МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Идеально сбалансированные БДП

Студент гр. 8381 _____ Сахаров В.М.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Сахаров В.М.

Группа 8381

Тема работы: Идеальные БДП.

Исходные данные: необходимо провести демонстрацию алгоритма вставки и удаления из идеального БДП дерева с поддержкой пошагового режима и сохранения текущего состояния в файл.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание», «Описание программы», «Демонстрация», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент		Сахаров В.М.
Преподаватель		Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была написана программа, которая может обрабатывать входную строку, строить по ней идеальное бинарное дерево. Возможен ввод из файла (скобочная запись). Доступно быстрое восстановление предыдущей сессии. Также сделана визуализация данных и пошаговая демонстрация алгоритма

SUMMARY

During the course of the course work, a program was written that can process the input string, build an ideal binary tree on it. It is possible to proceed input string from file (with brackets). There is ability of restoring previous session. Visualization of data and stepped execution of algorithms in also included.

СОДЕРЖАНИЕ

	Введение	6
1.	Задание	7
2.	Описание программы	8
2.1.	Описание интерфейса пользователя	8
2.2.	Описание основных классов для хеш-таблицы	10
2.3.	Описание алгоритма вставки в хеш-таблицу	12
2.4.	Описание класса StrStrWorker для работы с хеш-таблицей	12
2.5.	Описание структур и функций для исследования	14
3	Тестирование	16
3.1.	Вид программы	16
3.2.	Тестирование генерации файла	17
3.3.	Тестирование создания хеш-таблицы	18
4	Исследование	21
4.1	План экспериментального исследования	21
4.2	Технология проведения исследования	22
4.3	Исследование зависимостей от максимальной длины ключа	24
4.4	Исследование зависимостей от множителя размера таблицы	26
4.5	Исследование зависимостей от числа пар	29
4.6	Вывод об эффективности хеш-функций	34
4.7	Исследование худшего случая вставки	35
4.8	Выводы об исследовании алгоритма	37
	Заключение	38
	Список использованных источников	39
	Приложение А. Исходный код программы. main.c	40
	Приложение Б. Исходный код программы. pair.h	41
	Приложение В. Исходный код программы. cvector.h	46
	Приложение Г. Исходный код программы. hashtable.h	48
	Приложение Д. Исходный код программы. strstrworker.h	53
	Приложение Е. Исходный код программы. strstrworker.cpp	54
	Приложение Ж. Исходный код программы. strstrtester.h	60
	Приложение И. Исходный код программы. strstrtester.cpp	61
	Приложение К. Исходный код программы. mainwindow.h	65
	Приложение Л. Исходный код программы. mainwindow.cpp	66
	Приложение М. Исходный код программы. libraries.h	70
	Приложение Н. Исходный код программы. lr5.pro	71
	Приложение О. Исходный код программы. mainwindow.ui	72

ВВЕДЕНИЕ

Цель работы

Реализация и демонстрация алгоритмов вставки и удаления элементов из идеального БДП.

Основные задачи

Реализация ввода из файла и из поля ввода, визуализация дерева, а также алгоритмов вставки и удаления с возможностью пошагового выполнения.

Методы решения

Разработка программы велась на базе операционной системы Windows 10 в среде разработки QtCreator. Для создания графической оболочки использовался редактор интерфейса в QtCreator и система сигналов-слотов Qt. Для реализации дерева были созданы классы пары utils_tree и node, а также вспомогательные список utils_vector и utils_list, реализующие интерфейс IList. Для графического отображения результатов исследования использовались классы QgraphicsView и QGraphicsScene

1. ЗАДАНИЕ

Необходимо продемонстрировать алгоритм удаления и вставки в идеальное БДП.

Демонстрация должна содержать:

1. Анализ задачи, цели, технологию проведения и план экспериментального исследования.
2. Реализация ввода требуемых данных несколькими способами (из файла, из предыдущей сессии, из поля ввода).
3. Выполнение исследуемых алгоритмов, в том числе, в пошаговом режиме.

2. ОПИСАНИЕ ПРОГРАММЫ

2.1. Описание интерфейса пользователя

Интерфейс программы разделен на три части: верхняя панель ввода данных, построения дерева и масштаба визуализации, нижняя панель управления ходом демонстрации и центральный виджет визуализации. Основные виджеты и их назначение представлены в табл. 1.

Таблица 1 – Основные виджеты верхней панели программы

Класс объекта	Название виджета	Назначение
QLineEdit	input	Поле ввода дерева
QPushButton	butLoad	Кнопка загрузки предыдущей сессии
QPushButton	butFile	Кнопка ввода дерева из файла
QPushButton	butGenerate	Кнопка генерации БДП
QPushButton	butGenerateOptimal	Кнопка генерации оптимального БДП

Основные виджеты нижней панели программы и их назначение представлены в табл. 2.

Таблица 2 – Основные виджеты нижней панели программы

Класс объекта	Название виджета	Назначение
QPushButton	butUndo	Флаг выбора генерации худшего случая
QPushButton	butUndo	Кнопка отмены действия
QPushButton	butRedo	Кнопка отмены отменённого действия
QPushButton	butAdd	Кнопка добавления элемента в дерево
QPushButton	butStepAdd	Кнопка пошагового добавления элемента в дерево
QPushButton	butRemove	Кнопка удаления элемента из дерева
QPushButton	butStepRemove	Кнопка пошагового удаления элемента из дерева

2.2. Описание основных классов для идеального БДП

Для реализации идеального БДП были созданы шаблонные классы для узла дерева - node и его логики - utils_tree. Класс node содержит поле для данных любого типа, цвет и насыщенность цвета для визуализации, а также указатели на левый и правый дочерние узлы. Классы utils_linked и utils_vector были реализованы для предыдущей лабораторной работы. Класс node не содержит методов.

Класс utils_tree для представления дерева содержит в себе ряд методов для работы с ним и его узлами:

Таблица 3 – Основные методы класса utils_tree (некоторые методы имеют перегрузку для удобства работы с корнем)

Метод	Назначение
void clean(); void clean(node<T>* n);	Очищает дерево, рекурсивно удаляя все узлы и устанавливая root = nullptr
bool search(T key); node<T>* search(node<T>* n, T key);	Рекурсивно производит поиск элемента со значением key
bool undo(); bool redo();	Соответственно восстанавливают состояние дерева из списков undolist и redolist
void save(); void load();	Соответственно сохраняют и загружают состояние дерева
bool parse_tree(node<T>*& n, std::string &s, int &i);	Считывает дерево из скобочной записи в строке
void go_darker(node<T>* n);	Рекурсивно уменьшает яркость всех узлов для улучшения качества демонтрации
int height(node<T>* n);	Возвращает высоты узла для алгоритма балансировки
int bfactor(node<T>* n);	Возвращает разность высот дочерних узлов заданного узла

Окончание таблицы 3

void fixheight(node<T>* n);	Восстанавливает высоту заданого узла с учётом того, что высоты дочерних узлов верные
node<T>* rotateleft(node<T>* n); node<T>* rotateright(node<T>* n);	Поворачивает два узла дерева для балансировки, перевешивая родительский и центральный узлы
node<T>* balance(node<T>* n);	Используя разницу высот, использует нужный поворот для балансировки узла
node<T>* findmin(node<T>* n);	Рекурсивно находит минимальный дочерний узел заданного узла
node<T>* removemin(node<T>* n);	Рекурсивно удаляет минимальный дочерный узел заданного узла

void insertbalanced(T data);
void removebalanced(T data);
node* insertbalanced(node* n, T data);
node* removebalanced(node* n, T data);
	Используя предыдущие описанные методы, соответственно, вставляют и удаляют узлы с заданными значениями, после чего восстанавливают баланс дерева

2.3. Описание алгоритма вставки в идеальное БДП

Показатель сбалансированности - разность между высотой левого и правого поддерева. Непосредственно при вставке узлу присваивается нулевой баланс. Процесс включения вершины состоит из трех частей:

Таблица 4 — алгоритм вставки в идеальное БДП

1	Прохода по пути поиска для проверки, что ключа в дереве нет.
2	Включения новой вершины в дерево и определения результирующих показателей балансировки.
3	Возвращение назад по пути поиска и проверки в каждой вершине показателя сбалансированности. Если модуль разности больше 1 - балансировка.

2.4. Описание алоритма удаления из идеального БДП

Если вершина — лист, то удалим её и вызовем балансировку всех её предков в порядке от родителя к корню. Иначе найдём самую близкую по значению вершину в поддереве наибольшей высоты (правом или левом) и переместим её на место удаляемой вершины, при этом вызвав процедуру её удаления.

Для простоты опишем рекурсивный алгоритм удаления. Если вершина — лист, то удалим её, иначе найдём самую близкую по значению вершину, переместим её на место удаляемой вершины. От удалённой вершины будем подниматься вверх к корню и пересчитывать баланс у вершин. Если пришли в вершину и её баланс стал равным 1 или −1, то это значит, что высота этого поддерева не изменилась и подъём можно остановить. Если баланс вершины стал равным нулю, то высота поддерева уменьшилась и подъём нужно продолжить. Если баланс стал равным 2 или −2, следует выполнить одно из четырёх вращений и, если после вращений баланс вершины стал равным нулю, то подъём продолжается, иначе останавливается.

