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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: «Бинарные деревья»

Студентка гр. 8381 Бердникова А.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы

Изучить стек и методы работы с ним на примере написания программы по заданию данной лабораторной работы на языке программирования C++ в фреймворке Qt.

Основные теоретические положения

Бинарное дерево - иерархическая структура данных, в которой каждый узел имеет не более двух потомков (детей). Как правило, первый называется родительским узлом, а дети называются левым и правым наследниками. Двоичное дерево не является упорядоченным ориентированным деревом.

Существует следующее рекурсивное определение двоичного дерева: <дерево> ::=(<данные> <дерево> <дерево>) || null

То есть двоичное дерево либо является пустым, либо состоит из данных и двух поддеревьев (каждое из которых может быть пустым). Важным фактом является то, что каждое поддерево в свою очередь тоже является деревом. Если у некоторого узла оба поддерева пустые, то он называется листовым узлом (листовой вершиной) или конечным (терминальным) узлом.

Задание

С помощью построения дерева-формулы t преобразовать заданную формулу t из постфиксной формы в инфиксную.

Ход работы

Программа имеет графический интерфейс.т графический интерфейс.

Программа получает на вход строку, содержащего исходное выражение в постфиксной форме, строит по нему дерево, которое используется для преобразования выражения в инфиксную форму.

Программа имеет функцию вывода построенного дерева, при желании пользователя.

Программа предполагает, что выражение введено правильно, в противном случае она может работать не корректно.

Описание программы:

Программа считывает исходные данные введённые пользователем. Затем происходит посимвольная обработка входной строки с её конца и построение бинарного дерева.

После того как элемент был добавлен в дерево, он проверяется на то, является ли он цифрой или арифметической операцией. Если элемент является операцией, то следующий символ добавляется в качестве потомка обрабатываемого элемента. Если элемент является числом, то функция ищет среди «предков» элемент, содержащий всего одного «потомка».

Вывод графического представления дерева происходит в специально созданном для этого окне. Элемент, хранящийся в экземпляре класса Tree, выводится на экран в квадрате, цвет которого определяется случайным образом, с черной окантовкой. Элемент соединятся с «предками» и «потомками» черными линиями.

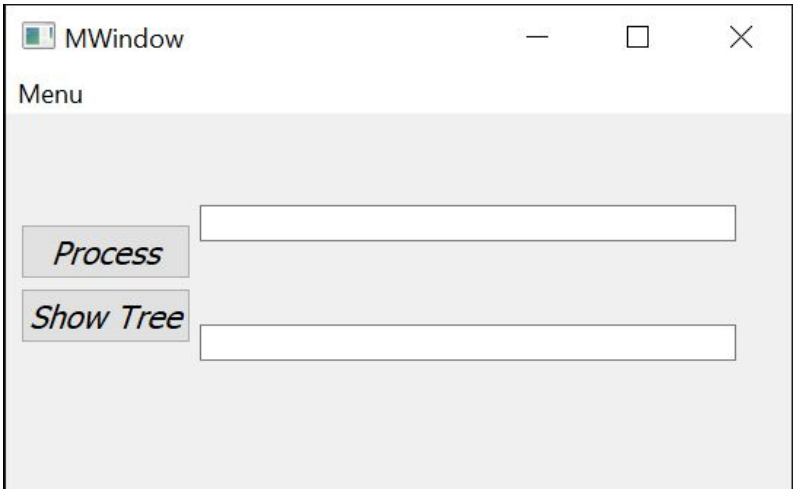
Класс Help содержит интерфейс, представляющий виджет с инструкциями к программе, и поле isOpen, в котором храниться булево значение, показывающее открыто окно Help или нет.

