**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЁТ**

**по лабораторной работе №4**

**по дисциплине «АиСД»**

**Тема: Бинарные деревья**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3303 |  |  |
| Преподаватель |  |  |

Санкт-Петербург

2018

Цель работы:

Изучение бинарных деревьев и способов работы с ними.

Задание:

Для заданного бинарного дерева *b* типа *ВТ* с произвольным типом элементов подсчитать число узлов на заданном уровне *п* дерева *b* (корень считать узлом 1-го уровня).

Коды всех функций, слотов, и описания структур находятся в Приложении А.

**Ход работы:**

Была написана структура для бинарного дерева и вынесена в отдельный заголовочный файл.

Затем были описаны функции работы со бинарным деревом:

* функция для создания пустого листа node\* make\_leaf(char\* date)
* функция для считывания бинарного дерева из потока void read(node \*\*tree, istringstream &in\_stream)
* функция для подсчёта узлов на уровне n бинарного дерева int number\_N\_level(node\* tree,int N)
* функция для ступенчатой отрисовки бинарного дерева void MainWindow::write\_tree(node\* tree,int n)

Далее были описаны слоты – функции, вызываемые при нажатии определённых пунктов меню:

* слот on\_actionInfo\_triggered() – вызывается при выборе в меню пункта «Info» и вызывает справку по программе;
* слот on\_actionAbout\_Authour\_triggered() – вызывается при выборе в меню пункта «Author» и вызвает справку об авторе;
* слот on\_pushButton\_clicked()– вызывает загрузку данных в бинарное дерево и последующую отрисовку и вывод результата;
* слот on\_action\_triggered ­ вызывает загрузку данных из файла.

Далее был описан класс HelpBrowser в файле helpbrowser.h, в котором содержится лишь описание конструктора, который в переданном по указателю виджете располагает кнопки для переключения между html страницами, и «соединяет» соответствующие сигналы со слотами.

Далее были созданы 5 html файлов, описывающих страницы для виджета HelpBrowser.

**Описание работы приложения:**

Само приложение представляет из себя окно с несколькими текстовым полями QTextEdit для ввода бинарного дерева и вывода его в схематическом виде, с полем QLineText для вывода результат, полем QSpinButton для выбора уровня дерева и меню вверху окна. В текстовое поле необходимо ввести бинарное дерево в скобочной записи, выбрать необходимый уровень и нажать кнопку «Считать». Также бинарное дерево можно ввести из файла через пункт меню File. Результат будет выведен в текстовом поле. Также в верхнем меню есть пункты Help и Author для просмотра информации о всех функциях программы, задании и её авторе соответственно. При их выборе создастся отдельное окно - QWidget.

**Пример работы программы:**

На рис. 2-3 приведены примеры работы программы

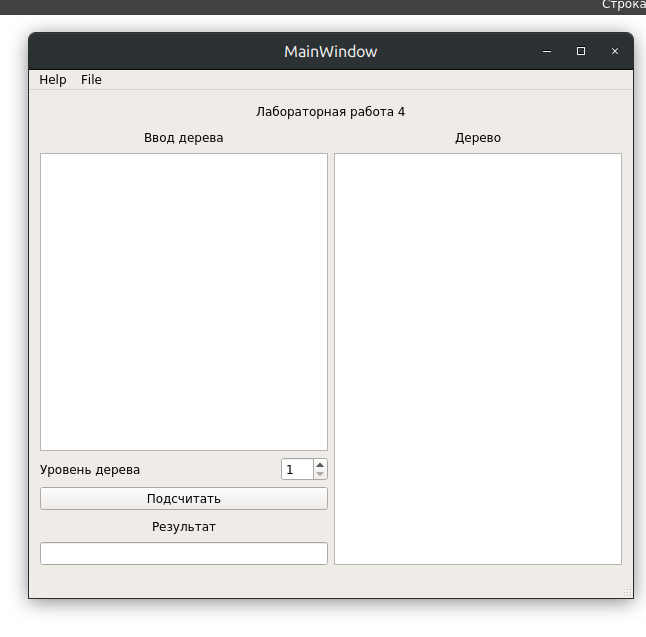


Рисунок 2 – Пример работы программы

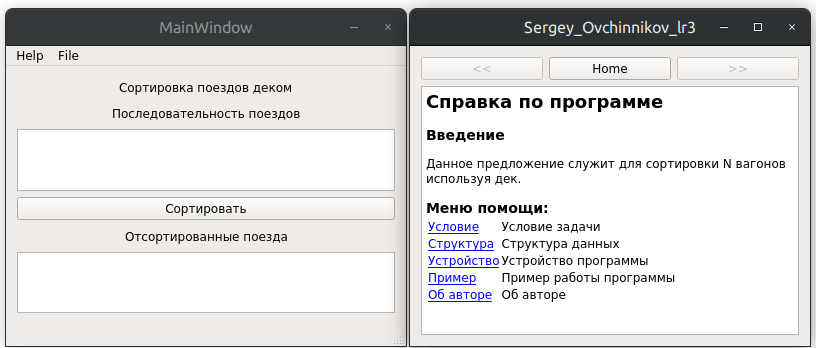


Рисунок 3 – Справка по программе

Вывод:  
С помощью функций стандартных библиотек c++ было осуществлено решение поставленной задачи с помощью дека, получены навыки в работе с стеками и списками.

