**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: Моделирование стека

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Цыганов М.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы**

Необходимо реализовать программу, которая моделирует работу стека.

**Задание**

Моделирование стека.

Требуется написать программу, моделирующую работу стека, реализовав перечисленные ниже методы. Программе на вход подается последовательность команд с новой строки (не более 100 команд), в зависимости от которых программа выполняет ту или иную операцию и выводит результат ее выполнения с новой строки.

Перечень команд:

push n - добавляет целое число n в стек. Программа должна вывести "ok"

pop - удаляет из стека последний элемент и выводит его значение на экран

top - программа должна вывести верхний элемент стека на экран не удаляя его из стека

size - программа должна вывести количество элементов в стеке

exit - программа должна вывести "bye" и завершить работу

Если в процессе вычисления возникает ошибка (например вызов метода pop при пустом стеке), программа должна вывести "error" и завершиться.

Стек требуется реализовать самостоятельно на базе массива.

Функции

1. Push n-добавляет число в стек

stack\* push(stack\* sc, int i)

{

sc->n=i;

sc->sec = (stack\*)malloc(sizeof(stack));

sc->sec->first = sc;

sc = sc->sec;

printf("ok\n");

return sc;

}

1. Pop выводит число и удаляет его из стека.

stack\* pop(stack \*sc, int\* err)

{

if (sc->first != NULL)

{

printf("%d\n", sc->first->n);

sc = sc->first;

free(sc->sec);

return sc;

}

else

printf("error\n");

\*err=1;

}

1. Top-выводит последнее число стека.

stack\* top(stack \*sc, int\* err)

{

if (sc->first != NULL)

{

printf("%d\n", sc->first->n);

return sc;

}

else

printf("error\n");

\*err=1;

}

1. Size выводит длину стека.

int size(stack \*sc)

{

int i=0;

while (sc->first != NULL)

{

i++;

sc = sc->first;

}

return i;

}

Исходный код программы:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct stack

{

int n;

struct stack\* sec;

struct stack\* first;

} stack;

stack\* push(stack\* sc, int i)

{

sc->n=i;

sc->sec = (stack\*)malloc(sizeof(stack));

sc->sec->first = sc;

sc = sc->sec;

printf("ok\n");

return sc;

}

stack\* pop(stack \*sc, int\* err)

{

if (sc->first != NULL)

{

printf("%d\n", sc->first->n);

sc = sc->first;

free(sc->sec);

return sc;

}

else

printf("error\n");

\*err=1;

}

stack\* top(stack \*sc, int\* err)

{

if (sc->first != NULL)

{

printf("%d\n", sc->first->n);

return sc;

}

else

printf("error\n");

\*err=1;

}

int size(stack \*sc)

{

int i=0;

while (sc->first != NULL)

{

i++;

sc = sc->first;

}

return i;

}

int main()

{

int i=0;

int err=0;

stack\* sc=(stack\*)malloc(sizeof(stack));

sc->first=NULL;

char input[100];

char \*qwe;

while((strcmp(input, "exit") !=0) && (err == 0))

{

gets(input);

if (strcmp("\0", input)!=0)

{

qwe = strtok (input," ");

if (strcmp("push", qwe) == 0)

{

qwe = strtok (NULL," ");

i=atoi(qwe);

sc = push(sc, i);

}

if (strcmp("pop", qwe) == 0)

{

sc = pop(sc, &err);

}

if (strcmp("top", qwe) == 0)

{

sc = top(sc, &err);

}

if (strcmp("size", qwe) == 0)

{

printf("%d\n", size(sc));

}

}

}

if (!err)

printf("bye\n");

return 0;

}

Выполняются действия в зависимости от введенных команд: размер численно равен индексу текущего элемента, добавление элемента увеличивает индекс текущего элемента на 1, удаление уменьшает на 1.

**Вывод.**

В данной работе была реализована программу, моделирующую работу стека. И написали несколько функций, выполняющих заданных в условии дейтсвий.