2.5 Описание методов демонстрации

Все методы визуализации располагаются в классе mainwindow. Ниже перечислены основные из них:

Таблица 5 – Основные методы визуализации дерева

void UpdateGraphics(bool save = true);	Обновляет graphicsscene, визуализируя текущее состояние дерева
void DrawNode(node<int>* n, int maxdepth, int depth = 0, int x = 0, int y = 0);	Рекурсивно рисует на graphicsscene все дочерние узлы, включая текущий
void SetActiveButtons(bool mode);	Деактивирует кнопки, которые не должны быть нажаты в данный момент
bool ReadElement();	Считывает элемент из строки ввода

Окончание таблицы 5

void ChangeMode(mode newmode, QPushButton* but);	Устанавливает текущий режим выполнения (для пошаговых алгоритмов)
bool TryRedo();	Восстанавливает отменённое состояние дерева, если в tree.redolist остались элементы

Особенностями данной визуализации являются цветовая маркировка изменённых элементов, сохранение состояния на диск при каждом изменении дерева, ручное асштабирование больших деревьев, не умещающихся на экран, а также возможность отменить любое действие (поддерживаются комбинации клавиш Ctrl+Z и Ctrl+X соответственно).

3. ДЕМОНСТРАЦИЯ

3.1. Вид программы

Программа представляет собой окно с графическим интерфейсом, запускающимся в полноэкранном режиме. Вид программы после запуска представлен на рис. 1.

Рисунок 1 – Вид программы после запуска

На рис. 2 представлено сгенерированное дерево

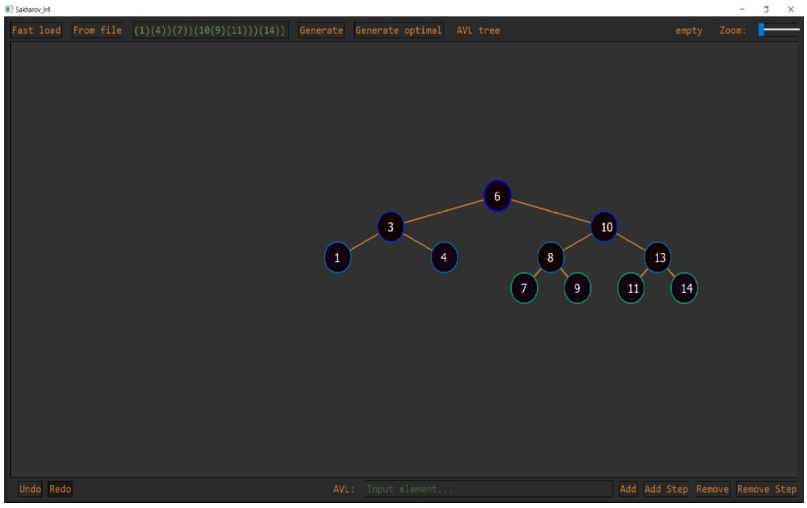

Рисунок 2 – Вид программы после генерации идеального БДП

3.2. Демонстрация вставки элемента

После ввода в строку для элемента значения производится нажатие кнопки «Add» (в случае отсутствия элемента выбирается случайный элемент из небольшого диапазона):

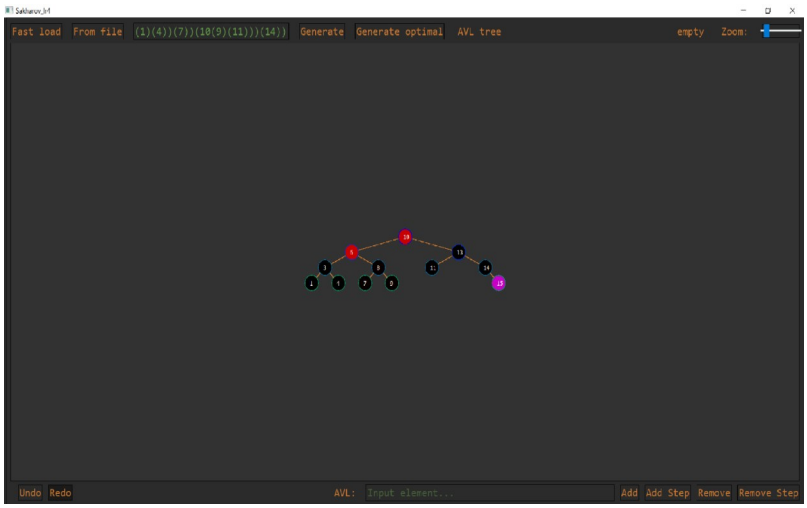

Рисунок 3 – Вид программы после генерации идеального БДП

3.3. Демонстрация пошаговой вставки элемента

На рис. 4, 5, 6 представлен процесс вставки элемента в идеальное БДП:

