FIGURE_H

```

point.h

```

#ifndef POINT_H
#define POINT_H
#include <iostream>

class Point {
public:

```

```

    Point();
    Point(std::istream &is);
    Point(double x, double y);

    double dist(const Point &other);

    double get_x();
    double get_y();

    friend std::istream &operator>>(std::istream &is, Point &p);
    friend std::ostream &operator<<(std::ostream &os, Point &p);
    bool operator==(const Point &p);
    Point &operator=(const Point &p);

private:
    double x_;
    double y_;
};

#endif // POINT_H

```

point.cpp

```
#include "point.h"
```

```
#include <cmath>
```

```
Point::Point() : x_(0.0), y_(0.0) {}
```

```
Point::Point(double x, double y) : x_(x), y_(y) {}
```

```
Point::Point(std::istream &is) {
```

```
    is >> x_ >> y_;
```

```
}
```

```
double Point::get_x() {
```

```
    return x_;
```

```
}
```

```
double Point::get_y() {
```

```
    return y_;
```

```
}
```

```
double Point::dist(const Point &other) {
```

```
    double dx = (other.x_ - x_);
```

```
    double dy = (other.y_ - y_);
```

```
    return std::sqrt(dx * dx + dy * dy);
```

```
}
```

```
std::istream &operator>>(std::istream &is, Point &p) {
```

```
    is >> p.x_ >> p.y_;
```

```
    return is;
```

```
}
```

```
std::ostream &operator<<(std::ostream &os, Point &p) {  
    os << "(" << p.x_ << ", " << p.y_ << ")";  
    return os;  
}
```

```
bool Point::operator==(const Point &p) {  
    if (this->x_ == p.x_ && this->y_ == p.y_) {  
        return true;  
    } else return false;  
}
```

```
Point &Point::operator=(const Point &p) {  
    if (this == &p) {  
        return *this;  
    }  
    this->x_ = p.x_;  
    this->y_ = p.y_;  
    return *this;  
}
```

pentagon.h

```
#ifndef MAI_OOP_PENTAGON_H  
#define MAI_OOP_PENTAGON_H  
#include "figure.h"
```

```
class Pentagon {  
private:  
    Point a_, b_, c_, d_, e_;  
public:
```



```
Pentagon();  
Pentagon(const Pentagon &pentagon);  
Pentagon(std::istream &is);  
size_t VertexesNumber();  
double Area();  
void Print(std::ostream &os);  
friend std::istream &operator>>(std::istream &is, Pentagon &object);  
friend std::ostream &operator<<(std::ostream &os, Pentagon &object);  
Pentagon &operator=(const Pentagon &object);  
bool operator==(const Pentagon &object);  
  
};  
#endif //MAI_OOP_PENTAGON_H
```

pentagon.cpp

```
#include "pentagon.h"
```

```
#include <math.h>
```

```
Pentagon::Pentagon() : a_(0, 0), b_(0, 0), c_(0, 0), d_(0, 0), e_(0, 0) {}
```

```
Pentagon::Pentagon(const Pentagon &pentagon) {
```

```
    this->a_ = pentagon.a_;
```

```
    this->b_ = pentagon.b_;
```

```
    this->c_ = pentagon.c_;
```

```
    this->d_ = pentagon.c_;
```

```
    this->e_ = pentagon.c_;
```

```
}
```

```
Pentagon::Pentagon(std::istream &is) {
```

```
    std::cin >> a_ >> b_ >> c_ >> d_ >> e_;
```

```
}
```

```
size_t Pentagon::VertexesNumber() {
```

```
    return (size_t) 5;
```

```
}
```

```
double Pentagon::Area() {
```

```
    double p = fabs(a_.get_x()*b_.get_y()-b_.get_x()*a_.get_y()+b_.get_x()*c_.get_y()-  
c_.get_x()*b_.get_x()+c_.get_x()*d_.get_y()-d_.get_x()*c_.get_y()+d_.get_x()*e_.get_y()-  
e_.get_x()*d_.get_y()+e_.get_x()*a_.get_y()-a_.get_x()*e_.get_y())/2;
```

