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

# ОТЧЕТ

**по лабораторной работе №1**
**по дисциплине «Алгоритмы и структуры данных»**
**Тема: Списочные структуры**

Студентка гр. 1373 _____ Куприянова А. М.

Преподаватель _____ Бондаренко Б. Е.

Санкт-Петербург
2022

**Цель работы:**

Освоить реализацию и принцип работы с основными видами списочных структур, на их основе построить алгоритм сортировочной станции.

**Задание:**

Реализовать следующие структуры: *односвязный список*, *динамический массив* и *стек*. Стек можно реализовать как на базе списка, так и отдельно. Использовать стек для реализации алгоритма сортировочной станции Разрешённые символы в исходном выражении: +, -, *, /, ^, sin, cos, (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Для упрощения разбиения входной строки на токены разрешается отделять каждый символ пробелом.

**Выполнение работы:**

1. Динамический массив

Для его реализации был создан класс Arr, содержащий поля массива int *arr и его объема int capacity.

Тип класса public содержит следующие функции:

- Void FillArray – заполнение массива элементами; массив заполняется при выполнении проверки на вид элементов массива: все они должны быть числами
- Void FillRandArray – заполнение массива рандомными элементами: массив заполняется элементами от 1 до 20;
- Void PrintArray – вывод элементов массива;
- Void DeleteElement (int i) – удаление элемента по индексу: сначала создаем временный массив *tempArr, потом сравниваем его с исходным, когда индекс элемента исходного массива i совпадет с индексом временного k, элемент удаляется. После временный массив перезаписываем как исходный;

- Void AddElement (int i, int val) – добавление элемента в массив: процесс похож на удаление элемента, но, когда доходим до нужного элемента, меняем его значение на val.

2. Односвязный список

Вначале создается структура узла Node. В ней два поля: data - данные узла и указатель на следующий узел *nextNode типа Node.

Далее создается класс списка List. В private объявляем указатель startNode, который указывает на первый узел.

Описание функций:

- addNode -- создается функция добавления узла в конец списка: при помощи указателя node передаем в data само значение d, а в nextNode NULL. Далее рассматриваются два случая: если в списке еще нет узлов и если в списке уже есть узлы(т.е. он не пустой). Если в списке нет узлов, то просто по указателю startNode записываем node. Если список не пустой, то создаем указатель thisNode на текущий узел (в начале он указывает на начальный узел) и далее при его помощи идем по узлам, пока узел не станет последним. Когда доходим до последнего узла, то в последний nextNode(вместо NULL) записываем новый добавленый узел, уже со своими новыми data и nextNode (указывает на NULL);
- printList() -- функция для вывода списка на экран: в ней выводим значение data каждого узла;
- getListSize() – функция для подсчета размера списка: создаем счетчик listSize: пока узел не последний увеличиваем счетчик и проходим к следующему узлу;
- DeleteNode (int i) – функция для удаления узла по индексу: в функцию передаем индекс элемента i, который хотим удалить; если индекс == 0, то переходим на следующий узел, а если не равен – просто делаем так,

чтобы предыдущий узел указывал не на удаляемый (он идет после предыдущего), а на следующий после него.

3. Стек + алгоритм сортировочной станции

Создается класс стека Stack, в public находятся следующие члены класса: указатель для арифметического выражения string* arr_string, вершина стека int top и его объем int capacity. Эти переменные являются основными в реализации стека.

Память для стека выделяем, как и для массива, при помощи new.

Описание функций:

- Void Stack::push – функция для добавления элемента на вершину стека;
- String Stack::pop – функция для удаления элемента из стека;
- String Stack:: topElement – функция для вывода вершины стека;
- Int Stack::stackSize – функция вывода размера стека;
- Bool Stack::isEmpty – булевая функция, проверяющая, пуст ли стек;
- Bool Stack::isFull – булевая функция, проверяющая, полон ли стек;
- Void Stack::printStack – функция вывода элементов стека;
- Bool check_number -- функция проверяет является ли данный токен числом(тип int). Далее циклом проходим по нему и смотрим является ли конкретный индекс токена цифровым значением. Функция возвращает true, если токен число, false в противном случае;
- Int operatorRank – функция определения приоритета оператора;
- Void SortString – функция, выполняющая сортировку входной строки строго в соответствии с ниже приложенным алгоритмом.

