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

ЛАБОРАТОРНАЯ РАБОТА №2 по курсу

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

Студент *Васютинский Вадим Александрович, группа М80-208Б-20*
Преподаватель *Дорохов Евгений Павлович*

Цель работы

Целью лабораторной работы является:

- Закрепление навыков работы с классами.
- Создание простых динамических структур данных.
- Работа с объектами, передаваемыми «по значению».

Задание

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:

Требования к классу фигуры аналогичны требованиям из лаб.работы 1.

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 (==)

Класс-контейнер должен соджержать объекты фигур “по значению” (не по ссылке).

Класс-контейнер должен содержать набор следующих методов:

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.

Дневник отладки

Во время выполнения лабораторной работы программа был несколько раз переписан способ хранения прямоугольников в самом векторе, что сказалось, кстати, на следующих лабораторных работах. После нескольких отладок программа стала работать исправно.

Недочёты

Недочётов не было обнаружено.

Выводы

Лабораторная работа №4 - это модернизация последних лабораторных 2 семестра. Если на 1 курсе я реализовывал вектор при помощи структур на языке СИ, то сейчас я реализовал вектор при помощи ООП на языке C++. Лабораторная прошла успешно, я повторил старый материал и узнал, усвоил много нового. Также я освоил работу с выделением и очисткой памяти на языке C++ при помощи команд new и delete.

Исходный код

figure.h

```
#ifndef LAB1_FIGURE_H
#define LAB1_FIGURE_H

#include <cmath>
#include <iostream>
#include "point.h"

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


#endif //LAB1_FIGURE_H
```

main.cpp

```
#include "rectangle.h"
#include "TVector.h"
#include <string>

int main() {

    std::string command;
    TVector v;

    while (std::cin >> command){
        if(command == "print")
            std::cout << v;
        else if(command == "insertlast"){
            Rectangle p;
            std::cin >> p;
            v.InsertLast(p);
        }
        else if(command == "removelast"){
            v.RemoveLast();
        }
        else if(command == "last"){
            std::cout << v.Last();
```

```
        }
        else if(command == "idx"){
            int idx;
            std::cin >> idx;
            std::cout << v[idx];
        }
        else if(command == "length"){
            std::cout << v.Length() << std::endl;
        }
        else if(command == "clear"){
            v.Clear();
        }
        else if(command == "empty"){
            if(v.Empty()) std::cout << "Yes" << std::endl;
            else std::cout << "No" << std::endl;
        }
    }
}
```

rectangle.cpp

```
#include "rectangle.h"

std::istream& operator>>(std::istream& is, Rectangle& r) {
    std::cout << "Enter data: " << std::endl;
    is >> r.a >> r.b >> r.c >> r.d;
    return is;
}

std::ostream& operator<<(std::ostream& os, Rectangle& r) {
    os << "Pentagon: " << r.a << r.b << r.c << r.d;
    return os;
}

Rectangle& Rectangle::operator=(const Rectangle &other) {
    this->a = other.a;
    this->b = other.b;
    this->c = other.c;
    this->d = other.d;
    return *this;
}

bool Rectangle::operator==(const Rectangle &other) {
    return a == other.a && b == other.b && c == other.c;
}
```

```
void Rectangle::Print(std::ostream &os) {
    os << "Rectangle: " << a << b << c << d << std::endl;
}

size_t Rectangle::VertexesNumber() {
    return 4;
}

double Rectangle::Area() {
    return a.dist(b) * a.dist(d);
}


Rectangle::Rectangle() {}

Rectangle::Rectangle(Point a, Point b, Point c, Point d) : a(a), b(b), c(c), d(d) {}

Rectangle::Rectangle(std::istream &is) {
    std::cout << "Enter data:" << std::endl;
    is >> a >> b >> c >> d;
    std::cout << "Rectangle created via istream" << std::endl;
}

Rectangle::Rectangle(const Rectangle &other) : Rectangle(other.a, other.b,
other.c, other.d) {
    std::cout << "Made copy of rectangle" << std::endl;
}

Rectangle::~Rectangle() {
    std::cout << "Rectangle deleted" << std::endl;
}
```