# **ПРИЛОЖЕНИЕ А**

ИСХОДНЫЙ КОД ПРОГРАММЫ

Mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "HelpBrowser.h"

#include <QMessageBox>

#include <iostream>

#include <sstream>

#include <cstdlib>

#include <fstream>

using namespace std ;

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_actionInfo\_triggered()

{

HelpBrowser \*helpBrowser = new HelpBrowser(":/", "index.htm");

helpBrowser->resize(400, 300);

helpBrowser->show();

}

void MainWindow::on\_actionAbout\_Authour\_triggered()

{

HelpAuthor \*helpAuthor = new HelpAuthor(":/", "author.htm");

helpAuthor->resize(300, 400);

helpAuthor->show();

}

typedef struct lisp{

int field;

struct lisp \*next;

struct lisp \*prev;

}lisp;

typedef struct DECK{

lisp\* head;

lisp\* tail;

int size;

}DECK;

void push\_deck\_head(DECK &deck,int number){

lisp \*temp = new lisp;

if(deck.head != NULL){

temp->field=number;

temp->prev= NULL;

temp->next=deck.head;

deck.head->prev=temp;

deck.head=temp;

}else{

deck.head = temp;

deck.head->field=number;

deck.head->next=NULL;

deck.head->prev=NULL;

deck.tail = deck.head;

}

deck.size++;

}

void push\_deck\_tail(DECK &deck,int number){

lisp \*temp = new lisp;

temp->field=number;

temp->prev= deck.tail;

temp->next=NULL;

deck.tail->next=temp;

deck.tail = temp;

deck.size++;

}

int pop\_deck\_head(DECK &deck){

int x = deck.head->field;

if(deck.size != 1){

deck.head = deck.head->next;

delete deck.head->prev;

deck.head->prev=NULL;

}else{

delete deck.head;

deck.head =NULL;

deck.tail =NULL;

}

deck.size--;

return x;

}

int pop\_deck\_tail(DECK &deck){

int x = deck.tail->field;

if(deck.size != 1){

deck.tail = deck.tail->prev;

delete deck.tail->next;

deck.tail->next=NULL;

}else{

delete deck.tail;

deck.head =NULL;

deck.tail =NULL;

}

deck.size--;

return x;

}

void read\_deck(DECK &deck, istringstream &in\_stream){

char c;

while((c = in\_stream.peek()) != '$'){

if(c == '\n' || c == ' ' || c == ',' || c == ';' ){

in\_stream.get();

}else{

int x;

in\_stream>>x;

push\_deck\_head(deck,x);

}

}

}

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

DECK deck ={ NULL ,NULL,0 };

QString input\_str;

string str = ui->Train\_input->toPlainText().toLocal8Bit().data();

str+="$";

istringstream input\_string(str);

read\_deck(deck,input\_string);

if(deck.size>0){

int N = deck.size;

for (int i = 0; i < N/2; i++) {

input\_str+=QString::number(pop\_deck\_head(deck));

input\_str+=", ";

input\_str+=QString::number(pop\_deck\_tail(deck));

if(deck.size != 0){

input\_str+=", ";

}else{

input\_str+=";";

}

}

if( deck.size == 1){

input\_str+=QString::number(pop\_deck\_head(deck));

input\_str+=";";

}

}else{

input\_str = "NULL";

}

ui->Train\_output->setText(input\_str);

}

void MainWindow::on\_action\_triggered(){

QString patch\_to\_file = QFileDialog::getOpenFileName(this,"Open file", QDir::currentPath(),

"File with date (\*.txt);;All files (\*.\*)");

if(patch\_to\_file.isEmpty()) return;

QFile file(patch\_to\_file);

QByteArray data;

if(!file.open(QIODevice::ReadOnly)) return;

data = file.readLine();

ui->Train\_input->setText(data);

}

Mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private slots:

void on\_actionInfo\_triggered();

void on\_actionAbout\_Authour\_triggered();

void on\_sort\_button\_clicked();

void on\_action\_triggered();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H