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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсивная обработка иерархических систем

Студентка гр. 9303 _____ Булыно Д. А.

Преподаватель _____ Филатов А. Ю.

Санкт-Петербург

2020

Цель работы.

Формирование практических навыков программирования рекурсивных алгоритмов для работы с иерархическими системами на языке программирования C++ путём решения поставленной задачи.

Основные теоретические положения.

В практических приложениях возникает необходимость работы с более сложными, чем линейные списки, нелинейными конструкциями. Рассмотрим одну из них, называемую иерархическим списком элементов базового типа El или S-выражением.

Определим соответствующий тип данных S_expr (El) рекурсивно, используя определение линейного списка (типа L_list):

< S_expr (El) > ::= < Atomic (El) > | < L_list (S_expr (El)) >,

< Atomic (E) > ::= < El >.

< *L_list*(*El*) > ::= < *Null_list* > | < *Non_null_list*(*El*) >

< *Null_lis t*> ::= *Nil*

< *Non_null_list*(*El*) > ::= < *Pair*(*El*) >

< *Pair*(*El*) > ::= (< *Head_l*(*El*) > . < *Tail_l*(*El*) >)

< *Head_l*(*El*) > ::= < *El* >

< *Tail_l*(*El*) > ::= < *L_list*(*El*) >

Традиционно иерархические списки представляют или графически, или в виде скобочной записи. Переход от полной скобочной записи, соответствующей определению иерархического списка, к сокращенной производится путем отбрасывания конструкции Nill и удаления необходимое число раз пары скобок вместе с предшествующей открывающей скобке точкой.

Согласно приведенному определению иерархического списка, структура непустого иерархического списка — это элемент размеченного объединения множества атомов и множества пар «голова-хвост».

Задание.

Вариант 21

Пусть выражение (логическое, арифметическое, алгебраическое*) представлено иерархическим списком. В выражение входят константы и переменные, которые являются атомами списка. Операции представляются в префиксной форме ((<операция> <аргументы>)), либо 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.

арифметическое, вычисление, постфиксная форма

Выполнение работы.

Программа предназначена для арифметического вычисления выражения, заданного в постфиксальной форме. Для этой программы был реализован класс, с помощью методов которого можно наблюдать промежуточные значения в результате выполнения программы, временно запоминать данные. С помощью функции `int chekSymbol(char val)` программа проверяет символ на число или знак. В функции `void calc(stack* Stack, string str)` происходит обработка

полученных данных, а именно определение строки на число или знак, написание промежуточных значений, которые в итоге приведут к результату. В главной функции происходит ввод-вывод информации.

Код программы см. в приложении А.

Результаты работы программы см. в приложении Б.

Выводы.

В ходе проведения лабораторной работы были получены навыки программирования рекурсивных алгоритмов для работы с иерархическими системами на языке программирования C++. В результате проведения лабораторной работы была написана программа, которая выполняет определенные действия с выражением, представляющим из себя иерархический список.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#define PATERN "0123456789+-*/ "

using namespace std;

class  stack{
private:
    int count;
    int top;
    int *Arr;
    void getSize(){
        count+=50;
        int *tmp = new int[count];
        for (int i = 0; i < top; i++){
            tmp[i] = Arr[i];
        }
        Arr = tmp;
    }
public:
    stack(){
        count = 50;
        top = -1;
        getSize ();
    }

    void push( int val ){
        top++;
        if(top >= (count-1)) getSize();

        Arr[top] = val;
        cout << "PUSH:\t"<< val <<"\n";
    };

    bool isEmpty(){
        return top < 0;
    }

    int pop(){
        cout << "POP::\t"<< Arr[top] <<"\n";
        return Arr[top--];
    }

    void print(){
        for(int i = 0; i < top; i++){
            cout << "["<<i<<"][  " << Arr[i] << "  ]\n";
        }
```

```
    }

    ~stack(){
        delete [] Arr;
    }
};


int chekSymbol(char val){
    for(int i = 0; i < 14 ; i++){
        if(val == PATERN[i]) return i;
    }
    return -1;
}


void calc(stack* Stack, string str){
    int key, a, b, c;
    for(int i = 0; i < str.length() && str[i] != '\0'; i++){
        key = chekSymbol(str[i]);
        int tmp = 0;
        if(key > 9 && key < 14){
            b = Stack->pop();
            a = Stack->pop();
            switch(key){
                case 10:
                    c = a+b;
                    cout << a << " + " << b <<" = " << c << endl;

                   break;
                case 11:
                    c = a-b;
                    cout << a << " - " << b<<" = " << c << endl;

                    break;
                case 12:
                    c = a*b;
                    cout << a << " * " << b <<" = " << c << endl;
                    break;
                case 13:
                    c = a/b;
                    cout << a << " / " << b <<" = " << c << endl;
                    break;
            }
            Stack->push(c);
        }
        else if(key >= 0 && key <= 9){
            int tmp = 0;
            while(key >= 0 && key <= 9){
                tmp = tmp*10+key;
                key = chekSymbol(str[++i]);
```

```
            }
            Stack->push(tmp);
            i--;
        }
        else{
            cout << endl;
        }
    }
}

int main(){
    string str;
    cout << "Введите выражение в постфиксной форме: ";
    getline(cin, str);
    stack *Stack = new stack();
    calc(Stack, str);
    cout << Stack->pop() << endl;
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Таблица 1. Результаты работы программы

№	Входные данные	Результат
1.	Введите выражение в постфиксной форме: 5 6 -	PUSH: 5 PUSH: 6 POP:: 6 POP:: 5 5 - 6 = -1 PUSH: -1 POP:: -1 -1
2.	Введите выражение в постфиксной форме: 34 23 + 23 4 * /	PUSH: 34 PUSH: 23 POP:: 23 POP:: 34 34 + 23 = 57 PUSH: 57 PUSH: 23 PUSH: 4 POP:: 4 POP:: 23 23 * 4 = 92 PUSH: 92 POP:: 92 POP:: 57 57 / 92 = 0 PUSH: 0 POP:: 0 0
3.	Введите выражение в постфиксной форме: 45 65 34 - + 21 435 / *	PUSH: 45 PUSH: 65 PUSH: 34 POP:: 34

		POP:: 65 65 - 34 = 31 PUSH: 31 POP:: 31 POP:: 45 45 + 31 = 76 PUSH: 76 PUSH: 21 PUSH: 435 POP:: 435 POP:: 21 21 / 435 = 0 PUSH: 0 POP:: 0 POP:: 76 76 * 0 = 0 PUSH: 0 POP:: 0 0