Rectangle.h

```
#ifndef OOP1_RECTANGLE_H
#define OOP1_RECTANGLE_H

#include "figure.h"
class Rectangle : Figure {
public:
    friend std::istream& operator>>(std::istream& is, Rectangle& p);
    friend std::ostream& operator<<(std::ostream& os, Rectangle& p);
    bool operator==(const Rectangle& other);
    Rectangle& operator=(const Rectangle& other);
    void Print(std::ostream &os) override;
```

```
    size_t VertexesNumber() override;
    double Area() override;
    Rectangle();
    Rectangle(Point a, Point b, Point c, Point d);
    Rectangle(std::istream &is);
    Rectangle(const Rectangle &other);
    virtual ~Rectangle();
private:
    Point a, b, c, d;
};
#endif //OOP1_RECTANGLE_H
```

Point.cpp

```
#include "point.h"

#include <cmath>

bool Point::operator==(const Point &other) {
    return (this->x_ == other.x_ && this->y_ == other.y_);
}

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::dist(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;
}
```

Point.h

```
#ifndef LAB1_POINT_H
#define LAB1_POINT_H
```

```
#include <iostream>

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

    double dist(Point& other);

    friend std::istream& operator>>(std::istream& is, Point& p);
    friend std::ostream& operator<<(std::ostream& os, Point& p);
    bool operator==(const Point& other);
private:
    double x_;
    double y_;
};

#endif //LAB1_POINT_H
```

TVector.cpp

```
#include "TVector.h"
#include <cassert>

TVector::TVector():size(0), data(nullptr), capacity(0) {
}

TVector::TVector(const TVector& other){
    size = other.size;
    capacity = other.capacity;
    data = new TVectorItem[capacity];
    for(int i = 0; i < size; ++i)
        data[i] = other.data[i];
}

TVector::~TVector() {
    if(capacity != 0)
      delete[] data;
}

void TVector::InsertLast(const Rectangle& rectangle){
    if(capacity != 0 && capacity > size){
        data[size++] = rectangle;
    }
    else{
```

```
        if(capacity == 0)
            capacity = 1;
        capacity *= 2;
        TVectorItem* data_new = new TVectorItem[capacity];
        for(int i = 0; i < size; ++i){
            data_new[i] = data[i];
        }
        data_new[size++] = rectangle;
        delete[] data;
        data = data_new;
    }
}

void TVector::RemoveLast(){
    if(size > 0)
        --size;
}

Rectangle& TVector::Last(){
    assert(size > 0);
    return data[size - 1].GetRectangle();
}

size_t TVector::Length() {
    return size;
}

Rectangle& TVector::operator[] (const size_t idx){
    assert(idx >= 0 && idx < size);
    return data[idx].GetRectangle();
}

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

void TVector::Clear() {
    if(capacity != 0)
        delete[] data;
    data = nullptr;
    capacity = size = 0;
}

std::ostream& operator<<(std::ostream& os, const TVector& arr){
    for(int i = 0; i < arr.size; ++i){
        os << arr.data[i].GetRectangle();
    }
    return os;
}
```

TVectorItem.cpp

```
#include <iostream>
#include "TVectorItem.h"

TVectorItem::TVectorItem(const Rectangle& pentagon){
    p = pentagon;
}

TVectorItem::TVectorItem(const TVectorItem& other){
    p = other.p;
}

Rectangle& TVectorItem::GetRectangle() {
    return p;
}

std::ostream &operator<<(std::ostream &os, TVectorItem &p){
    os << p;
    return os;
}
```

TVectorItem.h

```
#ifndef LAB1_TVECTORITEM_H
#define LAB1_TVECTORITEM_H

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

class TVectorItem {
public:
    TVectorItem(const Rectangle& rectangle);

    TVectorItem(const TVectorItem& other);

    Rectangle& GetRectangle();

    TVectorItem(){}

    friend std::ostream &operator<<(std::ostream &os, TVectorItem &p);


private:
```

```
    Rectangle p;
};

#endif //LAB1_TVECTORITEM_H
```