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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: Иерархические списки

Студентка гр. 8381 _____ Ивлева О.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивной обработки списков, изучить особенности реализации иерархического списка на языке программирования C++. Разработать программу, использующую иерархические списки и их рекурсивную обработку, удаляющую все вхождения заданного символа.

Задание.

Удалить из иерархического списка все вхождения заданного элемента (атома) x;

Основные теоретические положения.

Иерархические списки состоят из элементов различных уровней, при этом элементы нижних уровней подчинены элементам верхних уровней. Существует два вида иерархии списков: иерархия групп и элементов и иерархия элементов. Вид устанавливается конфигурацией.

В списке с иерархией групп и элементов содержатся два вида элементов – группы и собственно элементы. Группа обозначает узел, в который входят другие (подчиненные) группы и элементы, а элемент является конкретным объектом.

Для списков с иерархией элементов любой из элементов может быть как узлом, так и отдельным объектом. Примером может служить список подразделений. Каждое подразделение может содержать в своем составе другие подразделения, но набор свойств у всех подразделений будет одинаков.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки QtCreator с использованием фреймворка Qt.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Была добавлена кнопка, при

нажатии на которую, происходит удаление выбранного символа, добавлены Text_Edit для ввода строки и символа и добавлены Line_Edit для вывода текста.

Для реализации иерархического списка была реализована структура hlist элемента списка, которая состоит из указателей на следующий элемент списка, также из флага, который определяет, что будет храниться в элементе, символ или указатель на следующую иерархию, для этого было реализовано объединение union.

Функция r_make(), которая принимает на вход строку, индекс в данный момент и ошибку, создает иерархический список. Для этого сначала обнуляются указатели на голову списка, предыдущий и текущий элементы. Идет прохождение по каждому символу строки и запись в элемент списка, если встретился символ ‘(‘, то создается дочерний элемент и переход к следующему элементу будет происходить уже из него. Когда встретится символ ‘)’, то функция вернет указатель на начало списка, и данный адрес будет записан в указатель на дочерний элемент.

Функция del_x(), которая принимает на вход указатель на текущий элемент, символ, который следует удалить, строку для вывода и флаг на вывод подробной информации, удаляет из иерархического списка все вхождения заданного элемента (атома) x. Если необходимый символ встречается, то программа изменяет указатель на следующий элемент предыдущего на следующий элемент текущего. Если удаляемый символ является головой списка, то меняется указатель на голову списка, ей становится следующий элемент. Если является последним элементом списка – указатель на следующий элемент предыдущего становится nullptr.

Функция print_list(), которая принимает указатель на текущий элемент и ссылку на строку, записывает получаемый список в строку, для последующего его вывода.

Далее происходит вывод строки в графический интерфейс с помощью Line_Edit.

Оценка эффективности алгоритма.

Алгоритм, реализованный в программе, имеет линейную зависимость от длины строки, то есть сложность оценивается как O(n). Функции r_make() и del_x() запускаются не более 1 раза для каждого символа строки. Из-за выделения памяти для каждого элемента списка, скорость выполнения программы не такая высокая, как могла бы быть при реализации через массив.

Тестирование приложения.

Вид программы после запуска:

