**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:

Студент: Почечура Артемий Андреевич

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

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).

Фигура: ромб (Вариант 12, Лабораторная работа №3).

Структура данных для контейнера: Вариант 19 - N-дерево.

1. **Описание программы**

Реализация поставленной задачи происходит с помощью 7 файлов – трёх заголовочных, трёх формата cpp и сам main. В файлах Romb.h и Romb.cpp описан класс Romb и функции работы с ним: задание, ввод, вывод, нахождение площади и т.д.. В файлах T\_NTreeItem.h и T\_NTreeItem.cpp описаны параметры элемента n-дерева (значение типа Romb, ссылки на первого брата и первого ребёнка) и функции для работы с ним (получение значения первого ребёнка, получение значения первого брата, вывод элемента и т.д.). В файлах T\_NTree.h и T\_NTree.cpp описана структура дерева (ссылка на корень) и функции для работы с ним (добавление/удаление элемента дерева, получение значения элемента, вывод всего дерева). В файле main.cpp реализован интерактивный ввод, в котором происходит добавление элемента в дерево, удаление элемента из дерева, получение значения, вывод всего дерева, в зависимости от команды, которую задаёт пользователь.

1. **Набор тестов и результаты их исполнения**

Тест 1:

a <EL> <PATH> - add element on specified path.

d <PATH> - delete element for specified path.

Example for path: cbcc.

Example for el:

First diagonal(line x): 12

Second diagonal(line y): 15

P - print tree

p <PATH> - print element on specified path

e - end

Centre of all figures - point (0;0)

a 6 8

Element added

a 4 4

Element added

a 10 10 bb

Element added

a 12 12 bc

Element added

P

4, 4

6, 8

12, 12

10, 10

e

Тест 2:

USAGE:

a <EL> <PATH> - add element on specified path.

d <PATH> - delete element for specified path.

Example for path: cbcc.

Example for el:

First diagonal(line x): 12

Second diagonal(line y): 15

P - print tree

p <PATH> - print element on specified path

e - end

Centre of all figures - point (0;0)

f 6 8

Wrong command!

a r g

Wrong values!

d b

Wrong path!

e

Тест 3:

USAGE:

a <EL> <PATH> - add element on specified path.

d <PATH> - delete element for specified path.

Example for path: cbcc.

Example for el:

First diagonal(line x): 12

Second diagonal(line y): 15

P - print tree

p <PATH> - print element on specified path

e - end

Centre of all figures - point (0;0)

a 6 8

Element added

p

Diagonals: 6, 8

Points: (3;0), (0;-4), (-3;0), (0;4)

Side: 5

Square: 24

a 4 4 b

Element added

a 10 10 bb

Element added

a 6 6 bc

Element added

P

6, 8

4, 4

6, 6

10, 10

p b

Diagonals: 4, 4

Points: (2;0), (0;-2), (-2;0), (0;2)

Side: 2.82843

Square: 8

d b

Element deleted

P

6, 8

10, 10

e

1. **Листинг программы**

**T\_NTree.h:**

#ifndef T\_NTREE\_H

#define T\_NTREE\_H

#include <iostream>

#include <string>

#include "Romb.h"

#include "T\_NTreeItem.h"

using namespace std;

class T\_NTree {

public:

T\_NTree();

T\_NTree(T\_NTree& old);

T\_NTreeItem\*& get\_root();

bool empty();

friend ostream& operator<<(ostream& os, T\_NTree& n\_tree);

void insert(string path, Romb value);

void delete\_el(string path);

T\_NTreeItem\*& get\_el(string path);

virtual ~T\_NTree();

private:

T\_NTreeItem \*root;

};

**T\_NTree.cpp:**

#include "T\_NTree.h"

#include <iostream>

#include <string>

using namespace std;

T\_NTree::T\_NTree() : root(nullptr) {

}

T\_NTree::T\_NTree(T\_NTree& old) : root(old.get\_root()){};

T\_NTreeItem\*& T\_NTree::get\_root(){

return root;

}

bool T\_NTree::empty(){

return root == nullptr;

}

static void print(ostream& os, T\_NTreeItem\*& tree, int h){

if(tree == nullptr){

return;

}

for(int i=0;i<h;i++){

os << "\t";

}

(tree->get\_value()).print\_diagonals();

print(os, tree->get\_first\_child(), h+1);

print(os,tree->get\_next\_bro(), h);

