**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**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Смотрова А.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Оглавление

[Цель работы 3](#_Toc481927039)

[Формулировка задачи 3](#_Toc481927040)

[Ход работы 4](#_Toc481927041)

[Вывод 6](#_Toc481927042)

[Приложение 7](#_Toc481927043)

Цель работы

Написать программу, моделирующую работу стека с использованием массива.

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

Для работы программы необходимо подключение следующих библиотек языка С:

* <stdio.h> - printf(), fprintf(), fgets()
* <stdlib.h> - malloc(), realloc(), free()
* <string.h> - strstr(), strchr(), strcmp()

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

Необходимо реализовать функции, указанные в задании.

int\* pop(int\* stackArray, int\* num) *//функция рор*

{

printf("%d**\n**", stackArray[\*(num)-1]);

(\*num)-=1;

return stackArray;

}

int\* push (int\* stackArray, int\* num, int value) *//функция push*

{

stackArray[(\*num)] = value;

(\*num)++;

printf("ok**\n**");

return stackArray;

}

int size(int num) *//функция size*

{

printf("%d**\n**", num);

return 0;

}

void top(int\* stackArray, int num) *//функция top*

{

printf("%d**\n**", stackArray[num-1]);

}

И дополнительные функции, чтобы обеспечить гибкость вносимых изменений в сообщения программы: функция bye и функция errorMessage.

void bye(int\* stackArray){ *//функция выхода и освобождения памяти*

printf("bye**\n**");

free(stackArray);

}

void errorMessage(){ *//функция вывода ошибки*

fprintf(stdout, "error**\n**");

}

Кроме того, нужно учесть, что список команд может расширяться, поэтому стоит вынести сравнение сообщений пользователя с названием функций отдельно:

int commandCheck(char\* StrForCommand, int\* valueToPush){ *//функция проверки команд*

if(strcmp(StrForCommand, "pop") == 0) *//возвращает 2, если команда рор*

return 2;

if(strcmp(StrForCommand, "top") == 0) *//аналогично у следующих*

return 3;

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

return 4;

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

return 5;

else if (strstr(StrForCommand, "push ") && strlen(StrForCommand) >= 6) *//если команда push*

{

StrForCommand = StrForCommand + 5\*sizeof(char); *// убираем push с пробелом до элемента*

\*valueToPush = atoi(StrForCommand); *//преобразование char в int*

return 1;

}

return 0;

}

Вывод

В данной работе была реализована программа, моделирующая работу стека, с помощью функций стандартной библиотеки языка Си, с использованием массива.

Приложение

Полный код программы представлен ниже:

*/\**

*Лабораторная работа №2 на моделирование стека с использованием массива*

*Выполнила: Смотрова А. А.*

*\*/*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define STACKLEN 100 //длина блока стека

void addBlock (int stackBlock, int\* stackArray){ *//функция выделения памяти для еще одного блока стека*

stackBlock+=1; *//увеличиваем количество блоков*

stackArray = (int\*)realloc(stackArray, sizeof(int\*)\*STACKLEN\*stackBlock); *//перевыделяем память*

}

int\* pop(int\* stackArray, int\* num) *//функция рор*

{

printf("%d**\n**", stackArray[\*(num)-1]);

(\*num)-=1;

return stackArray;

}

int\* push (int\* stackArray, int\* num, int value) *//функция push*

{

stackArray[(\*num)] = value;

(\*num)++;

printf("ok**\n**");

return stackArray;

}

int size(int num) *//функция size*

{

printf("%d**\n**", num);

return 0;

}

void bye(int\* stackArray){ *//функция выхода и освобождения памяти*

printf("bye**\n**");

free(stackArray);

}

void top(int\* stackArray, int num) *//функция top*

{

printf("%d**\n**", stackArray[num-1]);

}

void errorMessage(){ *//функция вывода ошибки*

fprintf(stdout, "error**\n**");

}

int commandCheck(char\* StrForCommand, int\* valueToPush){ *//функция проверки команд*

if(strcmp(StrForCommand, "pop") == 0) *//возвращает 2, если команда рор*

return 2;

if(strcmp(StrForCommand, "top") == 0) *//аналогично у следующих*

return 3;

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

return 4;

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

return 5;

else if (strstr(StrForCommand, "push ") && strlen(StrForCommand) >= 6) *//если команда push*

{

StrForCommand = StrForCommand + 5\*sizeof(char); *// убираем push с пробелом до элемента*

\*valueToPush = atoi(StrForCommand); *//преобразование char в int*

return 1;

}

return 0;

}

int main() {

int stackBlock = 1; *//количество блоков, изначально равно 1*

int valueToPush = 0; *//элемент, который будет закидываться на стек*

int num = 0; *//длина массива*

int\* stackArray = (int\*)malloc(STACKLEN\*sizeof(int)); *//выделяем память под массив стека*

char\* StrForCommand = (char\*)malloc(100\*sizeof(char)); *//выделяем память по строку команд*

while(1) { *//бесконечный цикл, пока не "exit"*

if (num == STACKLEN\*stackBlock){ *//если количество элементов уже равно размеру блока*

addBlock(stackBlock, stackArray); *//добавляем еще один*

}

fgets(StrForCommand, 30, stdin); *//читаем команду*

if(strchr(StrForCommand, '**\n**')) { *//удаляем перенос строки*

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

}

switch (commandCheck(StrForCommand, &valueToPush)) { *//сверяем команды и производим действия*

case 1: *//если 1, то добавляем элемент*

stackArray = push(stackArray, &num, valueToPush);

**break**;

case 2: *//если 2, то удаляем последний элемент и выводим его*

if (num!=0){

stackArray=pop(stackArray, &num);

}

else {

errorMessage();

return 0;

} **break**;

case 3: *//если 3, то выводим последний элемент*

if (num!=0)

top(stackArray, num);

else {

errorMessage();

return 0;

} **break**;

case 4: *//если 4, выводим длину массива*

size(num);

**break**;

case 5: *//если 5, то прощаемся и очищаем память*

bye(stackArray);

return 0;

**break**;

case 0: *//в любом другом случае - выводим ошибку*

errorMessage();

return 0;

**break**;

}

}

return 0;

}