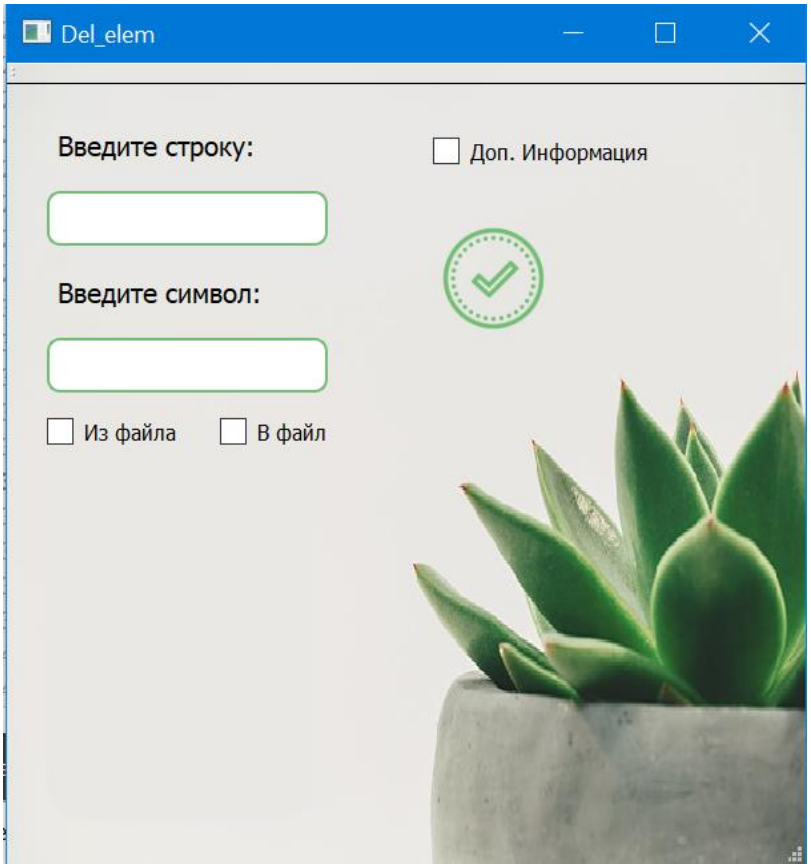

Таблица 1 – Тестирование программы

Входная строка	Удаляемый символ	Вывод
(asda(djsdh)(asa))	a	a Delete! s d a Delete! d j s d h a Delete! s a Delete! (sd(djsdh)(s))
(asd)	a	a Delete! s d (sd)
(asda(aaa)(asa))	a	a Delete! s d a Delete! a Delete! a Delete! a Delete! a Delete! s a Delete! (sd()(s))
(fdfgdf)	a	f d f g d f fdfgdf
(aaa(aa(aa(aa))))	a	a Delete! a Delete! a Delete! a Delete! a Delete! a Delete! a Delete! a Delete! a Delete! ((((()))))

Вывод.

В ходе выполнения лабораторной работы была изучена такая структура данных как иерархические списки, а также рекурсивные методы ее обработки. Была реализована программа на C++, использующая иерархические списки, которая удаляет выбранный пользователем символ.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#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.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "functions.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_Hello_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

Название файла: functions.h

```
#include <string>
#include <iostream>
using namespace std;

struct hlist {
    int flag;
    hlist* ptr_next;
    union {
        char symb;
```

```
        hlist *ptr_child;
    };
};

hlist* r_make(string &mas, int &start, int &err);
hlist* del_x(hlist *now, char &x, string &output,bool info);
void print_list(hlist *now, string &output);
```

Название файла: functions.cpp

```
#include "functions.h"

hlist* r_make(string &mas, int &start, int &err){
    hlist *head = nullptr;
    hlist *last = nullptr;
    hlist *now = nullptr;
    if (err)
        return head;
    while (mas[start]){
        if (mas[start] == ' ') {
            start++;
            continue;
        }
        if (mas[start] != '(' && mas[start] != ')') {
            if(head == nullptr){
                head = new hlist;
                head->flag = 1;
                head->symb = mas[start++];
                head->ptr_next = nullptr;
                last = head;
            }
            else
            {
                now = new hlist;
                last->ptr_next = now;
                now->flag = 1;
                now->symb = mas[start++];
                now->ptr_next = nullptr;
                last = now;
            }
        }
        else if (mas[start] == '('){
            now = new hlist;
            if (head == nullptr){
                head = now;
                last = now;
            }
            else {
                last->ptr_next = now;
                last = now;
            }
            now->flag = 0;
            now->ptr_next = nullptr;
            start++;
            now->ptr_child = r_make(mas, start, err);
        }
        else if (mas[start] == ')'){
```

```
            start++;
            return head;
        }
    }
        return head;
}

hlist* del_x(hlist *now, char &x, string &output, bool info){
    int st = 1;
    hlist *head = now;
    hlist *prev = now;
    while (now) {
        if (now->flag == 1) {
            if (info){
                output += now->symb;
                if (now->symb != x){
                    output += "\t";
                }
                else {
                    output += " ";
                }
            }
            if (now->symb == x) {
                if (info){
                    output += "Delete!";
                    output += "\t";
                }
                if (st == 1) {
                    if (head->ptr_next == nullptr)
                        return nullptr;
                    head = head->ptr_next;
                    prev = head->ptr_next;
                }
                else {
                    prev->ptr_next = now->ptr_next;
                }
            }
            else {
                prev = now;
            }
            now = now->ptr_next;
            st = 0;
        }
        else{
            now->ptr_child = del_x(now->ptr_child, x, output, info);
            prev = now;
            now = now->ptr_next;
        }
    }
    return head;
}

void print_list(hlist *now, string &output){
    if(now){
        if (now->flag == 1) {
            output += now->symb;
```

```
            }
            else {
                output += "(";
                print_list(now->ptr_child, output);
                output += ")";
            }
            print_list(now->ptr_next, output);
        }
    }
```

