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

ОТЧЕТ

по практической работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Иерархические списки

Студент гр. 9303 _____ Скворчевский Б.С.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с понятием иерархического списка, изучить его особенности и реализовать программу, решающую поставленную задачу с помощью иерархического списка.

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.

Постановка задачи.

21. Пусть арифметическое* выражение представлено иерархическим списком. В выражение входят константы и переменные, которые являются атомами списка. Операции представляются в постфиксной форме (<аргументы> <операция>). Аргументов может быть 1, 2 и более. Например (в префиксной форме): (+ a (* b (- c))) или (OR a (AND b (NOT c))). В задании даётся один из следующих вариантов требуемого действия с выражением: *проверка синтаксической корректности*, *упрощение* (преобразование), ***вычисление***.
Пример *упрощения*: (+ 0 (* 1 (+ a b))) преобразуется в (+ a b).
В задаче *вычисления* на входе дополнительно задаётся список значений переменных ((x_1 c_1) (x_2 c_2) … (x_k c_k)), где x_i – переменная, а c_i – её значение (константа).
В индивидуальном задании указывается: тип выражения (возможно дополнительно - состав операций), вариант действия и форма записи. Всего 9 заданий.

* - здесь примем такую терминологию: в *арифметическое* выражение входят операции +, -, *, /, а в *алгебраическое* – +, -, * и дополнительно некоторые функции.
Индивидуальное задание: арифметическое, вычисление, постфиксная форма.

Выполнение работы.

Программа принимает строку из файла и записывает её в массив символов `a`. Далее она выводит его на экран. После программа заменяет все буквы переменных в введённой строке на соответствующие им значения. После этого создаётся стек `s`. Далее программа с помощью цикла проходит по всем символам массива `a` и проверят каждый символ с помощью оператора `if`. Если очередной символ равен знаку операции (+ или – или * или /), то в список записывается число равное результату этой операции. Если же очередной символ равен цифре (или отрицательному числу), то она записывается в список.

После того, как программа проходится по всем символам из массива, она выводит результат вычислений и очищает список.

Выводы.

Были изучены основные понятия иерархического списка и его особенности. Была реализована программа, решающая поставленную задачу с помощью иерархического списка, а именно вычисление арифметического выражения в постфиксной форме.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <cctype>
#include "list.h"

using namespace std;
using namespace list;

int main () {
   char a[100];
   ifstream fin("input.txt");
   fin >> noskipws;
   if (!fin ) {
      cout << "Не удаётся открыть файл на чтение!\n";
      return 1;
   }
   int n1 = 100;
   int n = 0;
   cout << "Вы ввели: ";
   char sline[50];
    char tmp[100];
   while (n < n1 && fin >> tmp[n]) {
        cout << tmp[n];
       n++;
    }

    cout << '\n';
   int len = n;
   n = 0;
   int k = 0;
   int f = 0;
   while(n < len && tmp[n] != '\n'){
        a[n] = tmp[n];
        f++;
        n++;
   }
   n++;
   while(n < len){
        sline[k] = tmp[n];
        k++;
        n++;
   }
    sline[k] = '\0';
   n = 0;
```

```
    while(n < (len - f)){
        if(isalpha(sline[n])){
            for(int j = 0; j < f; j++){
                if(a[j] == sline[n])
                    a[j] = sline[n+2];
            }
        }
        n++;
    }
     cout << '\n';

    cout << "Вычисление:" << endl;
    Stack s;
    n = f;
    for (int i = 0; i < n; i++){
       if (a[i] == '+')
          s.push(s.pop() + s.pop());
         if (a[i] == '-' && !isdigit(a[i+1]))
             s.push(s.pop() - s.pop());
         if (a[i] == '/')
             s.push(s.pop() / s.pop());
       if (a[i] == '*')
          s.push(s.pop() * s.pop());
       if (a[i-1] != '-' && (a[i] >= '0') && (a[i] <= '9'))
          s.push(0);
         if (a[i-1] == '-' && isdigit(a[i]))
             s.push('0' - a[i]);
       while (a[i-1] != '-' && (a[i] >= '0') && (a[i] <= '9')){
          s.push(10 * s.pop() + (a[i++] - '0'));
       }
    }

    cout << "Результат = " << s.pop() << endl;
    s.destroy();
    return 0;
}
```

Название файла: list.h