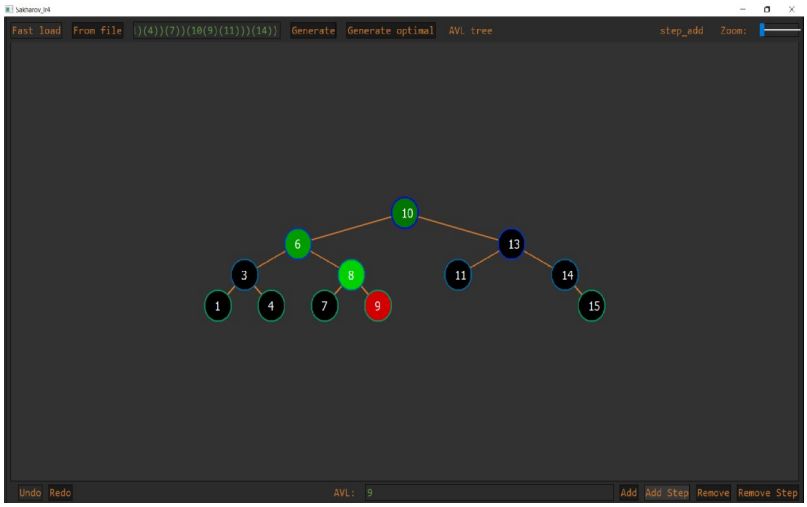

Рисунок 4 – проход вниз при вставке элемента

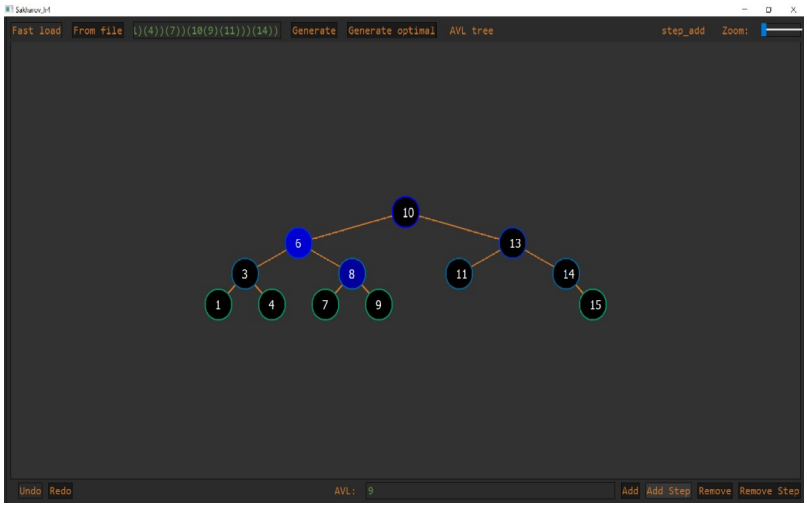

Рисунок 5 – проверка сбалансированности узлов при возврате к корню

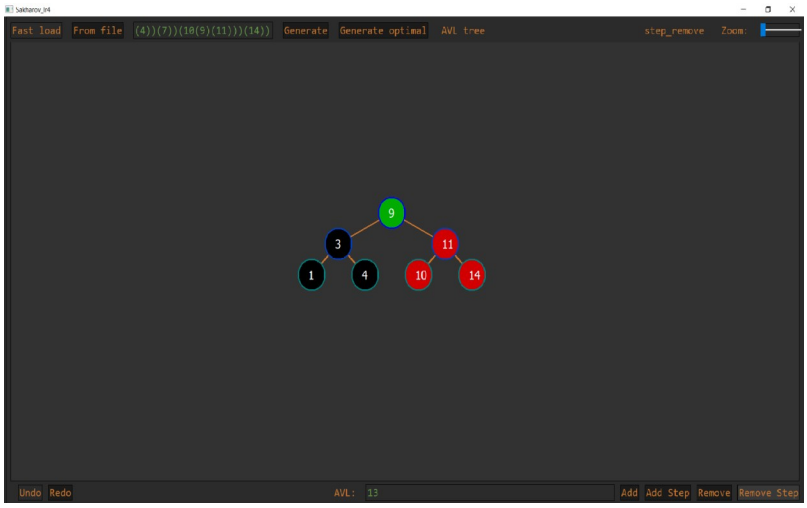

Рисунок 6 – балансировка при вставке элемента

3.4. Демонстрация удаления элемента

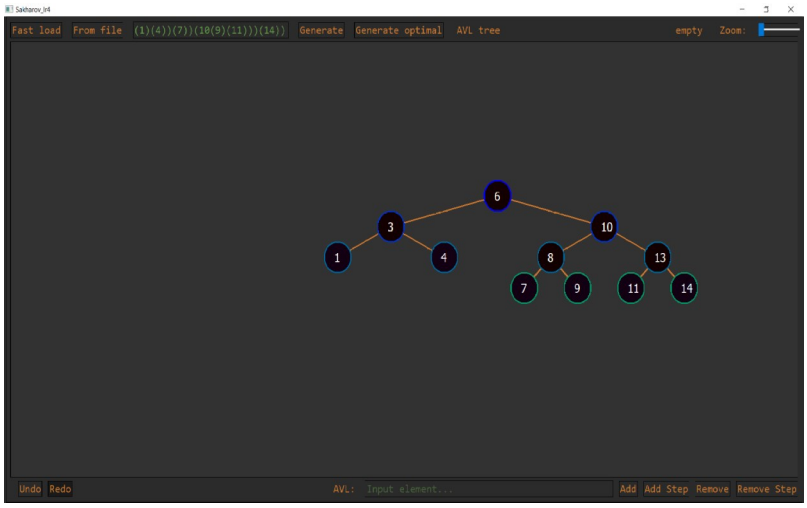

Рисунок 7 – Вид программы после удаления элемента из иидеального БДП

3.5. Демонстрация пошагового удаления элемента

На рис. 8,9,10 представлен процесс удаления элемента и идеального БДП:

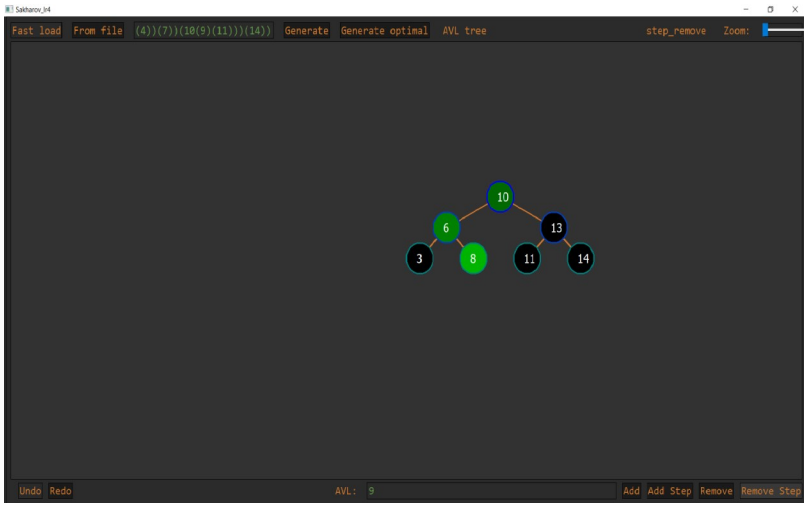

Рисунок 8 – Поиск требуемого элемента

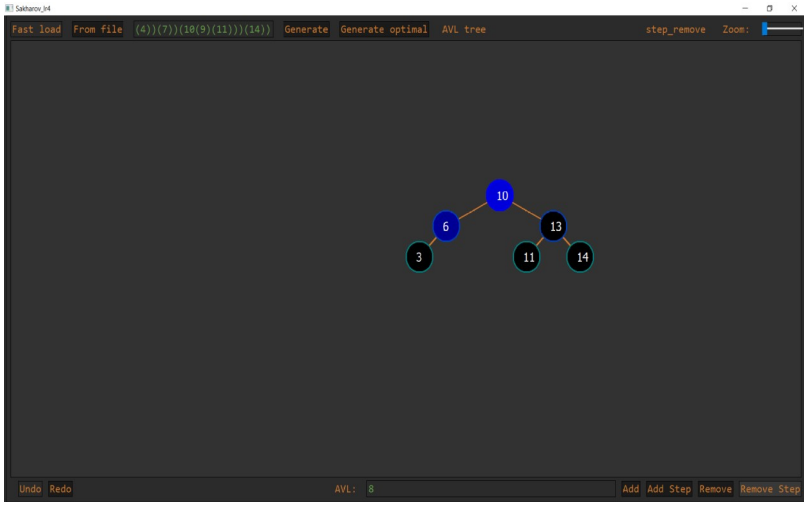

Рисунок 9 – Проверка сбалансированности узлов после удаления элемента

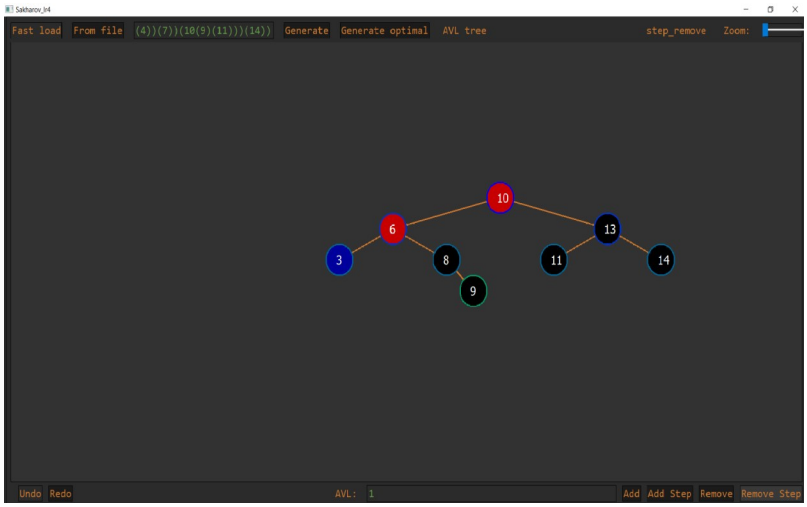

Рисунок 10 – Балансировка дерева после удаления элемента

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, которая обладает следующей функциональностью: генерация идеального БДП по входным данным из разных источников ввода, вставка и удаление элемента из идеального БДП; пошаговое выполнение вставки и удаления из идеального БДП, а также функциональность отмены и возврата к предыдущим состояниям программы. Все операции с деревом сопровождаются соответствующей цветовой маркировкой, что позволяет использовать её для изучения алгоритма вставки и удаления из идеального БДП (АВЛ-дерева).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bjarne Stroustrup. A Tour of C++. M.: Addison-Wesley, 2018. 217 с.
2. Макс Шлее. Qt5.10. Профессиональное программирование на C++. M.: BHV-СПб, 2018, 513 с.
3. Перевод и дополнение документации QT // CrossPlatform.RU. URL: http://doc.crossplatform.ru/
4. Qt Documentation // Qt. URL: https://doc.qt.io/qt-5/index.html
5. Habrahabr. // Habr. URL: https://habr.com/
6. Лекция 3: АВЛ-деревья. // M.Kurnosov URL: http://www.mkurnosov.net/teaching/uploads/DSA/dsa-fall-lecture3.pdf

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.C

```
#include "mainwindow.h"
#include "utils_cli.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    for(int i = 0; i < argc; i++) {
          if(!strcmp("console", argv[i]) || !strcmp("-console", argv[i]) || !
strcmp("-c", argv[i])) {
            return utils_cli::execute(argc - 1, argv + 1);
        }
    }
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. LIST.H

```
#ifndef ILIST_H
#define ILIST_H

template <class T>
struct IList {
    virtual T operator[] (int index) = 0;
    virtual T at (int index) = 0;
    virtual void clean() = 0;
    virtual void insert(int index, T element) = 0;
    virtual T remove(int index) = 0;

    virtual T back() = 0;
    virtual void push_back(T element) = 0;
    virtual T pop_back() = 0;

    virtual T front() = 0;
    virtual void push_front(T element) = 0;
    virtual T pop_front() = 0;

    virtual int size() = 0;
    virtual bool empty() = 0;
    virtual ~IList(){}
};

#endif // ILIST_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "utils_headers.h"
#include "utils_linked.h"
#include "utils_vector.h"
#include "utils_tree.h"
#include <cmath>

enum mode
{
    empty,
    bst,
    pyramid,
    step_add,
    step_remove,
};

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void UpdateGraphics(bool save = true);

private slots:
    void on_butFile_clicked();

    void on_butGenerate_clicked();

    void on_butRun_clicked();

    void on_horizontalSlider_sliderMoved(int position);

    void on_butStepBst_clicked();

    void on_butStepPyramid_clicked();
```

```
    void on_butAdd_clicked();

    void on_butStepAdd_clicked();

    void on_butRemove_clicked();

    void on_butGenerateOptimal_clicked();

    void on_butStepRemove_clicked();

    void updateTreeColor();

    void on_butUndo_clicked();

    void on_butRedo_clicked();

    void on_butLoad_clicked();

private:
    utils_tree<int>* tree;
    int current;
    bool is_bst;
    bool is_pyramid;
    utils_vector<node<int>*> bst_stepped;
    utils_vector<node<int>*> pyramid_stepped;
    utils_linked<node<int>*> stack;
    node<int>* state_node;
    int state;
    bool locked;
    bool lockedUpd;
    mode stepped_mode;
    Ui::MainWindow *ui;
    QGraphicsScene *mainGraphicsScene;
    QPen pen;
    QColor color;
    QBrush brush;
    QFont font;
    void DrawNode(node<int>* n, int maxdepth, int depth = 0, int x = 0, int y =
0);
    void SetActiveButtons(bool mode);
    bool ReadElement();
    void ChangeMode(mode newmode, QPushButton* but);
    bool TryRedo();
};