Алгоритм сортировочной станции:

- Пока не все токены обработаны:
    - Прочитать токен.

- Если токен — число, то добавить его в очередь вывода.
- Если токен — функция, то поместить его в стек.
- Если токен — разделитель аргументов функции (например запятая):

- Пока токен на вершине стека не открывающая скобка:
  - Переложить оператор из стека в выходную очередь.
  - Если стек закончился до того, как был встречен токен открывающая скобка, то в выражении пропущен разделитель аргументов функции (запятая), либо пропущена открывающая скобка.

- Если токен — оператор op1, то:

- Пока присутствует на вершине стека токен оператор op2, чей приоритет выше или равен приоритету op1, и при равенстве приоритетов op1 является левоассоциативным:
  - Переложить op2 из стека в выходную очередь;
- Положить op1 в стек.

- Если токен — открывающая скобка, то положить его в стек.
- Если токен — закрывающая скобка:

- Пока токен на вершине стека не открывающая скобка
  - Переложить оператор из стека в выходную очередь.
  - Если стек закончился до того, как был встречен токен открывающая скобка, то в выражении пропущена скобка.
- Выкинуть открывающую скобку из стека, но не добавлять в очередь вывода.
- Если токен на вершине стека — функция, переложить её в выходную очередь.

- Если больше не осталось токенов на входе:

- Пока есть токены операторы в стеке:

- Если токен оператор на вершине стека — открывающая скобка, то в выражении пропущена скобка.
- Переложить оператор из стека в выходную очередь.

- Конец.

Исходный код см. в Приложении А.

**Вывод:**

В ходе лабораторной работы были изучены и реализованы следующие списочные структуры: односвязный список, динамический одномерный массив и стек, на основе которого был построен алгоритм сортировочной станции, выполняющий перевод выражения из префиксной нотации в постфиксную.

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <cstdlib>
#include<ctime>

using namespace std;
#define SIZE 100

// ARRAY
class Arr {
private:
    int* arr;
    int capacity;
public:
    Arr(int arrSize)
    {
        arr = new int[arrSize];
        capacity = arrSize;
    };
    ~Arr(){delete[] arr;}

    void fillArray(){
        string n;
        for(int j=0; j<capacity; j++){
            cout<<"Insert int"<<"("<<capacity-j<<" remain): ";
            cin>>n;
            bool wrongType=false;
            for (int i=0; i<n.length(); i++){
                if (isdigit(n[i]) == false){
                    cout<< "number must be integer!"<<endl;
                    j--;
                    wrongType = true;
                    break;
                }
            }
            if(wrongType==false)
                arr[j] = stoi(n);
        }
    }

    void fillRandArray(){
        for(int j=0; j<capacity; j++)
            arr[j] = rand()%20+1;
    }

    void printArray(){
        cout<<"Array: ";
        for(int j=0; j<capacity; j++)
            cout<<arr[j]<<" ";
        cout<<endl;
        cout<<"Array size: "<<capacity<<endl;
    }
```

```
    void deleteElement(int i)
    {
        if (i<0 || i>capacity)
            cout<<"Number is out of range!"<<endl;
        else{
            int *tempArr = new int[capacity-1];
            int k = 0;
            for(int j=0; j<capacity; j++){
                if(j==i)
                    j++;
                tempArr[k]=arr[j];
                k++;
            }
        capacity--;

        arr = new int[capacity];
        for(int j=0; j<capacity; j++)
              arr[j] = tempArr[j];
        delete [] tempArr;
        }
    }

    void addElement(int i, int val)
    {
        if (i<0 || i>capacity)
            cout<<"Number is out of range!"<<endl;
        else{
            int *tempArr = new int[capacity+1];
            int k = 0;
            for(int j=0; j<capacity; j++){
                if(j==i){
                    tempArr[k]=val;
                    k++;
                }
                tempArr[k]=arr[j];
                k++;
            }
        capacity++;

        arr = new int[capacity];
        for(int j=0; j<capacity; j++)
              arr[j] = tempArr[j];
        delete [] tempArr;
        }
    }
};

