## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

## ОТЧЕТ

**по лабораторной работе** №4

**по дисциплине «Алгоритмы и структуры данных»**

## ТЕМА: Бинарные деревья

|  |  |  |
| --- | --- | --- |
| Студент гр. 7303 |  | Юсковец А.В. |
| Преподаватель |  | Балтрашевич Т.А. |

Санкт-Петербург  
2018

**Оглавление**

[Цель работы 3](#__RefHeading___Toc1709_692322312)

[Задание 3](#__RefHeading___Toc8601_2844259154)

[Ход работы 3](#__RefHeading___Toc1711_692322312)

[Вывод 3](#__RefHeading___Toc1731_692322312)

[Приложение А. Бинарное дерево 4](#__RefHeading___Toc8079_1876010945)

[Приложение Б. Интерфейс 5](#__RefHeading___Toc7979_1876010945)

# Цель работы

Реализовать структуру данных: бинарное дерево. Решить поставленное задание с использованием своей реализации бинарного дерева.

# Задание

Для заданного бинарного дерева b типа ВТ с произвольным типом элементов напечатать элементы из всех листьев дерева b.

# Ход работы

Для решения поставленной задачи используется бинарное дерево, реализация которого приведена в приложении А.

Реализация требуемой функции приведена в приложении А(void **print\_leaves**(std::ostream& out)):

Функция **print\_leaves** принимает поток, в который нужно вывести листья данного дерева. Пока алгоритм не наткнется на лист (потомки — нулевые указатели) он будет рекурсивно вызывать себе от потомков. Как только алгоритм доходит до листа, его значение выводится.

Функция считывания дерева приведена в приложении A.

# Вывод

На языке программирования С++ было реализовано бинарное дерево на базе указателей. Был разработан метод, выводящий на экран все листья дерева. При компиляции и выполнении программы не возникает ошибок и предупреждений.

# Приложение А. Бинарное дерево

#ifndef TREE\_H

#define TREE\_H

#include <iostream>

#include <exception>

template <typename T>

class **Tree** {

class **Node** {

public:

T value;

Node\* left = nullptr;

Node\* right = nullptr;

~**Node**() {

if (left)

delete left;

if (right)

delete right;

}

void **output**(std::ostream& out) {

if (this) {

out << value;

if (left) {

out << '(';

left->output(out);

out << ')';

}

if (right) {

out << '(';

right->output(out);

out << ')';

}

}

}

void **read**(std::istream& in) {

char c = 0;

if (this) {

in >> value;

/\* left node \*/

c = in.get();

if (c == '(') {

if (in.peek() != ')') {

left = new Node();

left->read(in);

} else {

c = in.get();

left = nullptr;

}

} else if (c == ')')

return;

/\* right node \*/

c = in.get();

if (c == '(') {

if (in.peek() != ')') {

right = new Node();

right->read(in);

} else {

c = in.get();

right = nullptr;

}

} else if (c == ')')

return;

/\* this node \*/

c = in.get();

if (c != ')')

throw std::logic\_error("Incorrect input format");

} else

throw std::logic\_error("Incorrect input format");

}

void **print\_leaves**(std::ostream& out) {

if (this) {

if (!left && !right)

out << value << ' ';

if (left)

left->print\_leaves(out);

if (right)

right->print\_leaves(out);

}

}

};

public:

**Tree**(): \_root(new Node) {}

~**Tree**() {

\_root->~Node();

}

void **print\_leaves**(std::ostream& out) {

\_root->print\_leaves(out);

}

template<typename U>

friend std::istream& operator>>(std::istream& in, Tree<U>& tree);

private:

Node\* \_root;

};

template<typename U>

std::istream& operator>>(std::istream& in, Tree<U>& tree) {

if (tree.\_root)

tree.~Tree<U>();

if (in.get() == '(' && in.peek() != ')') {

tree.\_root = new typename Tree<U>::Node();

tree.\_root->read(in);

} else

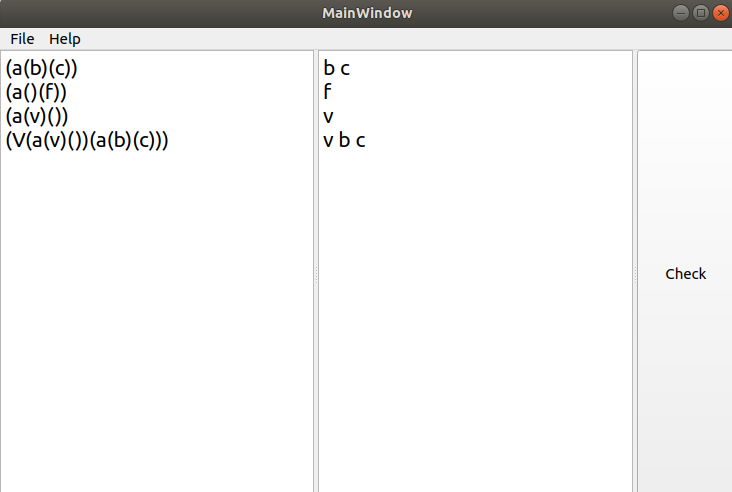
tree.\_root = nullptr;

