ММИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр. 9384 _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучение структуры бинарных деревьев и написание программы с использованием бинарных деревьев

Задание.

Формулу вида

<формула> ::=<терминал>|(<формула><знак><формула>)

<знак> ::= +|-|*

<терминал> ::= 0|1|…|9|a|b|…|z

Можно представить в виде бинарного дерева с элементами типа Elem = char согласно следующим правилам:

- формула из одного терминала представляется деревом из одной вершины с этим терминалом;

- формула вида (f1 s f2) представляется деревом, в котором корень – это знак s, а левое и правое поддеревья – соответствующим представлением формул f1 и f2.

Требуется:

- для заданной формулы f построить дерево-формулу t;
- для заданного дерева-формулы t напечатать соответствующую формулу f.
- построить дерево-формулу t
- упростить дерево-формулу t, выполнив в нем все операции вычитания, в которых уменьшаемое и вычитаемое – цифры. Результат вычисления – цифра или формула вида (0 - цифра)

Выполнение работы.

Для создания дерева, был реализован шаблонный класс BinaryTree.

Для создания дерева-формулы был описан вспомогательный класс maketree, реализующий преобразование формулы в дерево-формулу. Для разделения символов была описана функция split.

Функция createTree с помощью рекурсии реализует заполнение дерева.

Функция printTree, принимает на вход дерево и выводит формулу в формате ЛПК.

Для выполнения упрощения дерева-формулы, описана функция calcTree, производящая операцию вычитания с числами.

Также была реализована отрисовка дерева с помощью вспомогательных библиотек Qt.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a+b-c	ab+c-	Формула f в формате ЛПК
2.	b-(4-3)	b1-	Выполнено упрощение дерево формулы
3.	b-(3-4)	b01-	Выполнено упрощение дерево формулы. Результат отрицательное число

Выводы.

В ходе выполнения лабораторный работы были изучены бинарные деревья, а также была реализована программа выполняющее создание дерево-формулы по заданной функции.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>
#include "tree.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_draw_clicked();

private:
    Ui::MainWindow *ui;
    QGraphicsScene *scene;
```

```
    void treeDrawer(QGraphicsScene *&scene, BinaryTree<char> *tr, int w, int
h, int depth);
};
#endif // MAINWINDOW_H
```

Название файла: maketree.h

```
#ifndef MAKETREE_H
#define MAKETREE_H


#include "tree.h"

class maketree
{
public:
    maketree();
    BinaryTree<char> *createTree(std::string str);
    char slit(std::string str, std::string &data1, std::string &data2);
    void printTree(BinaryTree<char> *symb, std::string& output, int offset);
    void calcTree(BinaryTree<char> *tr);

private:
};

#endif // MAKETREE_H
```

Название файла: tree.h

```
#ifndef TREE_H
#define TREE_H
```

```
#include <string>

template <class T>
class BinaryTree
{
public:
    BinaryTree() {};
    BinaryTree(T value) {
        this->data = value;
    }
    ~BinaryTree(){
        delete left;
        delete right;
    }

    T getData()
    {
        return this->data;
    }

    BinaryTree* left;
    BinaryTree* right;
    T data;
private:
};

#endif // TREE_H
```

Название файла: main.cpp

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

```
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Название файла: mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "tree.h"
#include "maketree.h"
#include <iostream>
#include <string>
#include <QGraphicsTextItem>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    scene = new QGraphicsScene;
    ui->graphicsView->setScene(scene);
}

MainWindow::~MainWindow()
```

```
{
    delete ui;
}

void MainWindow::on_draw_clicked()
{
    std::string str = qPrintable(ui->input->text());
    BinaryTree<char> *tr = new BinaryTree<char>();
    maketree *qt = new maketree();
    tr = qt->createTree(str);

    if (ui->checkBox->isChecked())
        qt->calcTree(tr);

    scene->clear();
    treeDrawer(scene, tr, 0, 0, 20);
    std::string output = "";
    qt->printTree(tr, output, 0);
    ui->output->setText(QString::fromStdString(output));
}

void MainWindow::treeDrawer(QGraphicsScene *&scene, BinaryTree<char>
*tr, int w, int h, int depth)
{
    if (tr == nullptr)
        return ;
    QString text;
    text = tr->getData();
    QGraphicsTextItem *let = new QGraphicsTextItem;
    let->setPos(w, h);
```