#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. UTILS_CLI.H

```
#ifndef UTILS_CLI_H
#define UTILS_CLI_H
#include <string>
#include <iostream>
#include "utils_headers.h"
#include "utils_tree.h"

class utils_cli
{
public:
    static int execute(int argc, char *argv[]);
private:
    utils_cli(){}
};

#endif // UTILS_CLI_H
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. UTILS_LINKED.H

```
#ifndef UTILS_LINKED_H
#define UTILS_LINKED_H
#include "ilist.h"
#include <QColor>

template <class T = int>
struct node
{
    node* right;
    T data;

    int height;
    float light;
    QColor color;
    node* left;

    node(T d = 0)
    {
        left = nullptr;
        right = nullptr;
        data = d;
        light = 1;
        color = QColor::fromRgb(255, 0, 0);
        height = 1;
    }
};

template <class T = int>
class utils_linked : public IList<T>
{
private:
    node<T>* head;
    node<T>* tail;

public:
    utils_linked();
    utils_linked(const utils_linked& copy);
    T operator[] (int index) override;
    T at (int index) override;
    void clean() override;
    void insert(int index, T element) override;
    T remove(int index) override;

    T back() override;
    void push_back(T element) override;
```

```
    T pop_back() override;

    T front() override;
    void push_front(T element) override;
    T pop_front() override;

    int size() override;
    bool empty() override;
    ~utils_linked();
};


template<class T>
utils_linked<T>::utils_linked()
{
    head = nullptr;
    tail = nullptr;
}

template <class T>
T utils_linked<T>::operator[] (int index)
{
    node<T>* t = head;
    for (int i = 0; i < index; i++)
    {
        t = t->right;
    }
    return t->data;
}

template <class T>
T utils_linked<T>::at (int index)
{
    return operator[](index);
}

template <class T>
void utils_linked<T>::clean ()
{
    head = nullptr;
    tail = nullptr;
}

template <class T>
void utils_linked<T>::insert(int index, T element)
{
    node<T>* n = new node<T>(element);
    if (empty())
```

```
    {
        head = n;
        tail = n;
    }
    else if (index == 0)
    {
        push_front(element);
    }
    else if (index == size())
    {
        push_back(element);
    }
    else
    {
        node<T>* t = head;
        for (int i = 0; i < index; i++)
        {
            t = t->right;
        }
        n->right = t;
        n->left = t->left;
        t->left->right= n;
        t->left = n;
    }
}

template<class T>
T utils_linked<T>::remove(int index)
{
    T res = at(index);
    if (index == 0)
    {
        pop_front();
    }
    else if (index == size() - 1)
    {
        pop_back();
    }
    else {
        node<T> *t = head;
        for (int i = 0; i < index; i++) {
            t = t->right;
        }
        t->left->right = t->right;
        t->right->left = t->left;
        delete t;
    }
    return res;
```

```
}

template<class T>
T utils_linked<T>::back()
{
    return tail->data;
}

template<class T>
void utils_linked<T>::push_back(T element)
{
    node<T>* n = new node<T>(element);
    if (empty())
    {
        head = n;
        tail = n;
    }
    else
    {
        tail->right = n;
        n->left = tail;
        tail = n;
    }
}

template<class T>
T utils_linked<T>::pop_back()
{
    T data;
    if (size() == 1) {
        if (head != nullptr) {
            data = head->data;
            delete head;
            head = nullptr;
        } else if (tail != nullptr) {
            data = tail->data;
            delete tail;
            tail == nullptr;
        }
    }
    else {
        node<T> *n = tail;
        tail = tail->left;
        tail->right = nullptr;
        data = n->data;
        delete n;
    }
    return data;
```

```
}

template<class T>
T utils_linked<T>::front()
{
    return head->data;
}

template<class T>
void utils_linked<T>::push_front(T element)
{
    node<T>* n = new node<T>(element);
    if (empty())
    {
        head = n;
        tail = n;
    }
    else
    {
        head->left = n;
        n->right = head;
        head = n;
    }
}

template<class T>
T utils_linked<T>::pop_front()
{
    T data;
    if (size() == 1) {
        if (head != nullptr) {
            data = head->data;
            delete head;
            head = nullptr;
        } else if (tail != nullptr) {
            data = tail->data;
            delete tail;
            tail == nullptr;
        }
    }
    else {
        node<T> *n = head;
        head = head->right;
        head->left = nullptr;
        data = n->data;
        delete n;
    }
    return data;
```

```
}


template<class T>
int utils_linked<T>::size()
{
    int i = 0;
    node<T>* t = head;
    while (t)
    {
        t = t->right;
        i++;
    }
    return i;
}

template<class T>
bool utils_linked<T>::empty()
{
    return !size();
}

template<class T>
utils_linked<T>::~utils_linked()
{
    node<T>* t = head;
    while (t)
    {
        delete t;
        t = t->right;
    }
}