}

ostream& operator<<(ostream& os, T\_NTree& n\_tree){

T\_NTreeItem\*& root = n\_tree.get\_root();

if(root==nullptr){

cout << "Tree is empty\n";

}

print(os, root, 0);

return os;

}

void destroy(T\_NTreeItem\*& tree){

if(tree == nullptr) return;

destroy(tree->get\_first\_child());

destroy(tree->get\_next\_bro());

delete tree;

tree = nullptr;

}

T\_NTreeItem\*& way(T\_NTreeItem\*& tree, string path){

if((path != "") && (tree == nullptr)){

throw invalid\_argument("Wrong path!");

} else if(path == ""){

return tree;

} else if(path[0] == 'c'){

return way((tree)->get\_first\_child(), path.substr(1));

} else if(path[0] == 'b'){

return way((tree)->get\_next\_bro(), path.substr(1));

} else{

throw invalid\_argument("Wrong path!");

}

}

void T\_NTree::insert(string path, Romb value){

T\_NTreeItem\*& tree = way(root, path);

T\_NTreeItem\* new\_node = new T\_NTreeItem(value);

new\_node->set\_next\_bro(tree);

tree = new\_node;

}

void T\_NTree::delete\_el(string path){

T\_NTreeItem\*& tree = way(root, path);

if(tree == nullptr) throw invalid\_argument("Wrong path!");

T\_NTreeItem\* tmp = (tree)->get\_next\_bro();

destroy((tree)->get\_first\_child());

//delete tree;

tree = tmp;

}

T\_NTreeItem\*& T\_NTree::get\_el(string path){

T\_NTreeItem\*& tree = way(root, path);

if(tree==nullptr){

throw invalid\_argument("Wrong path!");

}

return tree;

}

T\_NTree::~T\_NTree() {

destroy(get\_root());

}

**T\_NTreeItem.h:**

#ifndef T\_NTREEITEM\_H

#define T\_NTREEITEM\_H

#include "Romb.h"

class T\_NTreeItem {

public:

T\_NTreeItem(const Romb& romb);

T\_NTreeItem(const T\_NTreeItem& orig);

friend std::ostream& operator<<(std::ostream& os, T\_NTreeItem& obj);

Romb get\_value();

T\_NTreeItem\*& get\_first\_child();

T\_NTreeItem\*& get\_next\_bro();

void set\_first\_child(T\_NTreeItem\*& child);

void set\_next\_bro(T\_NTreeItem\*& bro);

virtual ~T\_NTreeItem();

private:

Romb romb;

T\_NTreeItem \*first\_child;

T\_NTreeItem \*next\_bro;

};

#endif

**T\_NTreeItem.cpp:**

#include "T\_NTreeItem.h"

#include <iostream>

using namespace std;

T\_NTreeItem::T\_NTreeItem(const Romb& romb) {

this->romb = romb;

this->next\_bro = nullptr;

this->first\_child = nullptr;

}

T\_NTreeItem::T\_NTreeItem(const T\_NTreeItem& orig) {

this->romb = orig.romb;

this->next\_bro = orig.next\_bro;

this->first\_child = orig.first\_child;

}

Romb T\_NTreeItem::get\_value(){

return romb;

}

T\_NTreeItem\*& T\_NTreeItem::get\_first\_child(){

return first\_child;

}

T\_NTreeItem\*& T\_NTreeItem::get\_next\_bro(){

return next\_bro;

}

void T\_NTreeItem::set\_first\_child(T\_NTreeItem\*& child){

this->first\_child = child;

return;

}

void T\_NTreeItem::set\_next\_bro(T\_NTreeItem\*& bro){

this->next\_bro = bro;

return;

}

T\_NTreeItem::~T\_NTreeItem() {

delete next\_bro;

delete first\_child;

}

std::ostream& operator<<(std::ostream& os, T\_NTreeItem& obj) {

os << obj.get\_value() << "\n";

return os;

}

**Romb.h:**

#ifndef ROMB\_H

#define ROMB\_H

#include <cstdlib>

#include <iostream>

class Romb {

public:

Romb(double i,double j);

Romb();

