**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 4**

Тема: Основы метапрограммирования

Студент: Бутырев Даниил Вячеславович

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

[https://github.com/SteepturN/oop\_exercise\_0](https://github.com/SteepturN/oop_exercise_01)4

1. Постановка задачи

связный список + треугольник

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (одна из фигур ЛР3 на выбор). Вариант структуры данных для контейнера выбрать из документа “Варианты структур данных” (контейнер 1-го уровня) согласно своему номеру из Варианты ЛР4..

Классы должны удовлетворять следующим правилам:

· Требования к классу фигуры аналогичны требованиям из лабораторной работы 3

· Классы фигур должны иметь переопределенный оператор вывода в поток std::ostream (<<). Оператор должен распечатывать параметры фигуры (тип фигуры, длины сторон, радиус и т.д).

· Классы фигур должны иметь переопределенный оператор ввода фигуры из потока std::istream (>>). Оператор должен вводить основные параметры фигуры (длины сторон, радиус и т.д).

· Классы фигур должны иметь операторы копирования (=).

· Классы фигур должны иметь операторы сравнения с такими же фигурами (==).

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

· Класс-контейнер должен иметь метод по добавлению фигуры в контейнер.

· Класс-контейнер должен иметь методы по получению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь метод по удалению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).

· Класс-контейнер должен иметь деструктор, удаляющий все элементы контейнера.

· Классы должны быть расположены в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Программа должна позволять:

· Вводить произвольное количество фигур и добавлять их в контейнер.

· Распечатывать содержимое контейнера.

· Удалять фигуры из контейнера.

1. Описание программы

- обработка ввода в main

- переопределённые операторы вывода и ааода

- контейнер - однонаправленный список с forward итератором

1. Набор тестов

1)

p 0 1 2 3 4 5 6

p 0 1 2 3 4 5 6

print

2)

p 0 1 2 3 4 5 6

p 1 3 2 4 4 5 6

p 0 1 2 3 4 5 6

p 3 0 9 1 2 3 9

print

d 3

print

1. Результаты выполнения тестов

You can use

--put triangle: p [(point) 3 times]

--print container: print

--delete (figure number): d (number of figure)

--exit

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

print

You can use

--put triangle: p [(point) 3 times]

--print container: print

--delete (figure number): d (number of figure)

--exit

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

triangle

sides' length: 3.60555 2.23607 2.23607

coordinates: (3, 2)(4, 4)(5, 6)

triangle

sides' length: 9 7.07107 7.28011

coordinates: (0, 9)(1, 2)(3, 9)

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

triangle

sides' length: 2.23607 2.82843 2.82843

coordinates: (1, 2)(3, 4)(5, 6)

triangle

sides' length: 3.60555 2.23607 2.23607

coordinates: (3, 2)(4, 4)(5, 6)

1. Листинг программы

#include "List.h"

#include "ListEl.h"

#include "Iterator.h"

Iterator::Iterator()

: list(NULL), el(NULL) {}

Iterator& Iterator::operator++() {

el = &((\*el)->next);

return \*this;

}

Iterator::Iterator(const List\* list, ListEl\*\* el)

: list(list), el(el) {}

bool Iterator::operator!=(Iterator iter) {

return ((list != iter.list) || (el != iter.el));

}

Iterator::Iterator(List\* l)

: list(l), el(l->begin().el) {}

Triangle& Iterator::operator\*() {

return (\*(el))->t;

}

bool Iterator::operator==(Iterator iter) {

return !(\*this != iter);

}

#ifndef ITERATOR\_H\_

#define ITERATOR\_H\_

#include "ListEl.h"

#include "List.h"

#include "Triangle.h"

class Iterator {

public:

const List\* list;

ListEl\*\* el;

Iterator();

Iterator(List\*);

Iterator(const List\*, ListEl\*\*);

Iterator& operator++();

Triangle& operator\*();

bool operator!=(Iterator iter);

bool operator==(Iterator);

};

#endif // ITERATOR\_H\_

#include "Triangle.h"

#include "List.h"

#include "ListEl.h"

#include "Iterator.h"

List::List()

: head(NULL), size(0), tail(NULL) {}

List::~List() {

/\*for(auto cur = this->begin(); cur != this->end();) {

auto del = cur; //Error because when I ++cur I still need to use

//pointer to the next ListEL which is contained in

//del, and pointer-attribute of cur points to del's attribute

++cur;

std::cout << (\*(del.el))->t;

delete(\*(del.el));

}\*/

for(auto cur = head; cur != NULL;) {

auto del = cur;

cur=cur->next;

delete(del);

}

}

int List::put(Triangle& t, Iterator i) { //inserts before

ListEl\* new\_list\_element = new ListEl(\*(i.el), t);

\*(i.el) = new\_list\_element;

++size;

if(new\_list\_element->next == NULL)

tail = &(new\_list\_element->next);

return 0;

}

void List::remove(Iterator i) {

auto del\_this = \*(i.el);

\*(i.el) = (\*(i.el))->next;

if(\*(i.el) == NULL)

tail = i.el;

--size;

delete(del\_this);