```
namespace list {

    typedef int base;
    class Stack {
    private:
        struct node;
        node *topOfStack;
    public:
        Stack (){
            topOfStack = nullptr;
        };
        base pop();
        void push (const base &x);
        void destroy ();
    };
}
```

Название файла: list.cpp

```
#include <iostream>
#include <cstdlib>
#include "list.h"

using namespace std;

namespace list{

    struct Stack::node {
        base *hd;
        node *tl;

        node() {
            hd = nullptr; tl = nullptr;
        }
    };
```

```
base Stack::pop() {
    if (topOfStack == nullptr) {
        cerr << "Error: pop (null) \n";
        exit(1);
    }
    else {
        node *oldTop = topOfStack;
        base r = *topOfStack->hd;
        topOfStack = topOfStack->tl;
        delete oldTop->hd;
        delete oldTop;
        return r;
    }
}

void Stack::push(const base &x) {
    node *p;
    p = topOfStack;
    topOfStack = new node;
    if (topOfStack != nullptr) {
        topOfStack->hd = new base;
        *topOfStack->hd = x;
        cout << "push -> " << x << endl;
        topOfStack->tl = p;
    }
    else {
        cerr << "Memory not enough\n"; exit(1);
    }
}

void Stack::destroy() {
    while (topOfStack != nullptr) {
        pop();
    }
```

```
        }
    }
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица 1 - Примеры тестовых случаев

№	Входные данные	Выходные данные	Комментарий
1	(((-2 0 /) -4 +) -7 -)	Вы ввели: (((-2 0 /) -4 +) -7 -) Вычисление: push -> -2 push -> 0 push -> 0 push -> 0 push -> -4 push -> -4 push -> -7 push -> -3 Результат = -3	Программа работает корректно
2	(2 6 /)	Вы ввели: (2 6 /) Вычисление: push -> 0 push -> 2 push -> 0 push -> 6 push -> 3 Результат = 3	Программа работает корректно
3	((2 6 /) 5 +)	Вы ввели: ((2 6 /) 5 +) Вычисление: push -> 0	Программа работает корректно

		push -> 2 push -> 0 push -> 6 push -> 3 push -> 0 push -> 5 push -> 8 Результат = 8	
4	(((2 6 /) 5 +) 7 -)	Вы ввели: (((2 6 /) 5 +) 7 -) Вычисление: push -> 0 push -> 2 push -> 0 push -> 6 push -> 3 push -> 0 push -> 5 push -> 8 push -> 0 push -> 7 push -> -1 Результат = -1	Программа работает корректно
5	(3 -6 /)	Вы ввели: (3 -6 /) Вычисление: push -> 0 push -> 3 push -> -6	Программа работает корректно

		push -> -2 Результат = -2	
6	((3 -6 /) 3 -)	Вы ввели: ((3 -6 /) 3 -) Вычисление: push -> 0 push -> 3 push -> -6 push -> -2 push -> 0 push -> 3 push -> 5 Результат = 5	Программа работает корректно
7	(((3 -6 /) 3 -) 5 *)	Вы ввели: (((3 -6 /) 3 -) 5 *) Вычисление: push -> 0 push -> 3 push -> -6 push -> -2 push -> 0 push -> 3 push -> 5 push -> 0 push -> 5 push -> 25 Результат = 25	Программа работает корректно

8	((((3 -6 /) 3 -) 5 *) 3 +)	Вы ввели: ((((3 -6 /) 3 -) 5 *) 3 +) Вычисление: push -> 0 push -> 3 push -> -6 push -> -2 push -> 0 push -> 3 push -> 5 push -> 0 push -> 5 push -> 25 push -> 0 push -> 3 push -> 28 Результат = 28	Программа работает корректно
9	(2 (a b -) +) ((a 2) (b 0))	Вы ввели: (2 (a b -) +) ((a 2) (b 0)) Вычисление: push -> 0 push -> 2 push -> 0 push -> 2 push -> 0 push -> 0 push -> -2 push -> 0	Программа работает корректно

		Результат = 0	
10	(a (c b -) +) ((a 2) (b 0) (c 1))	Вы ввели: (a (c b -) +) ((a 2) (b 0) (c 1)) Вычисление: push -> 0 push -> 2 push -> 0 push -> 1 push -> 0 push -> 0 push -> -1 push -> 1 Результат = 1	Программа работает корректно