Romb(const Romb& orig);

double square();

double side();

void print\_points();

void print\_diagonals();

friend std::ostream& operator<<(std::ostream& os, Romb& obj);

friend std::istream& operator>>(std::istream& is, Romb& obj);

bool operator==(Romb& copy);

Romb& operator=(const Romb& copy);

virtual ~Romb();

private:

double one;

double two;

};

#endif

**Romb.cpp:**

#include "Romb.h"

#include <iostream>

#include <cmath>

using namespace std;

Romb::Romb(double i, double j) : one(i), two(j) {

}

Romb::Romb() : one(0), two(0) {

}

Romb::Romb(const Romb& orig) {

one = orig.one;

two = orig.two;

}

double Romb::square(){

return (one \* two) / 2.0;

}

double Romb::side(){

return sqrt((one/2)\*(one/2) + (two/2)\*(two/2));

}

void Romb::print\_points(){

cout << "(" << one/2 << ";" << 0 << "), ";

cout << "(" << 0 << ";" << two/2\*(-1) << "), ";

cout << "(" << one/2\*(-1) << ";" << 0 << "), ";

cout << "(" << 0 << ";" << two/2 << ")";

}

void Romb::print\_diagonals(){

cout << one << ", " << two << "\n";

}

bool Romb::operator==(Romb& copy) {

if (one == copy.one && two == copy.two){

return true;

} else {

return false;

}

}

Romb& Romb::operator=(const Romb& copy) {

if (this == &copy) return \*this;

one = copy.one;

two = copy.two;

return \*this;

}

Romb::~Romb() {

}

std::ostream& operator<<(std::ostream& os, Romb& obj) {

os << "Points: ";

obj.print\_points();

os << "\nSide: " << obj.side() << "\n" << "Square: "<< obj.square();

return os;

}

std::istream& operator>>(std::istream& is, Romb& obj) {

if(!(is >> obj.one >> obj.two)){

throw invalid\_argument("Wrong values!");

}

return is;

}

**Файл main.cpp:**

// Почечура Артемий Андреевич

// М8О-206Б-20

// Лабораторная работа №4

// Cпроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий ромб (Вариант 12, Лабораторная работа №3).

// Структура данных для контейнера: Вариант 19 - N-дерево.

#include <iostream>

#include <sstream>

#include <math.h>

#include <string>

#include <vector>

#include <algorithm>

#include <map>

#include <queue>

#include <stack>

#include <set>

#include "T\_NTree.h"

using namespace std;

void usage(){

cout << "USAGE:\n\na <EL> <PATH> - add element on specified path.\n\nd <PATH> - delete element for specified path.\n\nExample for path: cbcc.\n";

cout << "Example for el:\n\n";

cout << "First diagonal(line x): 12\n";

cout << "Second diagonal(line y): 15\n";

cout << "\nP - print tree";

cout << "\np <PATH> - print element on specified path";

cout << "\ne - end\n";

cout << "\nCentre of all figures - point (0;0)\n\n";

}

int main(int argc, char\*\* argv) {

T\_NTree tree;

usage();

while(true){

try{

string s;

getline(cin,s);

istringstream iss(s);

char a;

iss >> a;

if(a=='a'){

string S="";

Romb a;

iss >> a;

iss >> S;

tree.insert(S,a);

cout << "Element added\n";

} else if(a=='d'){

string S="";

iss >> S;

tree.delete\_el(S);

cout << "Element deleted\n";

} else if(a=='P'){

cout << tree;

} else if(a=='p'){

string S="";

iss >> S;

if(tree.get\_el(S)){

cout << "Diagonals: ";

((\*tree.get\_el(S)).get\_value()).print\_diagonals();

cout << \*tree.get\_el(S);

}

} else if(a=='e'){

break;

}else {

throw invalid\_argument("Wrong command!");

}

} catch(const invalid\_argument &arg){

cout << arg.what() << "\n";

}

}

return 0;

}

1. **Вывод**

В ходе данной работы я попрактиковался работать с такой динамической структурой данных, как n-дерево. При написании программ новым было определение операции вывода для целой структуры, задействование заголовочных файлов в C++, а также работа с n-деревом в данной среде. Также было интересно взаимодействовать с классами и использовать их при создании структуры данных.

**ЛИТЕРАТУРА**

1. Уроки программирования на языке С++ [Электронный ресурс]. URL: <https://ravesli.com/uroki-cpp> (дата обращения 24.09.2021)