// LIST
struct Node
{
    int data;
    Node *nextNode;
};
class List
{
private:
    Node *startNode = NULL;
public:
```

```
void addNode(int d)
{
    Node *node = new Node;
    node->data = d;
    node->nextNode = NULL;
    if(startNode == NULL)
        startNode = node;
    else
    {
        Node *thisNode = startNode;
        while(thisNode->nextNode != NULL)
            thisNode = thisNode->nextNode;
        thisNode->nextNode = node;
    }
}
void printList()
{
    Node *thisNode = startNode;
    cout<<"Current list: ";
    while(thisNode->nextNode != NULL)
    {
        cout<<thisNode->data<<" ";
        thisNode = thisNode->nextNode;
    }
    cout<<thisNode->data<<endl;
}

int getListSize()
{
    Node *thisNode = startNode;
    int listSize = 0;
    while(thisNode != NULL){
        listSize+=1;
        thisNode = thisNode->nextNode;
    }
    return listSize;
}

void deleteNode(int i)
{
    if(i>getListSize())
    {
        cout<<"Index is out of range!"<<endl;
    }
    else if(i==0){
        startNode = startNode->nextNode;
    }
    else{
        Node *thisNode = startNode;
        Node *newNode = startNode;
        int j = 0;
        while(j!=i-1)
        {
            j++;
            thisNode = thisNode->nextNode;
        }
        newNode = thisNode->nextNode;
        newNode = newNode->nextNode;
```

```
            thisNode->nextNode = newNode;
        }
    }
};

// STACK + Algorythm
class Stack
{

public:
    string* arr_string;
    int top;
    int capacity;

    Stack(int stackSize = SIZE)
    {
        arr_string = new string[stackSize];
        capacity = stackSize;
        top = -1;
    };
    ~Stack(){delete[] arr_string;}

    void push(string);
    string pop();
    string topElement();

    int stackSize();
    bool isEmpty();
    bool isFull();
    void printStack();
};

void Stack::push(string x)
{
    if (!isFull())
        arr_string[++top] = x;

}

string Stack::pop()
{
    if (!isEmpty())
        return arr_string[top--];
}

string Stack::topElement()
{
    if (!isEmpty()) {
        return arr_string[top];
    }
}

int Stack::stackSize() {
    return top + 1;
}

bool Stack::isEmpty() {
```

```
    return stackSize() == 0;
}

bool Stack::isFull() {
    return stackSize() == capacity;
}

void Stack::printStack(){
    cout<<"Stack: "<< endl;
    for(int i=0; i<=top; i++)
    {
        cout<<arr_string[i]<<" "<<endl;
    }
}

bool check_number(string word) {
    for (int i = 0; i < word.length(); i++)
        if (isdigit(word[i]) == false)
            return false;
    return true;
}

int operatorRank(string str)
{
    if(str=="^")
        return 3;
    if(str=="*" || str=="/")
        return 2;
    if(str=="+" || str=="-")
        return 1;
    return 0;
}

void sortString(Stack& stackOperators, string word)
{
    if(check_number(word))
            cout<<word<<" ";
        else if(word == "sin" || word == "cos")
            stackOperators.push(word);
        else if(word == "(")
            stackOperators.push(word);
        else if(word == ")")
        {
            while(stackOperators.stackSize()>0 && stackOpera-
tors.topElement()!="(")
            {
                cout<<stackOperators.topElement()<<" ";
                stackOperators.pop();
            }
                stackOperators.pop();
                if(stackOperators.topElement()=="sin" || stackOpera-
tors.topElement()=="cos")
                {
                    cout<<stackOperators.topElement()<<" ";
                    stackOperators.pop();
                }
        }
        else
```

