**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Динамические структуры данных.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Куликов М.Л. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы:** провести обзор стандартной библиотеки, реализовать стек на базе односвязного списка.

**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.

**Ход работы:**

1. Подключены необходимые для работы заголовочные файлы

#include <stdio.h>

#include <stdlib.h>

1. Объявлена структура – , на базе которого будет реализован стек.

typedef struct Stack\_tag {

int data[STACK\_MAX\_SIZE];

size\_t size;

} Stack\_t;

1. Объявлены и определены все функции, необходимые для работы стековой машины (проблема выхода за пределы массива учтена).

* Функция для помещения элемента в стек

void push(Stack\_t \*stack, const int value) {

if (stack->size >= STACK\_MAX\_SIZE) {

exit(0);

}

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

stack->size++;

}

* Функция для удаление элемента из стека

int pop(Stack\_t \*stack) {

if (stack->size == 0) {

printf("error\n");

exit(0);

}

stack->size--;

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

}

1. В main()

* Объявлены переменная-структура, переменная ch, поле структуры size, отвечающее за размер массива, обнулено.

Stack\_t stack;

stack.size = 0;

char ch;

* Поданная строка считывается посимвольно в цикле while, объявляются вспомогательные переменные.

while ((ch = getchar()) != EOF) {

int number, temp1, temp2 = 0;

int sign = 1;

В цикле символ обрабатывается оператором множественного выбора.

switch(ch) {

case ' ' :

case '\n' :

break;

case '+' :

push(&stack, pop(&stack) + pop(&stack));

break;

case '\*' :

push(&stack, pop(&stack) \* pop(&stack));

break;

case '/' :

temp1 = pop(&stack);

temp2 = pop(&stack);

if(temp1==0){

printf("error\n");

exit(0);

}

push(&stack, temp2 / temp1);

break;

case '-' :

ch = getchar();

if ((ch == ' ') || (ch == EOF)) {

temp1 = pop(&stack);

temp2 = pop(&stack);

push(&stack, temp2 - temp1);

break;

} else {

sign = -1;

}

default :

ungetc(ch, stdin);

scanf("%d", &number);

push(&stack, number\*sign);

break;

}

* При удачном выходе из цикла производится проверка – если в стеке осталось больше одного числа, программа выводит «error», в противном случае - элемент, оставшийся в стеке, и завершает работу.

if(stack.size == 1){

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

} else {

printf("error\n");

}

return 0;

}

1. Затем файл main.c с кодом написанной программы и отчёт о проведенной работе был загружен на github c помощью консоли:

* Создана новая ветка kulikov\_sem2\_lr2

git checkout –b kulikov\_sem2\_lr2

* Файлы с кодом программы и отчетом были добавлены для загрузки

git add kulikov\_sem2\_lr2

* Добавлен комментарий

git commit –m “2 semestr. Lab 2”

* Файлы загружены на репозиторий

git push origin kulikov\_sem2\_lr2

**Вывод:** в ходе работы были получены навыки самостоятельной реализации стека, а также был отработан навык работы с обратной польской записью.