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 *Зинин Владислав Владимирович, 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:

TODO: по поводу методов в личку

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 10 файлах:

1. main.cpp - основная программа, взаимодействие с пользователем посредством команд из меню

2. include/figure.h - описание абстрактного класса фигур

3. include/point.h - описание класса точки

4. include/TVector.cpp - реализация функций контейнера первого уровня (в моем случае вектора)

5. include/TVector.h – реализация класса контейнера первого уровня (в моем случае вектора)

6. include/rhombus.h - описание класса ромба, наследующегося от figures

7. include/point.cpp - реализация класса точки

8. include/TVectorItem.cpp – реализация функций вспомогательного класса для контейнера

9. include/TVectorItem.h – описание вспомогательного класса для контейнера

10. include/rhombus.cpp: реализация класса ромба, наследующегося от figure

**Дневник отладки**

Во время выполнения лабораторной работы программа была несколько раз отлажена, так как плохо работали некоторые функции вектора. После нескольких отладок программа стала работать исправно.

**Недочёты**  
Недочётов не было обнаружено.

**Выводы**

Лабораторная работа №4 -э то модернизация последних лабораторных 2 семестра. Если на 1 курсе я реализовывал бинарное дерево при помощи структур на языке СИ, то сейчас я реализовал бинарное дерево при помощи ООП на языке С++. Лабораторная прошла успешно, я повторил старый материал и узнал, усвоил много нового.

**Исходный код**

**figure.h**

|  |
| --- |
| #ifndef FIGURE\_H |
|  |

|  |
| --- |
| #define FIGURE\_H |
|  |

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "point.h" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class Figure |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| virtual ~Figure(){}; |
|  |

|  |
| --- |
| virtual double Area() = 0; |
|  |

|  |
| --- |
| virtual void Print(std::ostream &os) = 0; |
|  |

|  |
| --- |
| virtual size\_t VertexesNumber() = 0; |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

#endif //FIGURE\_H

**point.h**

|  |
| --- |
| #ifndef POINT\_H |
|  |

|  |
| --- |
| #define POINT\_H |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class Point |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| Point(); |
|  |

|  |
| --- |
| Point(double x, double y); |
|  |

|  |
| --- |
| Point(std::istream &is); |
|  |

|  |
| --- |
| double dist(Point &other); |
|  |

|  |
| --- |
| friend double get\_x(Point &other); |
|  |

|  |
| --- |
| friend double get\_y(Point &other); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| friend std::istream& operator>>(std::istream& is, Point& p); |
|  |

|  |
| --- |
| friend std::ostream& operator<<(std::ostream& os, Point& p); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private: |
|  |

|  |
| --- |
| double x\_, y\_; |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

#endif //POINT\_H

**point.cpp**

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "point.h" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 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\_; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 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; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| double get\_x(Point &other) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return other.x\_; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| double get\_y(Point &other) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return other.y\_; |

**rhombus.h**

|  |
| --- |
| #ifndef RHOMBUX\_H |
|  |

|  |
| --- |
| #define RHOMBUX\_H |
|  |

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "point.h" |
|  |

|  |
| --- |
| #include "figure.h" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class Rhombus : public Figure |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| Rhombus(std::istream &is); |
|  |

|  |
| --- |
| double Area(); |
|  |

|  |
| --- |
| void Print(std::ostream &os); |
|  |

|  |
| --- |
| size\_t VertexesNumber(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| virtual ~Rhombus(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private: |
|  |

|  |
| --- |
| Point a, b, c, d; |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

#endif //RHOMBUX\_H

**rhombus.cpp**

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "rhombus.h" |
|  |

|  |
| --- |
| #include <math.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Rhombus::Rhombus(std::istream &is) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| is >> a; |
|  |

|  |
| --- |
| is >> b; |
|  |

|  |
| --- |
| is >> c; |
|  |

|  |
| --- |
| is >> d; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| void Rhombus::Print(std::ostream &os) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| os << "Rhombus" << std::endl; |
|  |

|  |
| --- |
| os << a << ',' << b << ',' << c << ',' << d << std::endl; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| double Rhombus::Area() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return 0.5 \* fabs(get\_x(a)\*get\_y(b) + get\_x(b)\*get\_y(c) + get\_x(c)\*get\_y(d) + get\_x(d)\*get\_y(a) - get\_x(b)\*get\_y(a) - get\_x(c)\*get\_y(b) - get\_x(d)\*get\_y(c) - get\_x(a)\*get\_y(d)); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Rhombus::~Rhombus() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| std::cout << "Rhombus deleted" << std::endl; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| size\_t Rhombus::VertexesNumber() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return 4; |
|  |

}

**Main.cpp**

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "TVector.h" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| TVector list; |
|  |

|  |
| --- |
| /\*-----Test InsertLast---\*/ |
|  |

|  |
| --- |
| list.InsertLast(Rhombus()); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus(Point(1,2), Point(3,4), Point(5,6), Point(7,8))); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus(Point(1,2), Point(3,4), Point(5,4), Point(7,8))); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus(Point(1,0), Point(3,2), Point(4,5), Point(9,9))); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus(Point(1,0), Point(3,2), Point(4,5), Point(9,9))); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus(Point(1,0), Point(3,2), Point(4,5), Point(9,9))); |
|  |

|  |
| --- |
| list.InsertLast(Rhombus()); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| /\*-----Test RemoveLast---\*/ |
|  |

|  |
| --- |
| list.RemoveLast(); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| list.RemoveLast(); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| /\*-----Test push\_back---\*/ |
|  |

|  |
| --- |
| // list.push\_front(Rhombus(Point(2,3), Point(2,3), Point(2,3), Point(2,3))); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| // /\*-----Test pop\_back---\*/ |
|  |

|  |
| --- |
| // list.pop\_front(); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| /\*-----Test Remove---\*/ |
|  |

|  |
| --- |
| // list.pop\_back(); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // list.push\_front(Rhombus(Point(2,3), Point(2,3), Point(2,3), Point(2,3))); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| // /\*-----Test insert---\*/ |
|  |

|  |
| --- |
| // list.insert(Rhombus(Point(0,1), Point(2,3), Point(4,5), Point(6,7)), 1); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| // list.insert(Rhombus(Point(0,1), Point(2,3), Point(4,5), Point(6,7)), 3); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| // list.insert(Rhombus(Point(0,1), Point(2,3), Point(4,5), Point(6,7)), 2); |
|  |

|  |
| --- |
| // std::cout << list << std::endl; |
|  |

|  |
| --- |
| /\*-----Test Remove---\*/ |
|  |

|  |
| --- |
| std::cout << "--------------" << std::endl; |
|  |

|  |
| --- |
| list.Remove(1); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| std::cout << list.Length() << std::endl; |
|  |

|  |
| --- |
| std::cout << "--------------" << std::endl; |
|  |

|  |
| --- |
| list.Resize(2); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| std::cout << "--------------" << std::endl; |
|  |

|  |
| --- |
| std::cout << list.Length() << std::endl; |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| std::cout << list[2] << std::endl; |
|  |

|  |
| --- |
| list.Resize(4); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| list.Resize(4); |
|  |

|  |
| --- |
| std::cout << list << std::endl; |
|  |

|  |
| --- |
| return 0; |
|  |

}