#endif // UTILS_LINKED_H
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. UTILS_TREE.H

```
#ifndef UTILS_TREE_H
#define UTILS_TREE_H
#include <string>
#include "utils_linked.h"
#include "utils_vector.h"
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

template <class T = int>
class utils_tree
{
public:
    utils_tree(std::string& str);
    utils_tree(utils_tree<T>& copy);
    void clean();
    void insert(T data);
    void remove(T data);
    bool search(T key);
    bool is_bst();
    bool is_pyramid();
    int max_depth();
    ~utils_tree();
    node<T>* root;
    vector<node<T>*> undolist;
    vector<node<T>*> redolist;
    bool undo();
    bool redo();
    void save();
    void load();
    bool is_bst_stepped(utils_vector<node<T>*>& v, node<T>* n, T min, T max);
    bool is_pyramid_stepped(utils_vector<node<T>*>& v, node<T>* n, int max);
    void to_optimal();
    void insertbalanced(T data);
    void removebalanced(T data);
    void go_darker();
    std::string node_string();
    std::string node_string(node<T>* n);
private:
    bool parse_tree(node<T>*& n, std::string &s, int &i);
    void clean(node<T>* n);
    void insert(node<T>*& n, T data);
    void remove(node<T>*& n, T data);
```

```
    node<T>* search(node<T>* n, T key);
    bool is_bst(node<T>* n, T min, T max);
    bool is_pyramid(node<T>* n, int max);
    int max_depth(node<T>* n, int i);
    void get_data(utils_vector<T>& vec, node<T>* n);
    void go_darker(node<T>* n);
    //
public:
    int height(node<T>* n);
    int bfactor(node<T>* n);
    void fixheight(node<T>* n);
    node<T>* rotateleft(node<T>* n);
    node<T>* rotateright(node<T>* n);
    node<T>* balance(node<T>* n);
    node<T>* findmin(node<T>* n);
    node<T>* removemin(node<T>* n);
    node<T>* insertbalanced(node<T>* n, T data);
    node<T>* removebalanced(node<T>* n, T data);
    node<T>* save(node<T>* n);
    void go_red(node<T>* n);
};


template<class T>
utils_tree<T>::utils_tree(std::string &str)
    : root(new node<T>())
{
    int i = 0;
    if(parse_tree(root, str, i))
    {
        delete root;
        root = nullptr;
    }
    undolist = vector<node<T>*>();
    redolist = vector<node<T>*>();
}

template<class T>
utils_tree<T>::utils_tree(utils_tree<T> &copy)
{

}

template<class T>
void utils_tree<T>::clean()
{
    clean(root);
}
```

```
template<class T>
void utils_tree<T>::insert(T data)
{
    insert(root, data);
}

template<class T>
void utils_tree<T>::remove(T data)
{
    remove(root, data);
}

template<class T>
bool utils_tree<T>::search(T key)
{
    return search(root, key) != nullptr;
}

template<class T>
bool utils_tree<T>::is_bst()
{
    return is_bst(root->left, INT_MIN, root->data) && is_bst(root->right, root-
>data, INT_MAX);
    // Костыль. Заменить INT_MIN/MAX на гендеро-нейтральный тип.
}

template<class T>
bool utils_tree<T>::is_pyramid()
{
     return is_pyramid(root->left, root->data) && is_pyramid(root->right, root-
>data);
}

template<class T>
int utils_tree<T>::max_depth()
{
    return max_depth(root, 1);
}

template<class T>
utils_tree<T>::~utils_tree()
{ // He protec
    clean(); // He attak
} // He destroy

template<class T>
bool utils_tree<T>::undo()
```

```
{
    if (undolist.empty()) return false;
    redolist.push_back(save(root));
    root = undolist.back();
    undolist.pop_back();
    return true;
}

template<class T>
bool utils_tree<T>::redo()
{
    if (redolist.empty()) return false;
    undolist.push_back(save(root));
    root = redolist.back();
    redolist.pop_back();
    return true;
}

template<class T>
void utils_tree<T>::save()
{
    if (root)
    {
        undolist.push_back(save(root));
        redolist.clear();
    }
}

template<class T>
void utils_tree<T>::load()
{
    ifstream myfile;
    myfile.open("backup.txt");
    if (myfile.is_open())
    {
        string s;
        myfile >> s;
        int i = 0;
        root = new node<T>();
        undolist.clear();
        redolist.clear();
        parse_tree(root, s, i);
    }
}

template<class T>
bool  utils_tree<T>::is_bst_stepped(utils_vector<node<T>*>  &v,  node<T>  *n,  T
min, T max)
```

```
{
    if (!n) return true;
    v.push_back(n);
    if (n->data <= min || n->data >= max) return false;
      return is_bst_stepped(v, n->left, min, n->data) && is_bst_stepped(v, n->right, n->data, max);
}

template<class T>
bool utils_tree<T>::is_pyramid_stepped(utils_vector<node<T>*> &v, node<T> *n, int max)
{
    if (!n) return true;
    v.push_back(n);
    if (n->data >= max) return false;
     return is_pyramid_stepped(v, n->left, n->data) && is_pyramid_stepped(v, n->right, n->data);
}

template<class T>
void utils_tree<T>::to_optimal()
{
    utils_vector<int> datas = utils_vector<int>();
    get_data(datas, root);
    //datas.sort();
    delete root;
    root = new node<T>(datas[0]);
    for (int i = 1; i < datas.size(); i++)
    {
        insertbalanced(datas[i]);
    }
}

template<class T>
void utils_tree<T>::insertbalanced(T data)
{
    if (search(data)) return;
    if (!root)
    {
        root = new node<T>(data);
        root->color = QColor::fromRgb(255, 0, 255);
        return;
    }
    if (data < root->data) root->left = insertbalanced(root->left, data);
    else root->right = insertbalanced(root->right, data);
    root = balance(root);
}
```

```
template<class T>
void utils_tree<T>::removebalanced(T data)
{
    root = removebalanced(root, data);
}

template<class T>
void utils_tree<T>::go_darker()
{
    go_darker(root);
}

template<class T>
std::string utils_tree<T>::node_string()
{
    return "Tree: {" + node_string(root) + "}";
}

// PRIVATE

template<class T>
bool utils_tree<T>::parse_tree(node<T>*& n, std::string &s, int &i) {
    if (i >= s.size() || s[i] == ')')
    {
        delete n;
        n = nullptr;
        return false;
    }
    if (s[i] == '(')
    {
        i++;
    }
    if (s[i] == ')')
    {
        i++;
        delete n;
        n = nullptr;
        return false;
    }
    int num;
    int start = i;
    while (i != s.size() && s[i] != '(' && s[i] != ')')
    {
        i++;
    }
    try
    {
        num = stoi(s.substr(start, i - start));
```

```
    }
    catch (...)
    {
        return true;
    }
    n->data = num;
    n->left = new node<T>();
    n->right = new node<T>();
    if(parse_tree(n->left, s, i) || parse_tree(n->right, s, i)) return true;
    if (s[i] == ')')
    {
        i++;
    }
    return false;
}


template<class T>
void utils_tree<T>::clean(node<T> *n)
{
    if (!n) return;
    clean(n->left);
    clean(n->right);
    delete root;
}

template<class T>
void utils_tree<T>::insert(node<T>*& n, T data)
{
    if (!n)
    {
        n = new node<T>(data);
    }
    else if (n->data < data)
    {
        insert(n->left, data);
    }
    else if (n->data > data)
    {
        insert(n->right, data);
    }
}

template<class T>
void utils_tree<T>::remove(node<T>*& n, T data)
{
    if (!n) return;
    if (data < n->data)
```

```
    {
        remove(n->left, data);
    }
    else if (data > n->data)
    {
        remove(n->right, data);
    }
    else
    {
        if (!n->left && n->right)
        {
            node<T>* temp = n->right;
            delete n;
            n = temp;
        }
        else if (!n->right && n->left)
        {
            node<T>* temp = n->left;
            delete n;
            n = temp;
        }
        node<T>* min = n->right;
        if (!min->left)
        {
            n->right = nullptr;
        }
        else
        {
            node<T>* t = min;
            while(t->left->left)
            {
                t = n->left;
            }
            min = t->left;
            t->left = nullptr;
        }
        n->data = min->data;
        delete min;
    }
}

template<class T>
node<T>* utils_tree<T>::search(node<T> *n, T key)
{
    if (!n) return nullptr;
    if (n->data == key) return n;
    node<T>* l = search(n->left, key);
    if (l) return l;
```

```
    node<T>* r = search(n->right, key);
    if (r) return r;
    return nullptr;
}

template<class T>
bool utils_tree<T>::is_bst(node<T> *n, T min, T max)
{
    if (!n) return true;
    if (n->data <= min || n->data >= max) return false;
    return is_bst(n->left, min, n->data) && is_bst(n->right, n->data, max);
}

template<class T>
bool utils_tree<T>::is_pyramid(node<T> *n, int max)
{
    if (!n) return true;
    if (n->data >= max) return false;
    return is_pyramid(n->left, n->data) && is_pyramid(n->right, n->data);
}

template<class T>
int utils_tree<T>::max_depth(node<T> *n, int i)
{
    if (!n) return i;
    int l = max_depth(n->left, i + 1);
    int r = max_depth(n->right, i + 1);
    if (l > r) return l;
    else return r;
}

template<class T>
void utils_tree<T>::get_data(utils_vector<T> &vec, node<T> *n)
{
    if (!n) return;
    vec.push_back(n->data);
    get_data(vec, n->left);
    get_data(vec, n->right);
}

template<class T>
void utils_tree<T>::go_darker(node<T> *n)
{
    if (!n) return;
    n->light /= 1.05f;
     //n->color.setHsv(n->color.hue(), n->color.saturation(), n->color.value() /
1.05f);
    go_darker(n->left);
```

```
    go_darker(n->right);
}

template<class T>
std::string utils_tree<T>::node_string(node<T> *n)
{
    if (!n) return "";
     return "" + std::to_string(n->data) + "(" + node_string(n->left) + ")(" +
node_string(n->right) + ")";
}

//АВЛ

template<class T>
int utils_tree<T>::height(node<T> *n)
{
    return n ? n->height : 0;
}

template<class T>
int utils_tree<T>::bfactor(node<T> *n)
{
    return height(n->right) - height(n->left);
}

template<class T>
void utils_tree<T>::fixheight(node<T> *n)
{
    int hl = height(n->left);
    int hr = height(n->right);
    n->height = (hl > hr ? hl : hr) + 1;
}

template<class T>
node<T>* utils_tree<T>::rotateleft(node<T> *n)
{
    n->color.setRgb(255, 0, 0);
    n->light = 1;
    n->right->color.setRgb(255, 0, 0);
    n->right->light = 1;
    //n->right->left->color.setRgb(255, 0, 0);
    node<T>* p = n->right;
    n->right = p->left;
    p->left = n;
    fixheight(n);
    fixheight(p);
    return p;
}
```

```
template<class T>
node<T>* utils_tree<T>::rotateright(node<T> *n)
{
    n->color.setRgb(255, 0, 0);
    n->light = 1;
    n->left->color.setRgb(255, 0, 0);
    n->left->light = 1;
    //n->left->right->color.setRgb(255, 0, 0);
    node<T>* q = n->left;
    n->left = q->right;
    q->right = n;
    fixheight(n);
    fixheight(q);
    return q;
}

template<class T>
node<T>* utils_tree<T>::balance(node<T> *n)
{
    fixheight(n);
    if (bfactor(n) == 2)
    {
        if (bfactor(n->right) < 0)
            n->right = rotateright(n->right);
        return rotateleft(n);
    }
    if (bfactor(n) == -2)
    {
        if (bfactor(n->left) > 0)
            n->left = rotateleft(n->left);
        return rotateright(n);
    }
    return n;
}

template<class T>
node<T> *utils_tree<T>::findmin(node<T> *n)
{
    return n->left ? findmin(n->left) : n;
}

template<class T>
node<T> *utils_tree<T>::removemin(node<T> *n)
{
    if (!n->left) return n->right;
    n->left = removemin(n->left);
    return balance(n);
```

```
}

template<class T>
node<T> *utils_tree<T>::insertbalanced(node<T> *n, T data)
{
    if (!n)
    {
        node<T>* res = new node<T>(data);
        res->color = QColor::fromRgb(255, 0, 255);
        return res;
    }
    if (data < n->data) n->left = insertbalanced(n->left, data);
    else n->right = insertbalanced(n->right, data);
    return balance(n);
}

template<class T>
node<T> *utils_tree<T>::removebalanced(node<T> *n, T data)
{
    if (!n) return nullptr;
    if (data < n->data)
    {
        n->left = removebalanced(n->left,data);
    }
    else if (data > n->data)
    {
        n->right = removebalanced(n->right,data);
    }
    else
    {
        node<T>* l = n->left;
        node<T>* r = n->right;
        delete n;
        if (!r) return l;
        node<T>* min = findmin(r);
        min->right = removemin(r);
        min->left = l;
        return balance(min);
    }
    return balance(n);
}

template<class T>
node<T> *utils_tree<T>::save(node<T> *n)
{
    if (!n) return nullptr;
    node<T>* copy = new node<T>();
    copy->data = n->data;
```

```
    copy->left = save(n->left);
    copy->right = save(n->right);
    copy->color = n->color;
    copy->light = 1;
    return copy;
}

template<class T>
void utils_tree<T>::go_red(node<T> *n)
{
    if (!n) return;
    if (!n->left && !n->right) n->color = QColor::fromRgb(255, 0, 255);
    else n->color = QColor::fromRgb(255, 0, 0);
    go_red(n->left);
    go_red(n->right);
}


#endif // UTILS_TREE_H
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. UTILS_VECTOR.H

