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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек, очередь, дек

Студентка гр. 7383 _____ Прокопенко Н.

Преподаватель _____ Размочаева Н. В.

Санкт-Петербург

2018

Содержание

Цель работы

Познакомиться со структурами данных и научиться реализовывать их на языке программирования C++.

Формулировка задачи: перевести выражение, записанное постфиксной в форме в заданном текстовом файле postfix, в обычную (инфиксную) форму (вариант 11-д-д).

Реализация задачи

В данной работе было написано несколько функций для реализации задачи. Перечень функций:

int main( ) – выводит результат запуска функции postfixToInfix( ).

char* postfixToInfix( ) – функция перевода выражения, записанного в постфиксной форме, в обычную (инфиксную) форму. Данные считываются из файла postfix.txt, если такого файла нет, то выдается ошибка "Error opening file.". Считанная строка посимвольно проверяется функцией bool isSign(char*) на знак операции, если же проверяемый символ не является знаком операции ("+", "-", "*", "^"), то символ добавляется в стек. Если функция bool isSign(char*) возвращает true, то два верхних значения в стеке склеиваются с этим знаком посередине, результат склеивания добавляется в стек. Когда в считанной из файла строке встречается "\0", то извлекается верхний элемент и проверяется, что в стеке пусто, тогда функция возвращает этот элемент. В противном случае выводится ошибка программы.

bool isSign(char*) – сравнивает переданную строку с "+", "-", "*", "^".

Методы класса Stack:

base top (void) – возвращает последний элемент стека.

void pop (void) – удаляет элемент из стека.

base pop2(void) – удаляет элемент из стека, возвращая его значение.

void push (const base &x) – добавляет значение в стек.

bool isNull(void) – проверяет стек на наличие элементов. При отстутствии элементов возвращает true, в противном случае false.

void destroy (void) – удаляет все из стека.

Код программы представлен в Приложении А.

Тестирование

Программа собрана в операционной системе Ubuntu 17.04 с использованием компилятора g++. В других ОС и компиляторах тестирование не проводилось. Результаты тестирования показали, что поставленная цель выполнена. Результаты тестирования представлены в Приложении Б.

Выводы

В ходе выполнения лабораторной работы были изучены основные понятия стеке, был реализован стек на базе массива на языке программирования C++. Также была написана программа для записи выражения из постфиксного в инфиксный вид.

ПРИЛОЖЕНИЕ А. Код программы

main.cpp

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include "st_interf1.h"
#include <fstream>
using namespace std;

namespace st_modul1
{
      struct Stack::node {//
            base *hd;
            node *tl;
            node()
            {
                  hd = NULL; tl = NULL;
            }
      };// end node
        //-------------------------------------
      base Stack::top(void)
      {// PreCondition: not null
            if (topOfStack == NULL) { cerr << "Error: top(null) \n";
exit(1); }
            else return *topOfStack->hd;
      }
      //-------------------------------------
      void Stack::pop(void)
      {// PreCondition: not null
            if (topOfStack == NULL) { cerr << "Error: pop(null) \n";
exit(1); }
            else
            {
                  node *oldTop = topOfStack;
                  topOfStack = topOfStack->tl;
                  delete oldTop->hd;
                  delete oldTop;
            }
      }
      //-------------------------------------
      base Stack::pop2(void)
      {// PreCondition: not null
            if (topOfStack == NULL) { 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 != NULL) {
                topOfStack->hd = new base;
                *topOfStack->hd = x;
                topOfStack->tl = p;
          }
          else { cerr << "Memory not enough\n"; exit(1); }
     }
     //-------------------------------------
     bool Stack::isNull(void)
     {
          return (topOfStack == NULL);
     }
     //-------------------------------------
     void Stack::destroy(void)
     {
          while (topOfStack != NULL) {
                pop();
          }
     }
}

bool isSign(char* c) {

     return !strcmp(c, "+\0") || !strcmp(c, "-\0") || !strcmp(c,
"*\0") || !strcmp(c, "^\0");
}

char* postfixToInfix() {
     st_modul1::Stack stack;
     string c;
     int i=0;
     char* str = new char[2];
     char* res = new char[30];
```

```
	char* operand1 = new char[30];
	char* operand2 = new char[30];
	char* ptr=new char[30];
	ifstream file("postfix.txt");
	if (!file.is_open()) {
		cout << "Error opening file.\n";
		exit(1);
	}
	else {
		getline(file, c);
	c[c.length()]='\0';
	while (i!=c.length()) {
			st_modul1:: base *p = (st_modul1::base*)malloc(30 *
sizeof(st_modul1:: base));
    *p = (st_modul1::base)malloc(30 * sizeof(char));
	while(c[i] == ' ')
			i++;
		res[0] = '\0';
		str[0] = c[i];
		str[1] = '\0';
		if (isSign(str)) {
			operand2 = stack.pop2();
			operand1 = stack.pop2();
			if(!strcmp(str, "+\0") || !strcmp(str, "-\0")){
			strcat(res, "(");
			strcat(res, operand1);
			strcat(res, str);
			strcat(res, operand2);
			strcat(res, ")");
		}
		else {
			strcat(res, operand1);
			strcat(res, str);
			strcat(res, operand2);
		}
		strncpy(*p, res, 30);
		}else	strncpy(*p, str, 30);
		stack.push(*p);
		i++;
	}
	file.close();
	if(!stack.isNull())
		res = stack.pop2();
	if(stack.isNull())
		return res;
	else {
		cout << "Error.\n";
```

```
            exit(1);
            }
      }
}

int main()
{
      cout << postfixToInfix() << endl;
      return 0;
}
```

"st_interf1.h"

```
#include<iostream>

namespace st_modul1
{
             typedef char* base;

      class Stack {
      private:
            struct node;
      node *topOfStack;
      public:
            Stack ()
                  { topOfStack = NULL;
                  };
            base top (void);
            void pop (void);
            base pop2(void);
            void push (const base &x);
            bool isNull(void);
            void destroy (void);
      };
}

int signValue (char* c);
```

ПРИЛОЖЕНИЕ Б. Тестовые случаи

Таблица 1 - Результаты тестов.

Input	Output	True/False
123-*	1*(2-3)	True
Cvb*^	C^v*b	True
Fhdjjkd----	Error	True
Fgh*^v^	F^g*h^v	True
F***	pop(null)	Trur