**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Динамические структуры данных. Тестирование.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Иванкова В.М. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Оглавление

[Цель: 3](#_Toc484110814)

[Задание: 3](#_Toc484110815)

[Содержание: 3](#_Toc484110816)

[Инициализация 3](#_Toc484110817)

[Разбиение на лексемы 3](#_Toc484110818)

[Работа стековой машины 3](#_Toc484110819)

[Вывод 5](#_Toc484110820)

[Приложение 6](#_Toc484110821)

Цель:

Написать программу, на вход которой подается последовательность из чисел и арифметических операций, разделенных пробелом, которые программа должна интерпретировать и выполнить по определенным правилам. Стек, который требуется реализовать самостоятельно, требуется выполнить на базе массива.

Задание:

**Стековая машина.**

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".

Стек требуется реализовать самостоятельно на базе массива.

Содержание:

Создание структуры

#define MAX\_STACK 100

typedef int N;

typedef struct index

{

N a[MAX\_STACK];

N size;

} index;

Добавление элемента в стек

void push(index \*stack, const N value)

{

stack->a[stack->size] = value;

stack->size++;

}

Размер стека

int sizeStack(index \*stack)

{

return stack->size;

}

Функция для выполнения арифметических действий

int funct(index \*stack, char c)

{

if (stack->size == 0 || stack->size == 1 )

{

return 1;

}

int a, b;

if(sizeStack(stack) != 0)

{

a=top(stack);

pop(stack);

b=top(stack);

pop(stack);

}

int result;

switch(c)

{

case '+': {

result = b+a;

break;

}

case '-': {

result = b-a;

break;

}

case '\*': {

result = b\*a;

break;

}

case '/': {

result = b/a;

break;

}

}

push(stack, result);

return 0;

}

Вынимаем элемент из стека

int pop(index \*stack)

{

stack->size--;

return stack->a[stack->size];

}

Разбиение на лексемы

char\* str1 = strtok(str, " ");

Работа стековой машины

while(str1 != NULL)

{

if(isdigit(\*str1)) //проверяет аргумент, является ли он десятичной цифрой

push(&stack, atoi(str1));

else

if(strlen(str1) > 1) //если число двузначное или больше

{

push(&stack, atoi(str1));

}

else

if(strlen(str1) == 1) //если это символ, то

{

if(funct(&stack, \*str1) == 1)

control = 1;

}

str1 = strtok(NULL, " \n");

}

if(sizeStack(&stack) == 1)

printf("%d\n", top(&stack));

else

{

control = 1;

printf("error");

return 0;

}

return 0;

}

# Вывод

В ходе выполнения лабораторной работы, используя стандартные средства языка С, была самостоятельно смоделирована такая динамическая структура данных как стек (в данном случае на базе массива), которая позволила удобно хранить и обрабатывать данные для решения поставленной задачи.

Приложение

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX\_STACK 100 //Определяем максимальный размер массива

typedef int N; //Тип данных, которые будут храниться в данном массиве

typedef struct index

{

N a[MAX\_STACK]; // Размер типа для того, чтобы в него можно было записать максимальный размер теоретически возможного массива любого типа

N size; // количество элементов, и вместе с тем указатель на вершину стека

// Вершина будет указывать на следующий элемент массива, в который будет занесено значение.

} index;

void push(index \*stack, const N value)

{

stack->a[stack->size] = value; //указатель на структуру

stack->size++;

}

int pop(index \*stack)

{

stack->size--;

return stack->a[stack->size];

}

int sizeStack(index \*stack)

{

return stack->size;

}

int top(const index \*stack)

{

return stack->a[stack->size - 1];

}

int funct(index \*stack, char c)

{

if (stack->size == 0 || stack->size == 1 )

{

return 1;

}

int a, b;

if(sizeStack(stack) != 0)

{

a=top(stack);

pop(stack);

b=top(stack);

pop(stack);

}

int result;

switch(c)

{

case '+': {

result = b+a;

break;

}

case '-': {

result = b-a;

break;

}

case '\*': {

result = b\*a;

break;

}

case '/': {

result = b/a;

break;

}

}

push(stack, result);

return 0;

}

int main()

{

index stack;

stack.size=0;

char str[100];

int control = 1;

fgets(str, 100, stdin);

char\* str1 = strtok(str, " "); //разбиение на лексемы

while(str1 != NULL)

{

if(isdigit(\*str1)) //проверяет аргумент, является ли он десятичной цифрой

push(&stack, atoi(str1));

else

if(strlen(str1) > 1) //если число двузначное или больше

{

push(&stack, atoi(str1));

}

else

if(strlen(str1) == 1) //если это символ, то

{

if(funct(&stack, \*str1) == 1)

control = 1;

}

str1 = strtok(NULL, " \n");

}

if(sizeStack(&stack) == 1)

printf("%d\n", top(&stack));

else

{

control = 1;

printf("error");

return 0;

}

return 0;

}