```
#ifndef UTILS_VECTOR_H
#define UTILS_VECTOR_H
#include "ilist.h"

template <class T = int>
class utils_vector : public IList<T>
{
private:
    T* array;
    int capacity;
    int count;
    void resize(int new_capacity);

public:
    utils_vector(int start_capacity = 4);
    utils_vector(const utils_vector& copy);
    T operator[] (int index) override;
    T at (int index) override;
    void clean() override;
    void insert(int index, T element) override;
    T remove(int index) override;

    T back() override;
    void push_back(T element) override;
    T pop_back() override;

    T front() override;
    void push_front(T element) override;
    T pop_front() override;

    int size() override;
    bool empty() override;
    ~utils_vector();
};

template<class T>
void utils_vector<T>::resize(int new_capacity)
{
    auto *arr = new T[count];
    for (int i = 0; i < count; ++i)
    {
        arr[i] = array[i];
    }
    delete [] array;
    array = new T[new_capacity];
```

```
    for (int i = 0 ; i < count; ++i)
    {
        array[i] = arr[i];
    }
    delete [] arr;
    capacity = new_capacity;
}

template<class T>
utils_vector<T>::utils_vector(int start_capacity)
{
    capacity = start_capacity;
    count = 0;
    array = new T[capacity];
}

template<class T>
utils_vector<T>::utils_vector(const utils_vector & copy) :
    count(copy.count),
    capacity(copy.capacity)
{
    array = new T[capacity];
    for (int i = 0; i < count; ++i)
    {
        *(array + i) = *(copy.array + i);
    }
}

template <class T>
T utils_vector<T>::operator[] (int index)
{
    return array[index];
}

template <class T>
T utils_vector<T>::at (int index)
{
    return operator[](index);
}

template <class T>
void utils_vector<T>::clean ()
{
    count = 0;
}

template <class T>
void utils_vector<T>::insert(int index, T element)
```

```
{
    if (capacity == count)
    {
        resize(count + 8);
    }
    if (count > 0) {
        for (int i = count; i > index; i--)
        {
            array[i] = array[i - 1];
        }
    }
    count++;
    array[index] = element;
}

template<class T>
T utils_vector<T>::remove(int index)
{
    auto temp = array[index];
    for (int i = index; i < count - 1; i++)
    {
        array[i] = array[i + 1];
    }
    count--;
    return temp;
}

template<class T>
T utils_vector<T>::back()
{
    return array[count - 1];
}

template<class T>
void utils_vector<T>::push_back(T element)
{
    if (capacity == count)
    {
        resize(count + 8);
    }
    array[count] = element;
    count++;
}

template<class T>
T utils_vector<T>::pop_back()
{
    return array[--count];
```

```
}

template<class T>
T utils_vector<T>::front()
{
    return *array;
}

template<class T>
void utils_vector<T>::push_front(T element)
{
    insert(0, element);
}

