**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Стековая машина.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Григорьев И.С. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Содержание**

[**Цель работы** 3](#_Toc480921099)

[**Задание** 3](#_Toc480921100)

[**Ход работы** 3](#_Toc480921101)

[**Структура стека** 3](#_Toc480921102)

[**Функция добавления элемента в стек** 3](#_Toc480921103)

[**Функция удаления элемента из стека** 3](#_Toc480921104)

[**Операции над входящей строкой** 4](#_Toc480921105)

[**Вывод** 5](#_Toc480921106)

[**Приложение** 6](#_Toc480921107)

# 

# **Цель работы**

Написание программного кода на языке СИ с реализацией стека на базе списка.

# **Задание**

На вход программе подается последовательность (не более 100 элементов) из чисел и арифметических операций (+, -, \*, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

* Если очередной элемент входной последовательности - число, то положить его в стек
* Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже)
* Если входная последовательность закончилась, то вывести результат (число в стеке)

Если в процессе вычисления возникает ошибка (для операции в стеке не хватает аргументов или по завершении работы программы в стеке более одного элемента), то вместо результата следует вывести "**error**".

# **Ход работы**

## **Структура стека**

Определение структуры узел:

typedef struct Node\_tag {

int num; //переменная для хранения целых чисел

struct Node\_tag \*next; //указатель на следующий узел

}Node\_t;

Для удобства тип struct Node\_tag переименован в Node\_t.

## **Функция добавления элемента в стек**

void push(Node\_t \*\*head, int value) {

Node\_t \*tmp =(Node\_t\*)malloc(sizeof(Node\_t)); //выделяется память под новый узел

tmp->next=\*head; //передвигается указатель на старый узел

tmp->num=value; //присваивается значение

\*head=tmp; //вершина стека передвигается на новый узел

}

## **Функция удаления элемента из стека**

Функция снимает элемент с вершины стека, перемещая вершину к следующему элементу.

void pop(Node\_t \*\*head){

Node\_t \*out; //переменная для очистки старой головы

out=\*head; //запоминается голова

\*head=(\*head)->next; //голове присваивается новый адрес

free(out); //удаляем старую голову

}

## **Операции над входной строкой**

int main() {

int count=0, num1, num2;

Node\_t \*head = NULL;

char \*string=(char\*)malloc(sizeof(char)\*100); //выделение памяти под строку

fgets(string,100,stdin); //считывание строки

char \*pch=strtok(string," "); //Берем первую лексему

while(pch){

if((atoi(pch)!=0)){ //возвращает 0, когда не находит int

push(&head, atoi(pch)); //кладется в стак

count++; //увеличивается размер стека

}

else if(count>=2){

num2=head->num; //запоминается верх стека

pop(&head); //удаляется верх

num1=head->num; //запоминается новый верх стека

pop(&head); //удаляется верх

count--; //в итоге размер всегда будет -1

/\*Лексема проверяется на знаковое соответствие\*/

if(pch[0]=='+')

push(&head,num1+num2);

else if(pch[0]=='-')

push(&head,num1-num2);

else if(pch[0]=='\*')

push(&head,num1\*num2);

else if(pch[0]=='/')

push(&head,num1/num2);

}

else{ //если элементов в стеке не достаточно

printf("error\n");

return 0;

}

pch=strtok(NULL," "); //указатель передвигается на следующую лексему

}

if(count!=1) //проверка на количество элементов

printf("error\n");

else

printf("%d", head->num);

return 0;

}

# **Вывод**

В ходе выполнения данной лабораторной работы были изучены принципы работы со стеком на базе списка.

# **Приложение**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node\_tag {

int num;

struct Node\_tag \*next;

}Node\_t;

void push(Node\_t \*\*head, int value) {

Node\_t \*tmp =(Node\_t\*)malloc(sizeof(Node\_t)); //sozdaem noviy uzel

tmp->next=\*head; //kidaem na stariy uzel

tmp->num=value;

\*head=tmp; //vershinu steka kidaem na noviy uzel

}

void pop(Node\_t \*\*head){ //snimaet element s vershini steka, peremeshaya vershinu k sled elemntu

Node\_t \*out;

out=\*head; //zapominaem staruyu golovu

\*head=(\*head)->next; //prisvaevaem golove noviy adress

free(out);

}

int main() {

int count=0, num1, num2;

Node\_t \*head = NULL;

char \*string=(char\*)malloc(sizeof(char)\*100);

fgets(string,100,stdin);

char \*pch=strtok(string," "); //berem 1st leksemu

while(pch){

if((atoi(pch)!=0)){ //vozvrashaet 0, kogda ne nahodit int

push(&head, atoi(pch)); //kladem v stack

count++; //uvelichivaem razmer stack'a

}

else if(count>=2){

num2=head->num; //zapominaem verh stack'a

pop(&head); //udalyaem verh

num1=head->num; //zapominaem noviy verh stack'a

pop(&head); //udalyaem

count--; //v itoge razmer vsegda budet -1

if(pch[0]=='+')

push(&head,num1+num2);

else if(pch[0]=='-')

push(&head,num1-num2);

else if(pch[0]=='\*')

push(&head,num1\*num2);

else if(pch[0]=='/')

push(&head,num1/num2);

}

else{ //esli elementov v stack'e ne dostatochno

printf("error\n");

return 0;

}

pch=strtok(NULL," "); //peredvigaem ukazatel na sled leksemu

}

if(count!=1) //proverka na kol-vo elementov

printf("error\n");

else

printf("%d", head->num); //pechataem edinstvenniy element v stack'e

return 0;

}