}

Triangle& List::get(Iterator i) {

return \*i;

}

Iterator List::begin() {

return Iterator(this, &head);

}

Iterator List::end() {

return Iterator(this, tail);

}

std::ostream& operator<<(std::ostream& cout, List& list) {

for(auto i = list.begin(); i != list.end(); ++i) {

cout << \*i << '\n';

}

return cout;

}

#ifndef LIST\_H

#define LIST\_H

#include "ListEl.h"

#include "Triangle.h"

class Iterator;

class List {

public:

List();

~List();

Iterator begin();

int put(Triangle& t);

int put(Triangle& t, Iterator i);

Triangle& get(Iterator p);

void remove(Iterator p);

Iterator end();

friend std::ostream& operator<<(std::ostream&, List&);

ListEl\* head;

unsigned int size;

ListEl\*\* tail;

};

#endif // LIST\_H

#include "ListEl.h"

#include "Triangle.h"

ListEl::ListEl(ListEl\* next, Triangle& t)

: next(next), t(t) {}

ListEl::ListEl(ListEl\* next, Triangle&& t)

: next(next), t(t) {}

#ifndef LISTEL\_H\_

#define LISTEL\_H\_

#include "Triangle.h"

class ListEl {

public:

ListEl(ListEl\* next, Triangle& t);

ListEl(ListEl\* next, Triangle&& t);

ListEl\* next;

Triangle t;

};

#endif // LISTEL\_H\_

#include <iostream>

#include <string>

#include <cstdio>

#include <set>

#include "Read\_input.hpp"

#include "Triangle.h"

#include "List.h"

read\_return\_t get\_command(std::set<std::string>& valid\_commands, char\* command)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> command;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

if(valid\_commands.count(command)) return VALID\_INPUT;

else return INVALID\_INPUT;

}

template <typename T>

read\_return\_t get\_value(T& d)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned int& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned long long& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

#include "Read\_input.tpp"

#ifndef READ\_INPUT\_H\_

#define READ\_INPUT\_H\_

#include <set>

#include <string>

typedef enum {

END\_OF\_FILE,

VALID\_INPUT,

INVALID\_INPUT,

END\_OF\_LINE,

} read\_return\_t;

read\_return\_t get\_command(std::set<std::string>&, char\*);

template <typename T>

read\_return\_t get\_value(T&);

template <>

read\_return\_t get\_value (unsigned int&);

template <>

read\_return\_t get\_value (unsigned long long&);

#endif // READ\_INPUT\_H\_

#pragma once

template read\_return\_t get\_value (Triangle& d);

#include <vector>

#include <utility>

#include <iostream>

#include <cmath>

#include "Triangle.h"

Triangle::Triangle(Triangle& t)

: verteces(t.verteces) {}

Triangle::Triangle(Triangle&& t)

: verteces(t.verteces) {}

Triangle::Triangle()

: verteces(std::vector<std::pair<double, double>>(3)) {}

Triangle& Triangle::operator=(const Triangle& t) {

for(int i = 0, j = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; i++, j++) {

verteces[i] = t.verteces[i];

}

return \*this;

}

bool Triangle::operator==(const Triangle& t) {

bool result = true;

for(int i = 0, j = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; i++, j++) {

if(verteces[i] != t.verteces[i]) result = false;

}

return result;

}

std::ostream& operator<<(std::ostream& cout, Triangle& t) {

cout << "triangle\nsides' length:\t";

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i) {

auto v1 = t.verteces[i-1<0?NUM\_OF\_VERTECES\_OF\_TRIANGLE:i-1];

auto v2 = t.verteces[i];

cout << sqrt(pow(v1.first-v2.first,2)+pow(v1.second-v2.second, 2)) << ' ';

}

cout << '\n' << "coordinates: ";

t.coordinates();

return cout;

}

std::istream& operator>>(std::istream& cin, Triangle& t) {

char ch(' ');

Triangle copy;

copy = t;

for(int i = 0, j = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE\*2; ++i, j=(j+1)%2) {

while((ch == '\t') || (ch == ' ') || (ch == '\n')) {

cin >> ch;

if(cin.eof()) {

cin.unget(); //?

cin.setstate(std::ios\_base::failbit);

t = copy;

return cin;

}

}

cin.unget();

ch = ' ';

if(j == 0)

cin >> t.verteces[i/2].first;

else

cin >> t.verteces[i/2].second;

if(cin.fail()) {

t = copy;

return cin;

}

}

return cin;

}

std::pair<double, double> Triangle::center() const noexcept{

std::pair<double, double> \_center(0, 0);

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i) {

\_center.first+=verteces[i].first;

\_center.second+=verteces[i].second;

}

\_center.first/=NUM\_OF\_VERTECES\_OF\_TRIANGLE;

\_center.second/=NUM\_OF\_VERTECES\_OF\_TRIANGLE;

return \_center;

}

void Triangle::coordinates() const noexcept {

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i)

std::cout << '(' << verteces[i].first << ", " << verteces[i].second << ')';