template<class T>
T utils_vector<T>::pop_front()
{
    return remove(0);
}

template<class T>
int utils_vector<T>::size()
{
    return count;
}

template<class T>
bool utils_vector<T>::empty()
{
    return !count;
}

template<class T>
utils_vector<T>::~utils_vector()
{
    delete [] array;
}

#endif //VECTOR_VECTOR_H
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

```
#include "utils_cli.h"

int utils_cli::execute(int argc, char *argv[])
{
    std::cout << "Starting in console mode..." << std::endl;
    utils_tree<int>* tree = nullptr;
    for(int i = 0; i < argc; i++) {
          if(!strcmp("console", argv[i]) || !strcmp("-console", argv[i]) || !
strcmp("-c", argv[i])) continue;
        std::cout << "Tree found: " << argv[i] << std::endl;
        std::string s = argv[i];
        tree = new utils_tree<int>(s);
    }
    if (!tree)
    {
        std::cout << "Thee is empty" << std::endl;
        return 1;
    }
    std::cout << tree->node_string() << std::endl;
    std::cout << "Insertion: 5" << std::endl;
    tree->insertbalanced(5);
    std::cout << tree->node_string() << std::endl;
    std::cout << "Insertion: 7" << std::endl;
    tree->insertbalanced(7);
    std::cout << tree->node_string() << std::endl;
    std::cout << "Insertion: 13" << std::endl;
    tree->insertbalanced(13);
    std::cout << tree->node_string() << std::endl;
    std::cout << "Removing: 4" << std::endl;
    tree->removebalanced(4);
    std::cout << tree->node_string() << std::endl;
    std::cout << "Removing: 13" << std::endl;
    tree->removebalanced(13);
    std::cout << tree->node_string() << std::endl;
    std::cout << "Removing: 8" << std::endl;
    tree->removebalanced(8);
    std::cout << tree->node_string() << std::endl;

//    char com;
//    do
//    {
//        std::cout << "1 - insert" << std::endl << "2 - remove" << std::endl;
//        std::cin >> com;
//        if (com == '1') {
//            std::cout << "What to insert?" << std::endl;
```

```
//            int i;
//            std::cin >> i;
//            tree->insertbalanced(i);
//        }
//        else if (com == '2') {
//            std::cout << "What to remove?" << std::endl;
//            int i;
//            std::cin >> i;
//            tree->removebalanced(i);
//        }
//        else {
//            std::cout << "Unknown command" << std::endl;
//        }
//        std::cout << tree->node_string() << std::endl;
//    } while (com != 'e');
    delete tree;
    return 0;
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QShortcut>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    tree(nullptr),
    state(0),
    locked(false),
    lockedUpd(false),
    stepped_mode(empty),
    ui(new Ui::MainWindow),
    mainGraphicsScene(new QGraphicsScene())
{
    ui->setupUi(this);
    ui->graphicsView->setScene(mainGraphicsScene);
    QMainWindow::showMaximized();
    QColor color = QColor(203, 119, 47);

    pen.setColor(color);
    brush.setColor(color);
    font.setFamily("Roboto");
    pen.setWidth(3);

    srand(static_cast<unsigned long>(time(nullptr)));
    QTimer *timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(updateTreeColor()));
    timer->start(100);
    stack = utils_linked<node<int>*>();
    ui->horizontalSlider->setValue(2);
    ui->graphicsView->resetTransform();
    ui->graphicsView->scale(1.0 / 2, 1.0 / 2);
                new    QShortcut(QKeySequence(Qt::CTRL    +    Qt::Key_Z),    this,
SLOT(on_butUndo_clicked()));
                new    QShortcut(QKeySequence(Qt::CTRL    +    Qt::Key_X),    this,
SLOT(on_butRedo_clicked()));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::UpdateGraphics(bool save)
```

```
{
    if (lockedUpd) return;
    lockedUpd = true;
    switch (stepped_mode) {
    case bst:
        ui->stateLabel->setText("bst");
        break;
    case empty:
        ui->stateLabel->setText("empty");
        break;
    case pyramid:
        ui->stateLabel->setText("pyramid");
        break;
    case step_add:
        ui->stateLabel->setText("step_add");
        break;
    case step_remove:
        ui->stateLabel->setText("step_remove");
        break;
    }
    if (!tree || tree->redolist.empty()) ui->butRedo->setEnabled(false);
    else ui->butRedo->setEnabled(true);
    if (!tree || tree->undolist.empty()) ui->butUndo->setEnabled(false);
    else ui->butUndo->setEnabled(true);
    mainGraphicsScene->clear();
    if (!tree) return;
    if (save) {
        ofstream myfile;
        myfile.open("backup.txt");
        myfile << tree->node_string(tree->root) + "\n";
        myfile.close();
    }
    DrawNode(tree->root, tree->max_depth());
    lockedUpd = false;
}

void MainWindow::on_butFile_clicked()
{
    std::string inputStr;
      QString fileName = QFileDialog::getOpenFileName(this, "Open TXT File",
QDir::homePath(), "TXT text (*.txt);;All Files (*)");
    if (fileName == nullptr)
    {
        QMessageBox::warning(this, "Warning", "File name is empty");
        return;
    }
    QFile file(fileName);
    if(file.open(QIODevice::ReadOnly | QIODevice::Text)) {
```

```
        QTextStream stream(&file);
          foreach (QString i ,QString(stream.readAll()).split(QRegExp("[ \t]"),
QString::SkipEmptyParts))
        inputStr.append(i.toUtf8().constData());
    }
    if(inputStr.empty())
        return;
    file.close();
    ui->input->setText(QString::fromUtf8(inputStr.c_str()));
}

void MainWindow::on_butGenerate_clicked()
{
    std::string readingStr;
    if (ui->input->text().isEmpty())
    {
        QMessageBox::warning(this, "Warining!", "Input is empty. Applying test:
13(8(6(3(1)(4))(7))(10(9)(11)))(14))");
        ui->input->setText("13(8(6(3(1)(4))(7))(10(9)(11)))(14))");
    }
    QString tempInp = ui->input->text();
    QTextStream stream(&tempInp);
       foreach  (QString  i,  QString(stream.readAll()).split(QRegExp("[  \t]"),
QString::SkipEmptyParts))
        readingStr.append(i.toUtf8().constData());
    tree = new utils_tree<int>(readingStr);
    is_bst = tree->is_bst_stepped(bst_stepped, tree->root, INT_MIN, INT_MAX);
         is_pyramid  =  tree->is_pyramid_stepped(pyramid_stepped,  tree->root,
INT_MAX);
    ui->balancedLabel->setText("Binary tree");
    ChangeMode(empty, nullptr);
    UpdateGraphics();
}

void MainWindow::on_butGenerateOptimal_clicked()
{
    if (!tree) on_butGenerate_clicked();
    if (!tree || tree->max_depth() <= 2) return;
    tree->to_optimal();
    ui->balancedLabel->setText("AVL tree");
    ChangeMode(empty, nullptr);
    UpdateGraphics();
}

void MainWindow::on_butRun_clicked()
{
    if (locked || !tree) return;
    locked = true;
```

```
    if (is_bst) QMessageBox::information(this, "BST?", "    YES    ");
    else QMessageBox::warning(this, "BST?", "    NO    ");
    if (is_pyramid) QMessageBox::information(this, "Pyramid?", "    YES    ");
    else QMessageBox::warning(this, "Pyramid?", "    NO    ");
    UpdateGraphics();
    locked = false;
}

void MainWindow::DrawNode(node<int> *n, int maxdepth, int depth, int x, int y)
{
    if (n == nullptr) return;
    int offset = pow(2, maxdepth + 3) / pow(2, depth);
    if (n->left) mainGraphicsScene->addLine(x + 32, y + 32, x - offset + 32, y
+ 64 + 32, pen);
    if (n->right) mainGraphicsScene->addLine(x + 32, y + 32, x + offset + 32, y
+ 64 + 32, pen);
    QColor c = n->color;
    c.setHsvF(c.hueF(), c.saturationF(), n->light);
    QBrush brush(c);
        color.setRgb(0,  255  *  (depth/(float)maxdepth),  255  *  ((maxdepth  -
depth)/(float)maxdepth));
    QPen pen(color, 3);
    mainGraphicsScene->addEllipse(x, y, 64, 64, pen, brush);
    QGraphicsTextItem *numb = new QGraphicsTextItem();
    numb->setPlainText(QString::number(n->data));
    numb->setDefaultTextColor(Qt::white);
    numb->setScale(2);
    numb->setPos(x + 16, y + 8);
    mainGraphicsScene->addItem(numb);
    DrawNode(n->left, maxdepth, depth + 1, x - offset, y + 64);
    DrawNode(n->right, maxdepth, depth + 1, x + offset, y + 64);
}

void MainWindow::SetActiveButtons(bool mode)
{
    ui->butAdd->setEnabled(mode);
    ui->butStepAdd->setEnabled(mode);
    ui->butRemove->setEnabled(mode);
    ui->butStepRemove->setEnabled(mode);
    ui->butFile->setEnabled(mode);
    ui->butStepBst->setEnabled(mode);
    ui->butStepPyramid->setEnabled(mode);
    ui->butRun->setEnabled(mode);
    ui->butGenerate->setEnabled(mode);
    ui->butGenerateOptimal->setEnabled(mode);

    ui->inputElement->setEnabled(mode);
    ui->input->setEnabled(mode);
```

```
}

bool MainWindow::ReadElement()
{
    if (!tree) return false;
    if (ui->inputElement->text().isEmpty())
    {
              QMessageBox::warning(this, "Warining!", "Input elemet is empty.
Applying random data");
        ui->inputElement->setText(QString::number(rand() % 20));
    }
    bool ok;
    int res = ui->inputElement->text().toInt(&ok);
    if (!ok)
    {
             QMessageBox::warning(this, "Warining!", "Input elemet isn't a
number!");
        ui->inputElement->setText("");
        return false;
    }
    current = res;
    return true;
}

void MainWindow::ChangeMode(mode newmode, QPushButton* but)
{
    if (newmode == stepped_mode)
    {
        tree->save();
    }
    else if (newmode == empty)
    {
        SetActiveButtons(true);
    }
    else
    {
        tree->go_red(tree->root);
        tree->undolist.clear();
        tree->redolist.clear();
        SetActiveButtons(false);
        if (but) but->setEnabled(true);
    }
    if (tree->undolist.empty()) ui->butUndo->setEnabled(false);
    else ui->butUndo->setEnabled(true);
    if (tree->redolist.empty()) ui->butRedo->setEnabled(false);
    else ui->butRedo->setEnabled(true);
    stepped_mode = newmode;
}
```

```
bool MainWindow::TryRedo()
{
    if (!tree->redolist.empty())
    {
        tree->redo();
        UpdateGraphics();
        return true;
    }
    return false;
}


void MainWindow::on_horizontalSlider_sliderMoved(int position)
{
    ui->graphicsView->resetTransform();
    ui->graphicsView->scale(1.0 / position, 1.0 / position);
}

void MainWindow::on_butStepBst_clicked()
{
    ChangeMode(bst, ui->butStepBst);
    if (bst_stepped.empty())
    {
        if (is_bst) QMessageBox::information(this, "BST?", "    YES    ");
        else QMessageBox::warning(this, "BST?", "    NO    ");
        stepped_mode = empty;
        SetActiveButtons(true);
    }
    else
    {
        node<int>* n = bst_stepped.pop_front();
        n->color.setRgb(0, 255, 0);
        n->light = 1;
    }
    UpdateGraphics();
}

void MainWindow::on_butStepPyramid_clicked()
{
    ChangeMode(pyramid, ui->butStepPyramid);
    if (pyramid_stepped.empty())
    {
          if (is_pyramid) QMessageBox::information(this, "Pyramid?", "     YES
");
        else QMessageBox::warning(this, "Pyramid?", "    NO    ");
        ChangeMode(empty, nullptr);
    }
```

```
    else
    {
        node<int>* n = pyramid_stepped.pop_front();
        n->color.setRgb(0, 0, 255);
        n->light = 1;
    }
    UpdateGraphics();
}

void MainWindow::on_butAdd_clicked()
{
    if (!tree) return;
    ChangeMode(empty, nullptr);
    if (!ReadElement()) return;
    if (tree->search(current))
    {
        QMessageBox::information(this, "Insertion", "Duplicate");
    }
    else
    {
        tree->insertbalanced(current);
    }
    ui->inputElement->setText("");
    UpdateGraphics();

}

void MainWindow::on_butStepAdd_clicked()
{
    if (!ReadElement()) return;
    if (!tree) return;
    ChangeMode(step_add, ui->butStepAdd);
    if (TryRedo()) return;
    node<int>* n;
    switch (state)
    {
        case 0: // Init
            stack.clean();
            state_node = tree->root;
            state = 1;
            break;
        case 1: // Down
            stack.push_back(state_node);
            state_node->color.setRgb(0, 255, 0);
            state_node->light = 1;
            if (current < state_node->data)
            {
                state_node = state_node->left;
```

```
                if (!state_node)
                {
                    stack.back()->left = new node<int>(current);
                    state = 2;
                }
            }
            else if (current > state_node->data)
            {
                state_node = state_node->right;
                if (!state_node)
                {
                    stack.back()->right = new node<int>(current);
                    state = 2;
                }
            }
            else
            {
                QMessageBox::information(this, "Insertion", "Duplicate");
                ChangeMode(empty, nullptr);
                state = 0;
            }
            break;
        case 2: // Up
            if (stack.size() <= 1)
            {
                QMessageBox::information(this, "Insertion", "Completed");
                ChangeMode(empty, nullptr);
                state = 0;
                break;
            }
            n = stack.pop_back();
            n->color.setRgb(0, 0, 255);
            n->light = 1;
            tree->fixheight(n);
            if (tree->bfactor(n) == 2)
            {
                        if (tree->bfactor(n->right) < 0) n->right = tree-
>rotateright(n->right);
                    if (n == stack.back()->left) stack.back()->left = tree-
>rotateleft(n);
                else stack.back()->right = tree->rotateleft(n);
            }
            if (tree->bfactor(n) == -2)
            {
