ММИНОБРНАУКИ 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: Рекурсивная обработка иерархических списков.

Студент гр. 9384 _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

С помощью рекурсивных функций создать иерархический список и вывести глубину списка.

Задание.

ВАРИАНТ 13.

Вычислить глубину (число уровней вложения) иерархического списка как максимальное число одновременных открытых левых скобок в сокращенной скобочной записи списка: принять, что глубина пустого списка и глубина атомарного S-выражения равны нулю; например глубина списка (a (b()c) d) равна двум;

Выполнение работы.

Для описания иерархического списка была создана структура, хранящая в себе значение (Atom), а также указатель на “голову” и “хвост”:

```
struct s_expr
{
    bool isAtom;
    s_expr* ptr_next;
    union
    {
        char atom;
        s_expr* ptr_child;
    } node;
};
```

Функция CreateList создает иерархический список из входящей строки (string). И возвращает указатель на “голову”.

Для подсчета глубины иерархического списка, реализована рекурсивная функция Depth.

Проверку на отсутствие скобок, осуществляет функция CheckBrakets:

```
bool CheckBrackets(const std::string str)
{
   size_t open = std::count(str.begin(), str.end(), '(');
   size_t closed = std::count(str.begin(), str.end(), ')');
   if (open > closed)
   {
      std::cout << "Missing closed bracket" << std::endl;
      return false;
   }
   else if (closed > open)
   {
      std::cout << "Missing open bracket" << std::endl;
      return false;
   }
   return true;
}
```

Ввод и вывод осуществлен по выбору пользователя(1- из файла, 2 – из терминала).

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(a(b)(b)(b(cd)))	Depth of ((a(b)(b)(b(cd)))) is 3	Глубина выражения - 3
2.	(asd))))))	Missing open bracket	Присутствует лишняя скобка
3.	(a(b(c(d(e(p))))))	Depth of ((a(b(c(d(e(p))))))) is 6	Глубина выражения - 6

Выводы.

В ходе выполнения лабораторный работы были изучены иерархические списки, а также была разработана программа, использующая обход по иерархическим спискам, для подсчета глубины выражения

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2_13.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>

struct s_expr
{
    bool isAtom;
    s_expr* ptr_next;
    union
    {
        char atom;
        s_expr* ptr_child;
    } node;
};

s_expr* CreateList(std::string& str, int& pos)
{
    s_expr *hd = nullptr;
    s_expr *tl = nullptr;
    s_expr *st = nullptr;
    do
    {
        if (str[pos] == '(')
        {
            st = new s_expr;
            if (!hd)
```

```
    {
        hd = st;
        tl = st;
    }
    else
    {
        tl->ptr_next = st;
        tl = st;
    }
    st->isAtom = false;
    st->ptr_next = nullptr;
    pos++;
    st->node.ptr_child = CreateList(str, pos);
}
else if (str[pos] == ')')
{
    pos++;
    return hd;
}
else
{
    if (!hd)
    {
        hd = new s_expr;
        hd->isAtom = true;
        hd->ptr_next = nullptr;
        hd->node.atom = str[pos];
        tl = hd;
        pos++;
    }
```

```
            else
            {
                st = new s_expr;
                tl->ptr_next = st;
                st->isAtom = true;
                st->ptr_next = nullptr;
                st->node.atom = str[pos];
                tl = st;
                pos++;
            }
        }
    } while (str[pos]);
    return hd;
}

void Depth(s_expr* pos, int count, int& answer) {
    if (pos) {
        if (!pos->isAtom && pos->node.ptr_child) {
            count++;
            Depth(pos->node.ptr_child, count,answer);
        }
        if (pos->isAtom) {
            if (answer < count) answer = count;
        }
        if (!pos->isAtom && pos->node.ptr_child) {
            Depth(pos->ptr_next, --count,answer);
        }
        else
            Depth(pos->ptr_next, count, answer);
        return;
```

```
    }
    return;
}

bool CheckBrackets(const std::string str)
{
    size_t open = std::count(str.begin(), str.end(), '(');
    size_t closed = std::count(str.begin(), str.end(), ')');
    if (open > closed)
    {
        std::cout << "Missing closed bracket" << std::endl;
        return false;
    }
    else if (closed > open)
    {
        std::cout << "Missing open bracket" << std::endl;
        return false;
    }
    return true;
}

char read_type()
{
 std::cout << "Введите 1 для считывания из файла\n\
 Введите 2 для считывания из консоли" << std::endl;
 char key = getchar();
 return key;
}

int main()
```

```
{
    switch (read_type())
    {
    case '1':
        {
            //Считывание с файла

            std::ifstream in_file("input.txt");
            std::ofstream out_file("output.txt");
            std::string line;
            int i = 0;
            int line_number = 1;
            while (std::getline(in_file, line))
            {

                int count = 0, answer = 0;
                line.pop_back();
                out_file << line_number++ << ": ";
                if (!CheckBrackets(line)) out_file << \
                "Brackets error, check terminal for more information\n" << std::endl;
                else
                {
                    out_file << "Depth of ( " << line << " )";
                s_expr* head = CreateList(line, count);
                Depth(head, 0, answer);
                out_file << " is  " << answer << " \n" << std::endl;
                }
            }
            break;
        }
```

```
    case '2':
        {
            // Считывание с консоли

            std::string str;
            int count = 0, answer = 0;
            std::cout << "Введите значение" << std::endl;
            std::cin >> str;
            CheckBrackets(str);
            std::cout << "Depth of ( " << str << " )";
            s_expr* head = CreateList(str, count);
            Depth(head, 0, answer);
            std::cout << " is  " << answer << " " << std::endl;
            break;
        }
    default:
        std::cout<<"Числа нет в списке команд" << std::endl;
        break;
    }
}
```