**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 |  | Филиппенко Д.Р. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы:** 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 списка.

**Ход работы:**

1. Подключены необходимые для работы заголовочные файлы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

2. Объявлена структура – односвязный список, на базе которого будет реализован стек.

typedef struct List

{

int a;

struct List\* next;

} List;

3. Определены функции, необходимые для дальнейшей работы стековой машины.

* Функция Push – добавление элемента в стек

void push(List\*\* head, int a)

{

List\* temp = (List\*)malloc(sizeof(List));

temp->a = a;

temp->next = \*head;

\*head = temp;

}

* Функция Pop – удаление элемента из стека

void pop(List\*\* head)

{

List\* last;

last = \*head;

\*head = (\*head)->next;

free(last);

}

* Функция Top – возвращение верхнего элемента

int top(List\* head)

{

return head->a;

}

* Функция Size – определение количества элементов в стеке

int size(List\* head) //функция, определяющая размер стека

{

int size = 0;

while (head)

{

size++;

head = head->next;

}

return size;

}

* Функция, определяющая действия с числами, в зависимости от того, что ввел пользователь

void func(List\*\* head, char c)

{

int a, b;

a = top(\*head);

pop(head);

b = top(\*head);

pop(head);

switch (c)

{

case '+':

push(head, (b + a));

break;

case '-':

push(head, (b - a));

break;

case '\*':

push(head, (b\*a));

break;

case '/':

push(head, (b / a));

break;

default:

printf("error");

}

}

4. В main()

* Объявлены указатель head на верхний элемент стека, массив char str[100], в которую будет записана поданная строка.

List\* head = NULL;

   char str[100];

fgets(str, 100, stdin);

Также объявлена логическая переменная, для своевременного прекращения работы программы. Изначально приравниваем ее к нулю.

int log=0;

* Заданная строка делится на лексемы, которые обрабатываются в цикле while.

char\* stroka;

stroka = strtok(str, " \n");

while(stroka)

В while(stroka):

* Выполняется проверка с помощью функции atoi(), является ли данный элемент числом

Если это верно, то элемент добавляется в стек

if (atoi(stroka) != 0)

{

push(&head, atoi(stroka));

}

* Если заданный элемент не число, а знак операции, то выполняется проверка размера стека

Для выполнения операции над добавленными элементами, необходимо, чтобы их было больше либо равно 2.

Проверяем это условие. Если оно истинно, то применяем функцию к последним двум элементам. Если оно ложно, то выводится ошибка.

else

{

if (size(head) >= 2)

{

func(&head, stroka[0]

}

else

{

("error");

log=1;

break;

}

}

* После того, как выполнены все действия над элементами, проверяем размер стека. Если в нем находится больше 1 элемента, то выводим ошибку. Если в стеке 1 число – результат математических операций, то выводим его на экран.

if(log==0)

{

if (size(head)>1)

     {

     printf("error");

}

else

{

printf("%d",top(head));

}

}

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

* Создана новая ветка Filippenko\_sem2\_lr2

git checkout –b Filippenko\_sem2\_lr2

* Создана новая директория Filippenko\_sem2\_lr2 и файл lr2.c в ней, куда помещен код написанной программы

mkdir Filippenko\_sem2\_lr2

cd Filippenko\_sem2\_lr2

Созданные файлы добавлены для загрузки

git add Filippenko\_sem2\_lr2

* Добавлен комментарий

git commit –m “LR2 done”

* Файлы загружены на репозиторий Git

git push origin Filippenko\_sem2\_lr2

**Вывод:** в ходе работы получены навыки реализации стека на основе односвязного списка и функций, необходимых для его работы.