**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Стековая машина**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Доброхвалов М.О. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель.** Реализовать стек на базе массива. Научиться работать со стеком.

**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"

Стек требуется реализовать самостоятельно на базе массива.

**Ход работы.**

1. Объявляется структура, в которой будет лежать стек и количество элементов, которые в нем содержатся.

typedef struct data

{

int \*arr;

int size\_of\_arr;

}Data;

1. Описываются функции, которые добавляют элемент в стек, удаляют элемент из стека, берут верхний элемент.
2. void push(Data \*my\_stack, int number)

{

my\_stack->arr[my\_stack->size\_of\_arr] = number;

my\_stack->size\_of\_arr++;

}

void pop(Data \*my\_stack)

{

my\_stack->arr[my\_stack->size\_of\_arr]=0;

my\_stack->size\_of\_arr--;

}

int top(Data \*my\_stack)

{

return my\_stack->arr[my\_stack->size\_of\_arr-1];

}

1. Описывается функция, выполняющая вычисление и добавляет число в стек.

void action(Data \*my\_stack, char c)

{

int a = 0;

int x = top(my\_stack); pop(my\_stack);

int y = top(my\_stack); pop(my\_stack);

switch (c)

{

case '+':

a = y + x;

break;

case '-':

a = y - x;

break;

case '\*':

a = y \* x;

break;

case '/':

a = y / x;

break;

default:

printf("Error");

break;

}

push(my\_stack, a);

}

1. Объявляются переменные, необходимые в программе, считаем поток ввода.

Data \*my\_stack = (Data \*)malloc(sizeof(Data));

my\_stack->arr = (int \*)malloc(100 \* sizeof(int));

my\_stack->size\_of\_arr = 0;

char input[100];

fgets(input, 100, stdin);

1. Разбивая поток ввода на строки и, преобразовывая их в числа, если это возможно, элементы записываются в стек функцией push. Если преобразование невозможно, значит это знак операции, вызывается функция action.

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

while (str != NULL)

{

if (atoi(str))

{

push(my\_stack, atoi(str));

}

else

{

if ((strstr(str, "+") || strstr(str, "-") || strstr(str, "\*") || strstr(str, "/"))

&& (size(my\_stack) > 1))

{

char c = str[0];

action(my\_stack, c);

}

else

{

my\_stack->size\_of\_arr++;

break;

}

}

str = strtok(NULL, " ");

}

1. Если после окончания работы в стеке лежит более одного элемента, выводится “error”. Иначе выводится результат работы программы. Высвобождается память.

if (size(my\_stack) == 1)

printf("%d", top(my\_stack));

else printf("error");

free(my\_stack);

1. Файл с лабораторной работой и отчет загружается на github с помощью консоли:

* Создана новая ветка с помощью команды git checkout –b Dobrokhvalov\_sem2\_lab2
* Добавлены файлы загрузки с помощью команды git add Dobrokhvalov\_sem2\_lab2
* Добавлен комментарий коммита командой git commit –m “Dobrokhvalov\_sem2\_lab2”
* Файлы были загружены командой git push origin Dobrokhvalov\_sem2\_lab2

**Вывод:** В ходе работы был реализован стек на базе массива. Были получены навыки взаимодействия со стеком.