**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: Моделирование стека

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Ковынев М.В. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Оглавление

[Цель работы 3](#_Toc478915831)

[Задание 3](#_Toc478915832)

[Содержание 3](#_Toc478915833)

[Вывод 5](#_Toc478915834)

# **Цель работы**

Реализация программы, моделирующая работу стека и выполняющая определенный перечень команд.

# **Задание**

Программе на вход подается последовательность команд с новой строки, 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 списка.

# 

# **Содержание**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Stack // Структура, описывающая стек

{

int value; // Значение

struct Stack\* next; // Указатель на следующий элемент стека

};

typedef struct Stack Stack; // Объявление типа данный Stack

int go = 1; // Переменная выхода из программы

// 1 - ошибок не обнаружено, 0 - присутствует ошибка

/\* Добавление элемента в стек \*/

void push(struct Stack \*\*head)

{

int value = 0;

scanf(" %d", &value); // Считывание добавляемого значения

struct Stack \*tmp = (Stack\*)malloc(sizeof(struct Stack));

tmp->value = value; // Добавление значения в новую структуру

tmp->next = \*head; // Осуществление связи

\*head = tmp;

printf("ok\n");

}

/\* Удаление элемента из стека \*/

void pop(struct Stack \*\*head)

{

if (\*head) // Проверка на существование стека

{

struct Stack \*tmp = \*head;

printf("%d\n", (\*head)->value); // Вывод удаляемого значения на экран

\*head = (\*head)->next; // Смещение указателя на новое значение головы

free(tmp);

}

else

{

printf("error\n"); // Стек отсутствует -> удаление невозможно

go = 0; // Ошибка, остановка работы программы

}

}

/\* Вывод верхнего элемента стека \*/

void top(struct Stack \*head)

{

if (head == NULL)

{

printf("error\n"); // Стек отсутствует -> удаление невозможно

go = 0; // Ошибка, остановка работы программы

}

else

{

printf("%d\n", head->value); // Вывод верхнего элемента стека

}

}

/\* Количество элементов в стеке \*/

void size(Stack\* head)

{

int count = 0; // Инициализация счетчика - количество элементов

while (head != NULL) // Пока существует стек, увеличиваем счетчик

{

count++;

head = head->next;

}

printf("%d\n", count); // Вывод значения

}

int main()

{

Stack\* head = NULL;

char\* string = (char\*)malloc(sizeof(char) \* 5); // Выделение памяти под строку

while (go == 1) // Проверка на наличие ошибок

{

fgets(string, 5, stdin);

if (!(strcmp(string, "push")))

{

push(&head); // Если команда push - выполняем push();

}

else if (!(strcmp(string, "pop\n")))

{

pop(&head); // Если команда pop - выполняем pop();

}

else if (!(strcmp(string, "top\n")))

{

top(head); // Если команда top - выполняем top();

}

else if (!(strcmp(string, "size")))

{

size(head); // Если команда size - выполняем size();

}

else if (!(strcmp(string, "exit")))

{

printf("bye\n"); // Остановка работы программы

return 0;

}

}

return 0;

}

# **Вывод**

Выполнив данную лабораторную работу, было освоено и закреплено на практике написание программы, моделирующая работу стека и выполняющая определенный перечень команд.