}

double Triangle::area() const noexcept {

double area(0);

for(int i = 0; i< NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i) {

area+= verteces[i].first \* verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_TRIANGLE].second - \

verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_TRIANGLE].first \* verteces[i].second;

}

area/=2;

return area<0?-area:area;

}

#ifndef TRIANGLE\_H\_

#define TRIANGLE\_H\_

#include <vector>

#include <utility>

#include <iostream>

#include <cmath>

#define NUM\_OF\_VERTECES\_OF\_TRIANGLE 3

class Triangle {

public:

std::vector<std::pair<double, double>> verteces;

Triangle(Triangle& t);

Triangle(Triangle&& t);

Triangle();

std::pair<double, double> center() const noexcept;

void coordinates() const noexcept;

double area() const noexcept;

Triangle& operator=(const Triangle& t);

bool operator==(const Triangle& t);

friend std::ostream& operator<<(std::ostream& cout, Triangle& t);

friend std::istream& operator>>(std::istream& cin, Triangle& t);

};

std::ostream& operator<<(std::ostream& cout, Triangle& t);

std::istream& operator>>(std::istream& cin, Triangle& t);

#endif // TRIANGLE\_H\_

/\*

Бутырев М8О-206Б-20

var 12

связный список + треугольник

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (одна из фигур ЛР3 на выбор). Вариант структуры данных для контейнера выбрать из документа “Варианты структур данных” (контейнер 1-го уровня) согласно своему номеру из Варианты ЛР4..

Классы должны удовлетворять следующим правилам:

· Требования к классу фигуры аналогичны требованиям из лабораторной работы 3

· Классы фигур должны иметь переопределенный оператор вывода в поток std::ostream (<<). Оператор должен распечатывать параметры фигуры (тип фигуры, длины сторон, радиус и т.д).

· Классы фигур должны иметь переопределенный оператор ввода фигуры из потока std::istream (>>). Оператор должен вводить основные параметры фигуры (длины сторон, радиус и т.д).

· Классы фигур должны иметь операторы копирования (=).

· Классы фигур должны иметь операторы сравнения с такими же фигурами (==).

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

· Класс-контейнер должен иметь метод по добавлению фигуры в контейнер.

· Класс-контейнер должен иметь методы по получению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь метод по удалению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).

· Класс-контейнер должен иметь деструктор, удаляющий все элементы контейнера.

· Классы должны быть расположены в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Программа должна позволять:

· Вводить произвольное количество фигур и добавлять их в контейнер.

· Распечатывать содержимое контейнера.

· Удалять фигуры из контейнера.

\*/

#include <iostream>

#include "Triangle.h"

#include "Read\_input.hpp"

#include "List.h"

#include "Iterator.h"

#include "ListEl.h"

#include <cstdio>

#include <set>

#include <string>

int main(){ // можно вынести названия команд как константы

std::string help\_message = "You can use\n\

--put triangle: p [(point) 3 times]\n\

--print container: print\n\

--delete (figure number): d (number of figure)\n\

--exit\n";

List list;

char ch(' ');

char command[20];

std::set<std::string> valid\_commands = {"p", "print", "d", "exit"};

//"

std::cout << help\_message;

do {

bool valid\_input = false;

do{

read\_return\_t answer = get\_command(valid\_commands, command);

switch(answer) {

case END\_OF\_FILE: return 0;

case END\_OF\_LINE: continue;

case VALID\_INPUT: valid\_input = true; break;

case INVALID\_INPUT:

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

else break;

}

} while(!valid\_input);

std::string&& command\_string = static\_cast<std::string>(command);

if(command\_string == "exit") return 0;

if(command\_string == "p") {

Triangle triangle;

unsigned int input\_figure\_number = 0;

if(get\_value<unsigned int>(input\_figure\_number) != VALID\_INPUT ||

input\_figure\_number > list.size) { //if there would be EOF

std::cout << "wrong input";

} else {

bool all\_done = false;

Iterator i = list.begin();

while(!all\_done) {

if(input\_figure\_number == 0) {

all\_done = true;

} else {

++i;

--input\_figure\_number;

}

}

if(get\_value<Triangle>(triangle) != VALID\_INPUT)

std::cout << "wrong input";

else {

list.put(triangle, i);

continue;

}

}

} else if(command\_string == "print") {

std::cout << list;

continue;

} else { //delete

unsigned int input\_figure\_number = 0;

if(get\_value<unsigned int>(input\_figure\_number) != VALID\_INPUT ||

input\_figure\_number >= list.size) { //if there would be EOF

std::cout << "wrong input";

} else {

bool all\_done = false;

Iterator i = list.begin();

while(!all\_done) {

if(input\_figure\_number == 0) {

list.remove(i);

all\_done = true;

} else {

++i;

--input\_figure\_number;

}

}

}

}

do ch = getchar(); while((ch != '\n') && (ch != EOF));

std::cout << std::endl;

if(ch == EOF) return 0;

} while(true);

return 0;

}

Вывод: Можно создавать собственные контейнеры и переопределять стандартный вывод для различных объектов.

ЛИТЕРАТУРА

1. Б. Страуструп "Язык программирования С++", четвёртое издание, 2013.