```
    let->setPlainText(text);
    scene->addEllipse(w-7, h, 15*5/2, 15*5/2);
    if (tr->left != nullptr)
        scene->addLine(w+15/2, h+15, w-(depth/2)*15+15/2, h+70+15);
    if (tr->right != nullptr)
        scene->addLine(w+15/2, h+15, w+(depth/2)*15+15/2, h+70+15);
    scene->addItem(let);
    treeDrawer(scene, tr->left, w-(depth/2)*15, h+70, depth/2);
    treeDrawer(scene, tr->right, w+(depth/2)*15, h+70, depth/2);
}
```

Название файла:maketree.cpp

```
#include "maketree.h"
#include <iostream>
#include <string>

maketree::maketree()
{

}

BinaryTree<char> *maketree::createTree(std::string str)
{
    BinaryTree<char> *symb = new BinaryTree<char>();
    std::string str1 ="";
    std::string str2 = "";
    if (str[1] != '\0') {
      symb = new BinaryTree<char>(slit(str, str1, str2));
    }
```

```
    else {
        symb = new BinaryTree<char>(str[0]);
        symb->left = nullptr;
        symb->right = nullptr;
        return symb;
    }
    symb->left = createTree(str1);
    symb->right = createTree(str2);
    return symb;
}

char maketree::slit(std::string str, std::string &data1, std::string &data2)
{
    unsigned int ind_now = 0;
    bool open_bracket = false, symval = false;;
    for(unsigned int i = 0; i < str.length(); i++) {
        if (str[i] == '(')
            open_bracket = true;
        if (str[i] == ')')
            open_bracket = false;
        if (!open_bracket) {
            if (str[i] == '-' || str[i] == '+') {
                symval = true;
                ind_now = i;
            }
            if (str[i] == '*')
            {
                if (!symval){
                symval = true;
                ind_now = i;
```

```
                }
            }
        }
    }
    for(unsigned int i = 0; i < str.length(); i++) {
        if (i < ind_now)
            data1 += str[i];
        if (i > ind_now)
            data2 += str[i];
    }
    if (data1[0] == '(' && data1[data1.length()-1] == ')'){
        data1.erase(0,1);
        data1.erase(data1.length()-1);
    }
    if (data2[0] == '(' && data2[data2.length()-1] == ')'){
        data2.erase(0,1);
        data2.erase(data2.length()-1);
    }
    return str[ind_now];
}

void maketree::printTree(BinaryTree<char> *symb, std::string &output, int offset)
{
    if(symb != nullptr){
        printTree(symb->left, output, offset);
        printTree(symb->right, output, offset);
        output+= symb->data;
    }
}
```

```
void maketree::calcTree(BinaryTree<char> *tr)
{
   if(tr->left != nullptr && tr->right != nullptr)
   {

      if (tr->getData() == '-')
      {
         calcTree(tr->left);
         calcTree(tr->right);
         if (tr->left->getData() < '0' || tr->left->getData() > '9' || \
            tr->right->getData() < '0' || tr->right->getData() > '9') return;
         int k = tr->left->getData()-tr->right->getData();
         if (k < 0 )
         {
            tr->left->data = 0 + 48;
            tr->right->data = abs(k) + 48;
            return;
         }
         tr->data = k + 48;
         std::cout << (tr->left->getData()-tr->right->getData()) << std::endl;
         tr->left = nullptr;
         tr->right = nullptr;
         return;
      }
         calcTree(tr->left);
         calcTree(tr->right);
   }
}
```