                  if (tree->bfactor(n->left) > 0) n->left = tree->rotateleft(n-
>left);
                if (!stack.back())
                {
```

```
                    QMessageBox::critical(this, "Insertion", "NULL");
                }
                      if (n == stack.back()->left) stack.back()->left = tree-
>rotateright(n);
                else stack.back()->right = tree->rotateright(n);
            }
            break;
    }
    UpdateGraphics();
}

void MainWindow::on_butRemove_clicked()
{
    if (!tree) return;
    ChangeMode(empty, nullptr);
    if (!ReadElement()) return;
    if (tree->search(current))
    {
        tree->removebalanced(current);
        ui->inputElement->setText("");
        UpdateGraphics();
    }
    else
    {
                QMessageBox::information(this, "Removing", "There is no " +
QString::number(current) + " element.");
    }
}

// FIX IT
void MainWindow::on_butStepRemove_clicked()
{
    if (!ReadElement()) return;
    ChangeMode(step_remove, ui->butStepRemove);
    if (TryRedo()) return;
    node<int>* l;
    node<int>* r;
    node<int>* min;
    switch (state)
    {
        case 0: // Init
            stack.clean();
            state_node = tree->root;
            state = 1;
            break;
        case 1: // Down
            if (!state_node)
            {
```

```
                QMessageBox::information(this, "Removing", "No element");
                ChangeMode(empty, nullptr);
                state = 0;
            }
            else
            {
                if (current < state_node->data)
                {
                    stack.push_back(state_node);
                    state_node->color.setRgb(0, 255, 0);
                    state_node->light = 1;
                    state_node = state_node->left;
                }
                else if (current > state_node->data)
                {
                    stack.push_back(state_node);
                    state_node->color.setRgb(0, 255, 0);
                    state_node->light = 1;
                    state_node = state_node->right;
                }
                else
                {
                    l = state_node->left;
                    r = state_node->right;
                    if (!r)
                    {
                        if (state_node == tree->root) tree->root = l;
                                    else if (state_node == stack.back()->left)
stack.back()->left = l;
                        else stack.back()->right = l;
                    }
                    else
                    {
                        min = tree->findmin(r);
                        min->right = tree->removemin(r);
                        min->left = l;
                              if (state_node == tree->root) tree->root = tree-
>balance(min);
                                    else if (state_node == stack.back()->left)
stack.back()->left = tree->balance(min);
                        else stack.back()->right = tree->balance(min);
                    }
                    delete state_node;
                    state = 2;
                }
            }
            break;
        case 2: // Up
```

```
            if (stack.size() <= 0)
            {
                QMessageBox::information(this, "Removing", "Completed");
                ChangeMode(empty, nullptr);
                state = 0;
            }
            else
            {
                l = stack.pop_back();
                l->color.setRgb(0, 0, 255);
                l->light = 1;
                if (l == tree->root) tree->root = tree->balance(l);
                  else if (l == stack.back()->left) stack.back()->left = tree-
>balance(l);
                else stack.back()->right = tree->balance(l);
            }
            break;
    }
    UpdateGraphics();
}

void MainWindow::updateTreeColor()
{
    if (!tree || locked || lockedUpd) return;
    tree->go_darker();
    UpdateGraphics(false);
}

void MainWindow::on_butUndo_clicked()
{
    tree->undo();
    UpdateGraphics();
}

void MainWindow::on_butRedo_clicked()
{
    tree->redo();
    UpdateGraphics();
}

void MainWindow::on_butLoad_clicked()
{
    string s = "";
    tree = new utils_tree<int>(s);
    tree->load();
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "strstrtester.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

    void on_GPushOk_clicked();

    void on_HPushOk_clicked();

    void on_MPushOk_clicked();

    void on_OutSetPlot_toggled(bool checked);

    void on_OutSaveFile_clicked();

private:
    void setupPlot();
    Ui::MainWindow *ui;
    StrStrWorker ssw;
};
#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ Л

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1245</width>
    <height>727</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>Sakharov_lr4</string>
  </property>
  <property name="windowOpacity">
   <double>1.000000000000000</double>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(43, 43, 43);
font: 14pt &amp;quot;Consolas&amp;quot;;
color: rgb(203, 119, 47);
border-color: rgb(1, 1, 1);
border-width : 1.2px;
border-style: ridge;
margin-left: 4px;
margin-right: 4px;
padding:2px;</string>
  </property>
  <property name="iconSize">
   <size>
    <width>0</width>
    <height>0</height>
   </size>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QVBoxLayout" name="verticalLayout_2">
    <item>
     <layout class="QVBoxLayout" name="verticalLayout">
```

```
      <item>
       <layout class="QHBoxLayout" name="horizontalLayout">
        <property name="spacing">
         <number>16</number>
        </property>
        <item>
         <widget class="QPushButton" name="butLoad">
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Fast load</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butFile">
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>From file</string>
          </property>
         </widget>
        </item>
        <item>
```

```
          <widget class="QLineEdit" name="input">
           <property name="styleSheet">
            <string notr="true">color: rgb(97, 150, 71);</string>
           </property>
           <property name="inputMask">
            <string/>
           </property>
           <property name="placeholderText">
            <string>Input your tree here... example - 4(2(3)(1))(6(5))</string>
           </property>
          </widget>
         </item>
         <item>
          <widget class="QPushButton" name="butGenerate">
           <property name="styleSheet">
            <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
           </property>
           <property name="text">
            <string>Generate</string>
           </property>
          </widget>
         </item>
         <item>
          <widget class="QPushButton" name="butGenerateOptimal">
           <property name="styleSheet">
            <string>QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
```

```
  </property>
  <property name="text">
   <string>Generate optimal</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="balancedLabel">
  <property name="styleSheet">
   <string notr="true">border-width: 0px;</string>
  </property>
  <property name="text">
   <string>Empty tree</string>
  </property>
 </widget>
</item>
<item>
 <spacer name="horizontalSpacer_2">
  <property name="orientation">
   <enum>Qt::Horizontal</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>40</width>
    <height>20</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <widget class="QLabel" name="stateLabel">
  <property name="styleSheet">
   <string notr="true">border-width: 0px;</string>
  </property>
  <property name="text">
   <string>State</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="label">
  <property name="styleSheet">
   <string notr="true">border-width: 0px;</string>
  </property>
  <property name="text">
   <string>Zoom:</string>
  </property>
 </widget>
```

```
          </item>
          <item>
           <widget class="QSlider" name="horizontalSlider">
            <property name="sizePolicy">
             <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
              <horstretch>0</horstretch>
              <verstretch>0</verstretch>
             </sizepolicy>
            </property>
            <property name="minimum">
             <number>1</number>
            </property>
            <property name="maximum">
             <number>10</number>
            </property>
            <property name="value">
             <number>1</number>
            </property>
            <property name="orientation">
             <enum>Qt::Horizontal</enum>
            </property>
            <property name="tickPosition">
             <enum>QSlider::NoTicks</enum>
            </property>
           </widget>
          </item>
         </layout>
        </item>
        <item>
         <widget class="QGraphicsView" name="graphicsView">
          <property name="styleSheet">
           <string notr="true">background-color: rgb(50, 50, 50);</string>
          </property>
         </widget>
        </item>
        <item>
         <layout class="QHBoxLayout" name="horizontalLayout_2">
          <item>
           <widget class="QPushButton" name="butStepBst">
            <property name="maximumSize">
             <size>
              <width>0</width>
              <height>16777215</height>
             </size>
            </property>
            <property name="styleSheet">
             <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
```

```
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Step Bst</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butStepPyramid">
          <property name="maximumSize">
           <size>
            <width>0</width>
            <height>16777215</height>
           </size>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Step Pyramid</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butRun">
          <property name="maximumSize">
           <size>
            <width>0</width>
```

```
            <height>16777215</height>
           </size>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Run</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butUndo">
          <property name="enabled">
           <bool>false</bool>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Undo</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butRedo">
```

```
          <property name="enabled">
           <bool>false</bool>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Redo</string>
          </property>
         </widget>
        </item>
        <item>
         <spacer name="horizontalSpacer">
          <property name="orientation">
           <enum>Qt::Horizontal</enum>
          </property>
          <property name="sizeHint" stdset="0">
           <size>
            <width>40</width>
            <height>20</height>
           </size>
          </property>
         </spacer>
        </item>
        <item>
         <widget class="QLabel" name="label_2">
          <property name="styleSheet">
           <string notr="true">border-width: 0px;</string>
          </property>
          <property name="text">
           <string>AVL:</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QLineEdit" name="inputElement">
          <property name="styleSheet">
```

```
           <string notr="true">color: rgb(97, 150, 71);</string>
          </property>
          <property name="placeholderText">
           <string>Input element...</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butAdd">
          <property name="styleSheet">
           <string>QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Add</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butStepAdd">
          <property name="styleSheet">
           <string>QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Add Step</string>
          </property>
         </widget>
```

```
        </item>
        <item>
         <widget class="QPushButton" name="butRemove">
          <property name="styleSheet">
           <string>QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Remove</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="butStepRemove">
          <property name="styleSheet">
           <string>QPushButton
{ background-color: rgb(43, 43, 43); }
QPushButton:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:disabled
{ background-color: rgb(27, 27, 27) }
QPushButton:focus:pressed
{ background-color: rgb(169, 183, 198); }
QPushButton:focus
{ background-color: rgb(43, 43, 43) }
QPushButton:hover
{ background-color:  rgb(57, 57, 57)}</string>
          </property>
          <property name="text">
           <string>Remove Step</string>
          </property>
         </widget>
        </item>
       </layout>
      </item>
     </layout>
    </item>
   </layout>
```

```
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

ПРИЛОЖЕНИЕ М

ИСХОДНЫЙ КОД ПРОГРАММЫ. SAHKAROV_LR5.PRO

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = Sakharov_lr51
TEMPLATE = app
DEFINES += QT_DEPRECATED_WARNINGS
CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp \
    utils_cli.cpp

HEADERS += \
        mainwindow.h \
    ilist.h \
    utils_linked.h \
    utils_vector.h \
    utils_cli.h \
    utils_headers.h \
    utils_tree.h

FORMS += \
        mainwindow.ui

qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```