**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе№2**

**по дисциплине «Программирование»**

Тема: **Моделирование стека**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Пискунов Я.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Оглавление**

[Цель работы 3](#_Toc481522990)

[Формулировка задачи 3](#_Toc481522991)

[Ход работы 4](#_Toc481522992)

[1. Реализация Стека 4](#_Toc481522993)

[2. Распознавание команд 4](#_Toc481522994)

[Вывод 4](#_Toc481522995)

[Приложение 1 5](#_Toc481522996)

**Цель работы**

Моделирование стека с помощью массива и симуляция основных принципов работы стека.

**Формулировка задачи**

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.

**Ход работы**

1. **Реализация Стека**

При реализации стека на базе массива, требуется обратить внимание на работу с памятью. Память под стек выделяется блоками, чтобы не использовать часто функцию **realloc()**.

if (size == STACKSZ\*STACKSZ\_Ammount){

STACKSZ\_Ammount++; stak\_massive=(int\*)realloc(stak\_massive,sizeof(int)\*STACKSZ\*STACKSZ\_Ammount);

}

1. **Распознавание команд**

Lля распознавания команд и аргументов используется следующий принцип: изначально номерной интерпретации команды присвоено занчени е 404(отсутствует/неверна). При считывании команды проверяется ее соответствие одной из списка и интерпретация изменяется на соответствующий номер. Если команда на сходится ни с одной из списка, то значение останется на 404 и в результате применения мы получим ошибку.

**Вывод**

В ходе данной лабораторной работы был смоделирован стек с использованием массива. Для этого использовались стандартные возможности и функции языка Си. Также была реализована симуляция основных действий стека с числовыми данными на основе команд, задаваемых пользователем.

**Приложение 1**

Код программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define STACKSZ 500

int main() {

int STACKSZ\_Ammount = 1;

int ElementToPush = 0;

int size = 0;

int\* stak\_massive = (int\*)malloc(STACKSZ\*sizeof(int));

char\* commandLine = (char\*)malloc(100\*sizeof(char));

int commandNumber = 404;

while(1==1) {

if (size == STACKSZ\*STACKSZ\_Ammount){

STACKSZ\_Ammount++;

stak\_massive = (int\*)realloc(stak\_massive, sizeof(int)\*STACKSZ\*STACKSZ\_Ammount);

}

fgets(commandLine, 100, stdin);

if(strchr(commandLine, '\n'))

\*strchr(commandLine, '\n') = '\0';

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

commandNumber = 2;

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

commandNumber = 3;

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

commandNumber = 4;

if(strcmp(commandLine, "exit") == 0)

commandNumber = 5;

else if (strstr(commandLine, "push ") && strlen(commandLine) >= 6)

{

commandLine = commandLine + 5\*sizeof(char);

ElementToPush = atoi(commandLine);

commandNumber = 1;

}

switch (commandNumber) {

case 1:

stak\_massive[size] = ElementToPush;

printf("ok\n");

size++;

break;

case 2:

if (size!=0){

printf("%d\n", stak\_massive [size-1]);

size--;

}

else {

printf("error\n");

return 0;

} break;

case 3:

if (size!=0)

printf("%d\n", stak\_massive[size-1]);

else {

printf("error\n");

return 0;

} break;

case 4:

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

break;

case 5:

printf("bye\n");

free(stak\_massive);

return 0;

break;

case 404:

printf("error\n");

return 0;

break;

}

}

return 0;

}