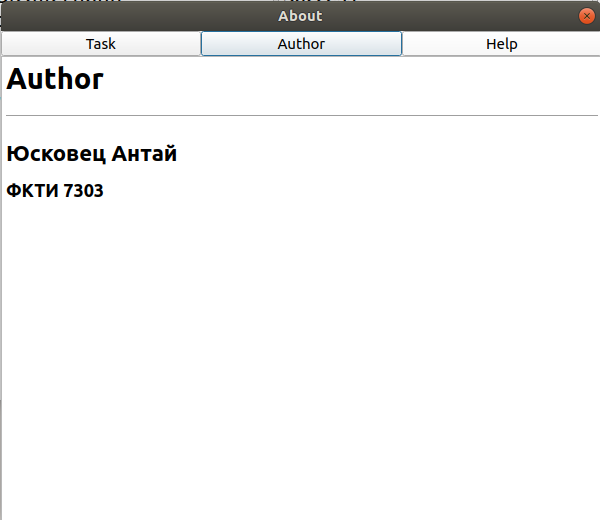
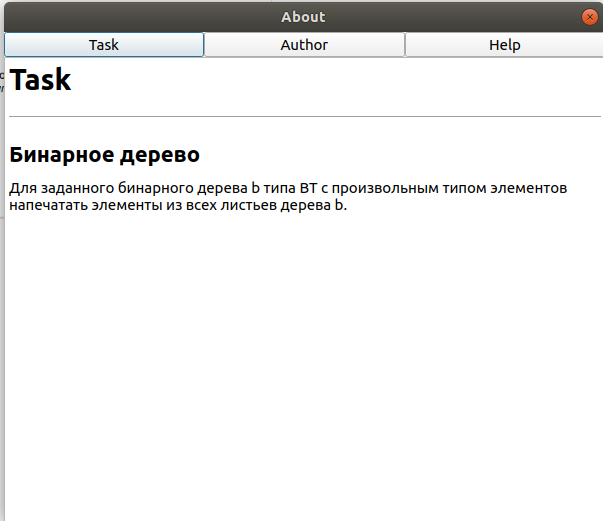
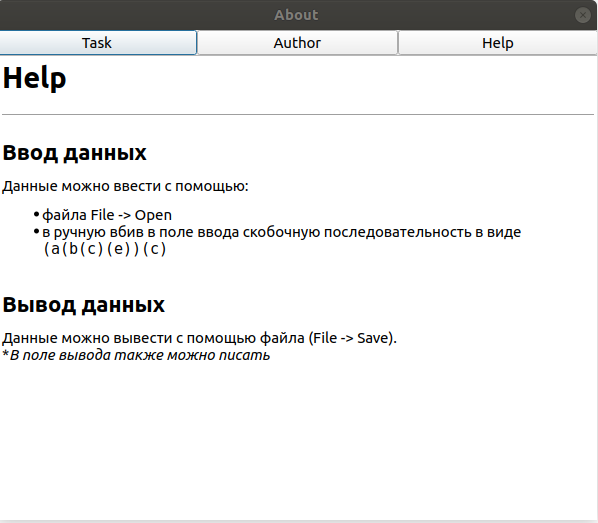
return in;

}

#endif // TREE\_H

# Приложение Б. Интерфейс





# Приложение В. Исходный код QT

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

using namespace std;

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(parent),

ui(new Ui::MainWindow) {

ui->setupUi(this);

ui->input->setFontPointSize(16);

ui->output->setFontPointSize(16);

setCentralWidget(ui->splitter);

}

MainWindow::~***MainWindow***() {

delete ui;

}

void MainWindow::**on\_actionOpen\_triggered**() {

input\_file = QFileDialog::getOpenFileName(this, "Choose input file");

std::ifstream file{input\_file.toStdString()};

while (file.good()) {

std::string list\_repr;

file >> list\_repr;

input << list\_repr << '\n';

}

ui->input->append(QString(input.str().c\_str()));

}

void MainWindow::**on\_actionSave\_triggered**() {

output\_file = QFileDialog::getSaveFileName(this, "Choose output file");

std::ofstream file{output\_file.toStdString()};

std::string output = ui->output->toPlainText().toStdString();

if (file.good())

file << output;

file.close();

}

void MainWindow::**on\_button\_clicked**() {

input = std::stringstream(ui->input->toPlainText().toStdString());

output = std::stringstream();

ui->output->clear();

Tree<char> tree;

while(input.good()) {

std::string str;

input >> str;

if (str != "") {

if (str[0] == '(') {

try {

std::stringstream{str} >> tree;

tree.print\_leaves(output);

output << '\n';

} catch (std::logic\_error& err) {

output << err.*what*() << '\n';

} catch (std::exception& err) {

output << "Something went wrong" << '\n';

}

} else

output << "First bracket is necessary" << '\n';

}

}

ui->output->append(QString(output.str().c\_str()));

}

void MainWindow::**on\_actionAbout\_triggered**() {

About about;

about.*exec*();

}

#include "about.h"

#include "ui\_about.h"

About::**About**(QWidget \*parent) : QDialog(parent),

ui(new Ui::About) {

ui->setupUi(this);

ui->textBrowser->*setSource*(QString("qrc:/task.html"));

setWindowTitle("About");

}

About::~***About***() {

delete ui;

}

void About::**on\_taskButton\_clicked**() {

ui->textBrowser->*setSource*(QString("qrc:/task.html"));

}

void About::**on\_helpButton\_clicked**() {

ui->textBrowser->*setSource*(QString("qrc:/help.html"));

}

void About::**on\_authorButton\_clicked**() {

ui->textBrowser->*setSource*(QString("qrc:/author.html"));

}