```
    return p;
```

```
}
```

```
void Pentagon::Print(std::ostream &os) {
```

```
    std::cout << "Pentagon " << a_ << b_ << c_ << d_ << e_ << std::endl;
```

```
}
```

```

std::istream &operator>>(std::istream &is, Pentagon &object) {
    is >> object.a_ >> object.b_ >> object.c_ >> object.d_ >> object.e_;
    return is;
}

```

```

std::ostream &operator<<(std::ostream &os, Pentagon &object) {
    os << "a side = " << object.a_.dist(object.b_) << std::endl;
    os << "b side = " << object.b_.dist(object.c_) << std::endl;
    os << "c side = " << object.c_.dist(object.d_) << std::endl;
    os << "d side = " << object.d_.dist(object.e_) << std::endl;
    os << "e side = " << object.e_.dist(object.a_) << std::endl;
    return os;
}

```

```

Pentagon &Pentagon::operator=(const Pentagon &object) {
    this->a_ = object.a_;
    this->b_ = object.b_;
    this->c_ = object.c_;
    this->d_ = object.d_;
    this->e_ = object.e_;
    return *this;
}

```

```

bool Pentagon::operator==(const Pentagon &object) {
    if (this->a_ == object.a_ && this->b_ == object.b_ && this->c_ == object.c_ && this->d_ ==
    object.d_ && this->e_ == object.e_) {
        return true;
    } else return false;
}

```

tstack.h

```

#ifndef MAI_OOP_TSTACK_H
#define MAI_OOP_TSTACK_H
#include "pentagon.h"

class TStack {
private:
    struct QueueItem {
        Pentagon data;
        QueueItem *next;
    };
    size_t size;
    QueueItem *top_;

public:
    TStack();
    TStack(const TStack &queue);
    size_t Length();
    bool Empty();
    Pentagon &Top();
    void Push(const Pentagon &t);
    void Pop();
    void Clear();
    friend std::istream &operator>>(std::istream &is, TStack &object);
    friend std::ostream &operator<<(std::ostream &os, const TStack &object);

    virtual ~TStack();
};

#endif

```

tstack.cpp

```
#include "tstack.h"
```

```
TStack::TStack() {  
    top_ = new QueueItem;  
    top_->next = nullptr;  
    size = 0;  
}
```

```
TStack::TStack(const TStack &stack) {  
    QueueItem *top = stack.top_;  
  
    while (top->next != nullptr) {  
        top_->data = top->data;  
        QueueItem *item1 = new QueueItem;  
        item1->next = nullptr;  
        top_->next = item1;  
        top = top->next;  
    }  
    size = stack.size;  
}
```

```
size_t TStack::Length() {  
    return (size_t) size;  
}
```

```
bool TStack::Empty() {  
    return (size == 0);  
}
```

```
Pentagon &TStack::Top() {
    return top_->data;
}
```

```
void TStack::Push(const Pentagon &t) {
    QueueItem *item = new QueueItem;
    item->data = t;
    item->next = top_;
    top_ = item;
    size++;
}
```

```
void TStack::Pop() {
    if (size == 0) {
        std::cout << "Unable to perform pop! The queue is empty!" << std::endl;
        return;
    }
    QueueItem *item = top_;
    top_ = top_->next;

    delete item;
    size--;
}
```

```
std::istream &operator>>(std::istream &is, TStack &object) {
    Pentagon t;
    is >> t;
    object.Push(t);
    return is;
}
```

}

```
std::ostream &operator<<(std::ostream &os, const TStack &object) {
```

```
    TStack::QueueItem *item = object.top_;
```

```
    os << "==" << " ";
```

```
    while (item->next != nullptr) {
```

```
        os << item->data.Area() << " ";
```

```
        item = item->next;
```

```
    }
```

```
    os << "==" << " ";
```

```
    return os;
```

```
}
```

```
void TStack::Clear() {
```

```
    while (top_->next != nullptr) {
```

```
        QueueItem *item = top_;
```

```
        top_ = top_->next;
```

```
        delete item;
```

```
    }
```

```
    size = 0;
```

```
    top_->next = nullptr;
```

```
}
```

```
TStack::~~TStack() {
```

```
    while (top_->next != nullptr) {
```

```
        QueueItem *item = top_;
```

```
        top_ = top_->next;
```

```
        delete item;
```

```
    }
```

```
    size = 0;
```

```
delete top_;  
}
```