Спецификация программы

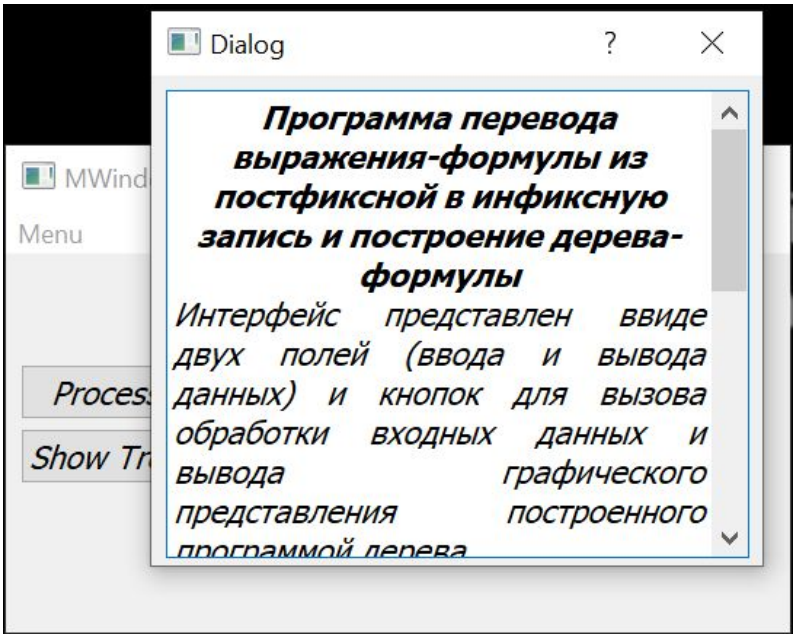
Программа предназначена для перевода записи выражения из постфиксной в инфиксную форму. Программа написана на языке C++. Входными данными является выражение в постфиксной форме. Выходными данными является вывод выражения в инфиксной форме, и графическое представление бинарного дерева, построенного по заданному выражению, выведенное в отдельное окно.

Тестирование программы

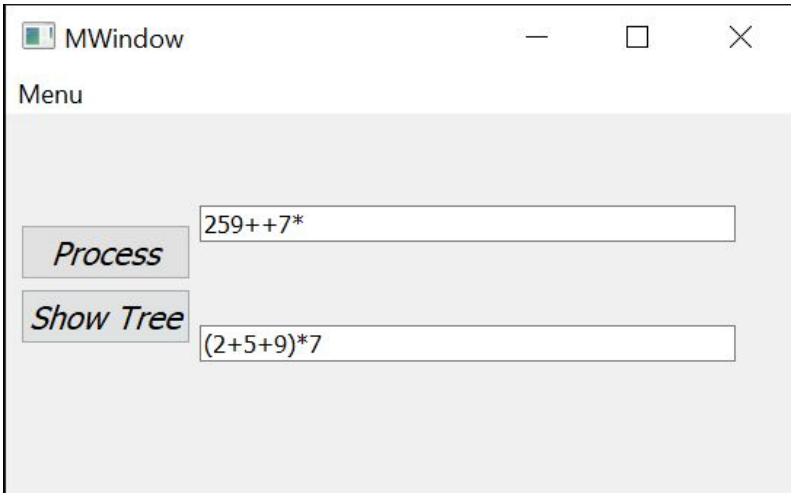
Интерфейс:

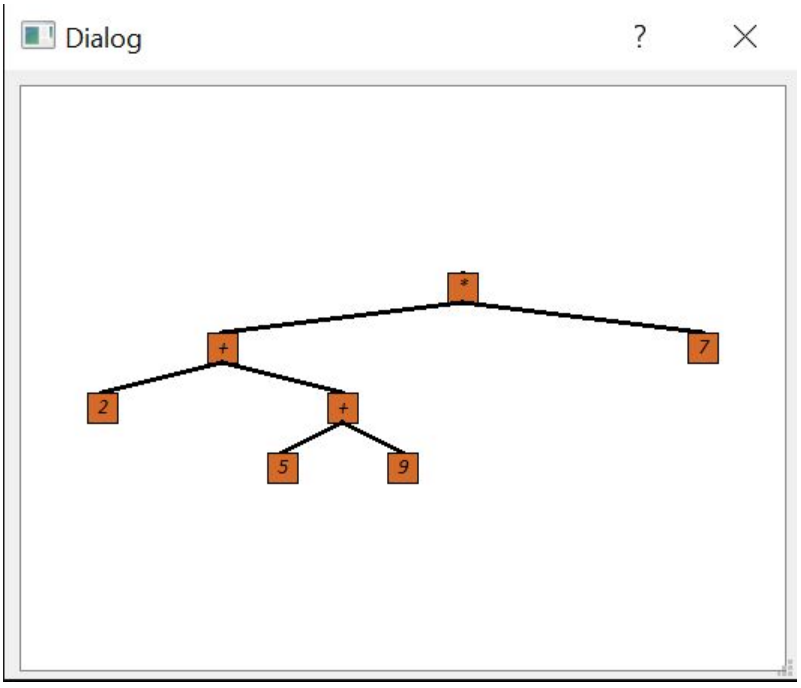


Help:



Результат работы:





Оценка эффективности алгоритма.

Алгоритм создания бинарного дерева по строке является итеративным, каждый элемент строки обрабатывается один раз, а значит сложность алгоритма можно оценить как O(n).

Алгоритм вывода дерева в постфиксной форме является рекурсивным, каждый узел дерева обрабатывается один раз, следовательно, сложность алгоритма также O(n).

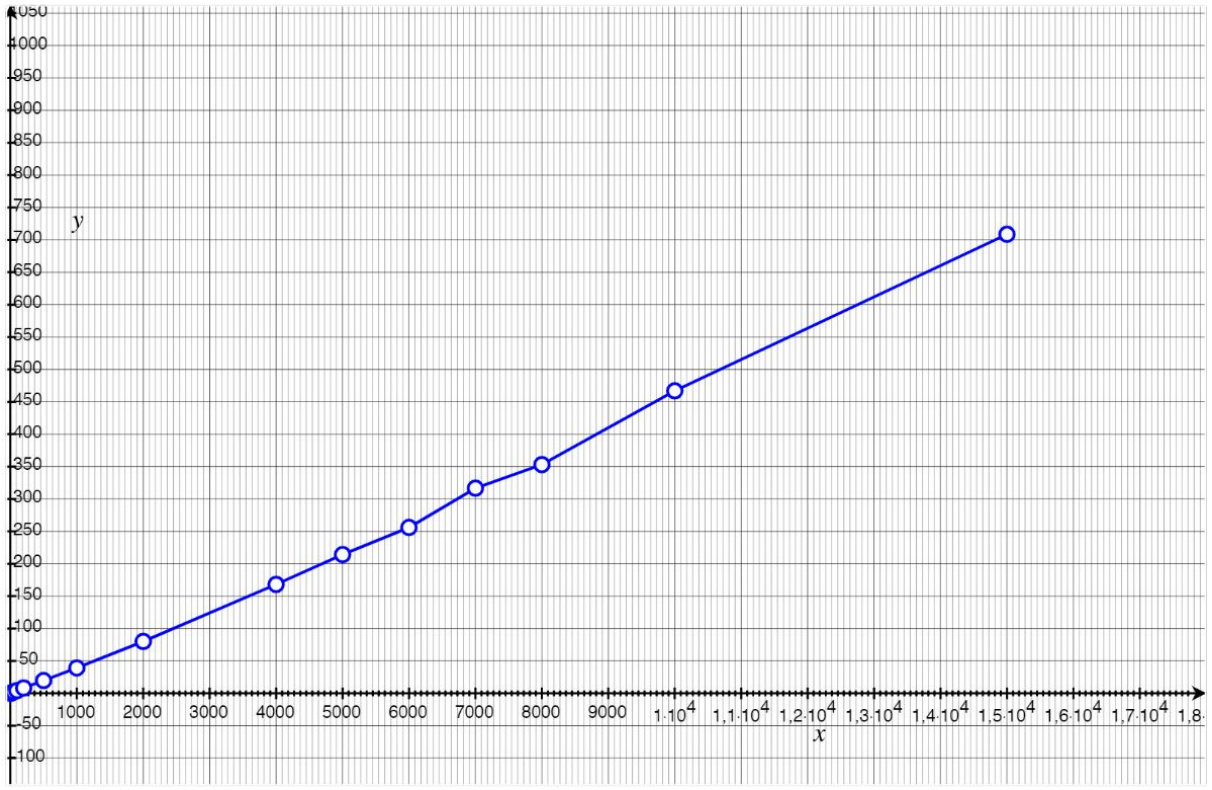


График 1 - Зависимость количества элементов к времени построения дерева

Вывод

В ходе выполнения работы было написано приложение на языке C++ с использованием фреймворка Qt, изучена работа с бинарным деревом на

примере программы переводящей выражение из постфиксной формы записи в обычную (инфиксную). Экспериментально определили сложность используемого алгоритма(O(n)).

ПРИЛОЖЕНИЕ А

Исходный код программы.

main.cpp

```
#include "mwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
  QApplication a(argc, argv);
  MWindow w;
  w.show();

  return a.exec();
}
```

mwindow.cpp

```
#include "mwindow.h"
#include "ui_mwindow.h"

MWindow::MWindow(QWidget *parent) :
  QMainWindow(parent),
  ui(new Ui::MWindow)
{
  ui->setupUi(this);
  help = new Help;
}

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

Tree* creat_tree(Tree *head, std::string str)
{
  head = new Tree;
  head->get_val(str[str.length()-1]);
  Tree* node = head;
  int i = str.length()-2;
  while(i>=0){
     node->create_node(str[i]);
     if(str[i]==42 || str[i]==43 || str[i]==45 || str[i]==47){
        if(node->ret_l())
           node = node->ret_l();
        else
           node = node->ret_r();
     }
     if(str[i]>=48 && str[i]<=57){
        while((node->ret_r() && node->ret_l()) && node->ret_parent()!=nullptr){
           node = node->ret_parent();
        }
     }
```

```
        i--;
    }
    return head;
}

void MWindow::on_process_clicked()
{
    std::stringstream outp;
//    Tree* head;
    head = creat_tree(head, ui->input->text().toStdString());
    std::string str = head->print_node();
    ui->output->setText(QString::fromStdString(str));
}

void MWindow::on_show_t_clicked()
{
    Dialog* dialog = new Dialog;
    int maxdeep = head->deepmax();
    head->displaytree(dialog, 1, maxdeep, 0, 0, 0, 0);

    dialog->exec();
}

void MWindow::on_actionHelp_triggered()
{
    if(!help->isOpen){
        help->show();
    }
}
```

mwindow.h

```
#ifndef MWINDOW_H
#define MWINDOW_H

#include <QMainWindow>
#include <string>
#include "tree.h"
#include "dialog.h"
#include "help.h"

namespace Ui {
class MWindow;
}

class MWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MWindow(QWidget *parent = 0);
    ~MWindow();

private slots:
    void on_process_clicked();
```

```
    void on_show_t_clicked();

    void on_actionHelp_triggered();

private:
    Ui::MWindow *ui;
    Tree* head;
    Help* help;
};

#endif // MWINDOW_H
```

tree.cpp

```
#include "tree.h"
#include <QGraphicsScene>

Tree::Tree()
{

}


char Tree::ret_val(){
    return val;
}
Tree* Tree::ret_l(){
    return left;
}
Tree* Tree::ret_r(){
    return right;
}
Tree* Tree::ret_parent(){
    return parent;
}
void Tree::get_val(char a){
    val = a;
}

void Tree::get_l(Tree* a){
    left = a;
}

void Tree::get_r(Tree* a){
    right = a;
}

void Tree::get_parent(Tree* a){
    parent = a;
}

void Tree::create_node(char val)
{
```

```
    if(!this->right){
        this->right = new Tree;
        this->right->val = val;
        this->right->parent = this;
    }
    else{
        this->left = new Tree;
        this->left->val = val;
        this->left->parent = this;
    }
}


std::string Tree::print_node(){

    std::stringstream ss;

    int flag = 0;
    if(parent!=nullptr && (val==43 || val==45) && (parent->val==42 || parent->val==47)){
        flag = 1;
        ss << "(";
    }

    if(left){
        ss << left->print_node();
    }

    std::string s;
    s.insert(0, 1, val);
    ss << s;

    if(right){
        ss << right->print_node();
    }

    if(flag)
        ss << ")";
    return ss.str();
}

int Tree::deepmax()
{
    int LHeight;
    if(left){
        LHeight = left->deepmax();
    } else LHeight = 0;

    int RHeight;
    if(right){
        RHeight = right->deepmax();
    } else RHeight = 0;

    return std::max(LHeight+1, RHeight+1);
```

```
}

void Tree::displaytree(Dialog *dialog, int deep, int maxdeep, int prevx, int prevy, int posx,
int posy)
{
  dialog->displaynode(val, deep, maxdeep, prevx, prevy, posx, posy);
  if(left)
     left->displaytree(dialog, deep + 1, maxdeep, posx, posy, posx -
(dialog->elemwidth*std::pow(2, maxdeep-deep)), posy + 2*dialog->elemheight);
  if(right)
     right->displaytree(dialog, deep + 1, maxdeep, posx, posy, posx +
(dialog->elemwidth*std::pow(2, maxdeep-deep)), posy + 2*dialog->elemheight);
}
```

tree.h

```
#ifndef TREE_H
#define TREE_H

#include <string>
#include <sstream>
#include <iostream>
#include <QGraphicsTextItem>
#include <cmath>

#include "dialog.h"

class Tree
{
public:
  Tree();
  void create_node(char val);

  char ret_val();
  Tree* ret_l();
  Tree* ret_r();
  Tree* ret_parent();

  void get_val(char a);
  void get_l(Tree* a);
  void get_r(Tree* a);
  void get_parent(Tree* a);
  int deepmax();
  std::string print_node();

  void displaytree(Dialog *dialog, int deep, int maxdeep, int prevx, int prevy, int posx, int
posy);

private:
  char val;
  Tree* parent = nullptr;
  Tree* left = nullptr;
  Tree* right = nullptr;
```

```
};


#endif // TREE_H
```

help.cpp

```
#include "help.h"
#include "ui_help.h"

Help::Help(QWidget *parent) :
  QDialog(parent),
  ui(new Ui::Help)
{
  ui->setupUi(this);
}

Help::~Help()
{
  delete ui;
}
```

help.h

```
#ifndef HELP_H
#define HELP_H

#include <QDialog>

namespace Ui {
class Help;
}

class Help : public QDialog
{
  Q_OBJECT

public:
  bool isOpen = 0;
  explicit Help(QWidget *parent = 0);
  ~Help();

private:
  Ui::Help *ui;
};

#endif // HELP_H
```

dialog.cpp

```
#include "dialog.h"
#include "ui_dialog.h"

Dialog::Dialog(QWidget *parent) :
  QDialog(parent),
  ui(new Ui::Dialog)
```

```
{
  QGraphicsScene *scene = new QGraphicsScene;

  ui->setupUi(this);

  ui->graphicsView->setScene(scene);

  mycolor = QColor::fromRgb(rand()%255,rand()%255,rand()%255);

  myfont.setItalic(1);
  myfont.setFamily("Calibri");
  myfont.setPixelSize(14);

  mybrush = QBrush(mycolor);
}


void Dialog::displaynode(char val, int deep, int maxdeep, int prevx, int prevy, int posx, int
posy)
{
  std::string s;
  s.insert(0, 1, val);
  QString line = QString::fromStdString(s);
  mytext = new QGraphicsTextItem(line);
  mytext->setFont(myfont);
  mytext->setPos(posx - line.length()*myfont.pixelSize()/2, posy-elemheight/2 -
myfont.pixelSize()/4);

ui->graphicsView->scene()->addLine(posx,posy-elemheight/2,prevx,prevy+elemheight/2,QP
en(Qt::black,3));

ui->graphicsView->scene()->addRect(posx-elemwidth/2,posy-elemheight/2,elemwidth,elemh
eight,mypen,mybrush);
  ui->graphicsView->scene()->addItem(mytext);
}


Dialog::~Dialog()
{
  delete ui;
}
```

dialog.h

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>
#include <QGraphicsTextItem>
#include <QString>
#include <QPen>
#include <string>
```

```
namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
  Q_OBJECT

public:
  explicit Dialog(QWidget *parent = 0);
  ~Dialog();
  void displaynode(char val, int deep, int maxdeep, int prevx, int prevy, int posx, int posy);
  int elemwidth = 20;
  int elemheight = 20;

private:
  Ui::Dialog *ui;


  QColor mycolor;
  QPen mypen = QPen(Qt::black,1);
  QFont myfont;
  QBrush mybrush;

  QGraphicsTextItem* mytext;


};

#endif // DIALOG_H
```