Название файла: mainwindow.cpp

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

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->output->setWordWrap(true);
    QPixmap bkgnd("D:\\prog\\Styles\\plant-2004483_640.jpg");
    bkgnd = bkgnd.scaled(this->size(), Qt::IgnoreAspectRatio);
    QPalette palette;
    palette.setBrush(QPalette::Background, bkgnd);
    this->setPalette(palette);
}

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

void MainWindow::on_Hello_clicked()
{
    string input = qPrintable(ui->input->text());
    string inputChar = qPrintable(ui->inputChar->text());
    string output = "";
    int start = 1;
    int err = 0;
    hlist *head = r_make(input, start, err);
    hlist   *del_head   =   del_x(head,   inputChar[0],   output,
ui->checkBox->isChecked());
    output += "\n";
    output += "\n";
    print_list(del_head, output);
    ui->output->setText(QString::fromStdString(output));

}
```

Название файла: 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>600</width>
          <height>600</height>
         </rect>
        </property>
        <property name="focusPolicy">
         <enum>Qt::NoFocus</enum>
        </property>
        <property name="windowTitle">
         <string>Del_elem</string>
        </property>
        <property name="autoFillBackground">
         <bool>false</bool>
        </property>
        <property name="styleSheet">
         <string notr="true"/>
        </property>
        <widget class="QWidget" name="centralWidget">
         <widget class="QPushButton" name="Hello">
          <property name="geometry">
           <rect>
            <x>320</x>
            <y>100</y>
            <width>91</width>
            <height>91</height>
           </rect>
          </property>
          <property name="cursor">
           <cursorShape>PointingHandCursor</cursorShape>
          </property>
          <property name="styleSheet">
           <string                                notr="true">border-image:
url(:/C:/Users/Олеся/Downloads/icons8-ok-100_1.png);</string>
          </property>
          <property name="text">
           <string/>
          </property>
          <property name="autoRepeat">
           <bool>false</bool>
          </property>
         </widget>
         <widget class="QLabel" name="output">
          <property name="geometry">
           <rect>
            <x>30</x>
            <y>250</y>
            <width>201</width>
            <height>301</height>
           </rect>
          </property>
          <property name="styleSheet">
           <string notr="true">font: 6pt &amp;quot;MS Shell Dlg 2&amp;quot;;
      border-radius: 30%;
```

```
background-color: #e9e8e4;
</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QLineEdit" name="input">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>80</y>
      <width>211</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-radius: 10%;
border: 2px solid #72be74;</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="inputChar">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>190</y>
      <width>211</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-radius: 10%;
border: 2px solid #72be74;</string>
    </property>
   </widget>
   <widget class="QCheckBox" name="checkBox">
    <property name="geometry">
     <rect>
      <x>320</x>
      <y>40</y>
      <width>191</width>
      <height>21</height>
     </rect>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="text">
     <string>Доп. Информация</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>30</y>
```

```
      <width>181</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">
font: 10pt &amp;quot;MS Shell Dlg 2&amp;quot;;
border-radius: 10%;
background-color: #e9e8e4;
</string>
    </property>
    <property name="text">
     <string> Введите строку:</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_2">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>140</y>
      <width>181</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">
font: 10pt &amp;quot;MS Shell Dlg 2&amp;quot;;
border-radius: 10%;
background-color: #e9e8e4;</string>
    </property>
    <property name="text">
     <string> Введите символ:</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>600</width>
     <height>17</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
```

```
 <resources/>
 <connections/>
</ui>
```