```
        {
            while(stackOperators.stackSize()>0  && operator-
Rank(stackOperators.topElement()) >= operatorRank(word))
            {
                cout<<stackOperators.topElement()<<" ";
                stackOperators.pop();
            }
            stackOperators.push(word);
        }
}

void algorythm()
{
    string input;
    string line;
    while(input != "q"){
        Stack stackOperators;

        cout<<"\n\nInput string with spaces (ex: 4 + 5 ^ 2 )"<<endl;
        cout<<"Or press 1 to use default example line"<<endl;
        cout<<"Press q to exit\n"<<endl;
        getline (cin>>ws, input);
        if (input == "q")
            break;
        if(input == "1")
            line = "54 + 6 / sin ( 5 ) * 5 ^ 4 - cos ( 33 + 1 )";
        else
            line = input;

        cout<<"String: "<<line<<endl;
        cout<<"Result: ";

        string word = "";
        for (auto x : line)
        {
            if (x == ' ')
                {
                  sortString(stackOperators, word);
                  word = "";
                }
            else {
              word = word + x;
            }
        }
        sortString(stackOperators, word);

        while(stackOperators.stackSize()>0)
        {
            cout<<stackOperators.topElement()<<" ";
            stackOperators.pop();
        }
    }

}

void arrayInterface()
{
    system("clear");
```

```
    int arrSize;
    cout<<"Input size of Array"<<endl;
    cin>>arrSize;
    Arr arr(arrSize);
    int func;
    string n;
    while(func != 6){
        cout<<"\n1 - input elements manualy\n2 - input random elements\n3
- delete element\n4 - add element\n5 - print array\n6 - exit"<<endl;
        cin>>func;
        system("clear");
        switch(func)
        {
        case 1:
            arr.fillArray();
            arr.printArray();
            break;
        case 2:
            arr.fillRandArray();
            arr.printArray();
            break;
        case 3:
            arr.printArray();
            cout<<"Insert number of element to delete: ";
            cin>>n;
            if(check_number(n)== false)
            {
                cout<< "number must be integer!"<<endl;
                break;
            }
            arr.deleteElement(stoi(n)-1);
            arr.printArray();
            break;
        case 4:
            arr.printArray();
            int val;
            cout<<"Insert number of element to insert: ";
            cin>>n;
            cout<<"Insert value to insert: ";
            cin>>val;
            if(check_number(n)== false)
            {
                cout<< "number and value must be integer!"<<endl;
                break;
            }
            arr.addElement(stoi(n)-1,val);
            arr.printArray();
            break;
        case 5:
            arr.printArray();
            break;
        }
    }
}

void listInterface()
{
```

```
    system("clear");

    List testList;
    int func;
    string n;
    while(func != 6){
        cout<<"\n1 - generate random number of nodes\n2 - add node\n3 -
delete node\n4 - print list\n5 - get list size\n6 - exit"<<endl;
        cin>>func;
        system("clear");
        switch(func)
        {
        case 1:
            for(int i=0; i<rand()%5+1; i++)
            {
                testList.addNode(rand()%100+1);
            }
            testList.printList();
            break;
        case 2:
            cout<<"Enter number"<<endl;
            cin>>n;
            if(check_number(n)== false)
            {
                cout<< "number must be integer!"<<endl;
                break;
            }
            testList.addNode(stoi(n));
            testList.printList();
            break;
        case 3:
            testList.printList();
            cout<<"Enter node number:"<<endl;
            cin>>n;
            if(check_number(n)== false)
            {
                cout<< "node number must be integer!"<<endl;
                break;
            }
            testList.deleteNode(stoi(n)-1);
            testList.printList();
            break;
        case 4:
            testList.printList();
            break;
        case 5:
            cout<<"Size of list: "<<testList.getListSize()<<endl;
            break;
        }
    }
}

int interface()
{
    int func;
    while(func != 4){
        system("clear");
```

```
        cout<<"1 - dynamic array example\n2 - list example\n3 - Shunting
yard algorithm\n4 - exit"<<endl;
        cin>>func;
        switch(func)
        {
        case 1:
            arrayInterface();
            break;
        case 2:
            listInterface();
            break;
        case 3:
            algorythm();
            break;
        case 4:
            return 0;
        }
    }
}

int main()
{
    srand(time(NULL));
    interface();
    return 0;
}
```

## ПРИЛОЖЕНИЕ В

## РАБОТА ПРОГРАММЫ

Для демонстрации работы программы приведу пример вставки узла:




И работу сортировочной станции: