**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. 6303 |  | Куценок Л.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы.**

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. В локальном репозитории git была создана новая ветка *kutsenok\_leonid\_sem2\_lr2* командой

git checkout -b kutsenok\_leonid\_sem2\_lr2

1. Создана папка *kutsenok\_leonid\_sem2\_lr2*

mkdir kutsenok\_leonid\_sem2\_lr2

и файл *main.c*

touch main.c

1. Далее приведено краткое описание работы программы.
2. Пока в потоке stdin есть данные, в переменную line считываются команды. Если на вход поступила команда push, то с потока считывается число, которое нужно положить в стек; команда pop – из стека удаляется верхний элемент и его значение выводится на экран; команда top – значение верхнего элемента стека выводится на экран; команда size – количество элементов стека выводится на экран; команда exit – программа завершается

**while(**fgets**(**line**,** LINE\_SIZE**,** stdin**)** **!=** NULL**)** **{**

token **=** strtok**(**line**,** " \n"**);**

**if(**strcmp**(**token**,** FUNC\_PUSH**)** **==** **0)** **{**

**if(**sscanf**(**strtok**(**NULL**,** " \n"**),** "%d"**,** **&**val**))** **{**

push**(**stack**,** val**);**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_POP**)** **==** **0)** **{**

**if(**pop**(**stack**,** **&**val**))** **{**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_TOP**)** **==** **0)** **{**

**if(**top**(**stack**,** **&**val**))** **{**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_SIZE**)** **==** **0)** **{**

printf**(**"%d\n"**,** stack**->**size**);**

**}** **else** **if(**strcmp**(**token**,** FUNC\_EXIT**)** **==** **0)** **{**

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

**}**

**}**

1. Функция init инициализурет стек, выделяя память под массив данных стека и устанавливая его размер значением 0

**void** init**(**Stack**\*** stack**)** **{**

stack**->**data **=** **(int\*)** malloc**(**INITIAL\_STACK\_SIZE **\*sizeof(int));**

stack**->**size **=** **0;**

**}**

1. Функция push добавляет в стек переменную data и увеличивает его размер на 1

**void** push**(**Stack**\*** stack**,** **int** data**)** **{**

stack**->**data**[**stack**->**size**++]** **=** data**;**

**}**

1. Функция pop записывает значение верхнего элемента стека в переменную val и удаляет его

**int** pop**(**Stack**\*** stack**,** **int\*** val**)** **{**

**if(**stack**->**size **==** **0)** **{**

**return** **0;**

**}** **else** **{**

**\***val **=** stack**->**data**[--**stack**->**size**];**

**return** **1;**

**}**

**}**

1. Функция top записывает значение верхнего элемента стека в переменную val

**int** top**(**Stack**\*** stack**,** **int\*** val**)** **{**

**if(**stack**->**size **==** **0)** **{**

**return** **0;**

**}** **else** **{**

**\***val **=** stack**->**data**[**stack**->**size **-** **1];**

**return** **1;**

**}**

**}**

1. Полный код программы

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <string.h>*

*#define INITIAL\_STACK\_SIZE 100*

*#define LINE\_SIZE 20*

*#define FUNC\_PUSH "push"*

*#define FUNC\_POP "pop"*

*#define FUNC\_TOP "top"*

*#define FUNC\_SIZE "size"*

*#define FUNC\_EXIT "exit"*

**typedef** **struct** Stack **{**

**int\*** data**;**

**int** size**;**

**}** Stack**;**

**void** init**(**Stack**\*** stack**);**

**int** top**(**Stack**\*** stack**,** **int\*** val**);**

**int** pop**(**Stack**\*** stack**,** **int\*** val**);**

**void** push**(**Stack**\*** stack**,** **int** data**);**

**int** main**()** **{**

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

init**(**stack**);**

**char\*** line **=** **(char\*)** malloc**(**LINE\_SIZE **\*** **sizeof(char));**

**char\*** token**;**

**int** val **=** **0;**

**while(**fgets**(**line**,** LINE\_SIZE**,** stdin**)** **!=** NULL**)** **{**

token **=** strtok**(**line**,** " \n"**);**

**if(**strcmp**(**token**,** FUNC\_PUSH**)** **==** **0)** **{**

**if(**sscanf**(**strtok**(**NULL**,** " \n"**),** "%d"**,** **&**val**))** **{**

push**(**stack**,** val**);**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_POP**)** **==** **0)** **{**

**if(**pop**(**stack**,** **&**val**))** **{**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_TOP**)** **==** **0)** **{**

**if(**top**(**stack**,** **&**val**))** **{**

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

**}** **else** **{**

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

**return** **0;**

**}**

**}** **else** **if(**strcmp**(**token**,** FUNC\_SIZE**)** **==** **0)** **{**

printf**(**"%d\n"**,** stack**->**size**);**

**}** **else** **if(**strcmp**(**token**,** FUNC\_EXIT**)** **==** **0)** **{**

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

**}**

**}**

**}**

**int** top**(**Stack**\*** stack**,** **int\*** val**)** **{**

**if(**stack**->**size **==** **0)** **{**

**return** **0;**

**}** **else** **{**

**\***val **=** stack**->**data**[**stack**->**size **-** **1];**

**return** **1;**

**}**

**}**

**int** pop**(**Stack**\*** stack**,** **int\*** val**)** **{**

**if(**stack**->**size **==** **0)** **{**

**return** **0;**

**}** **else** **{**

**\***val **=** stack**->**data**[--**stack**->**size**];**

**return** **1;**

**}**

**}**

**void** push**(**Stack**\*** stack**,** **int** data**)** **{**

stack**->**data**[**stack**->**size**++]** **=** data**;**

**}**

1. В локальный репозиторий git была добавлена папка с исходным кодом программы

git add kutsenok\_leonid\_sem2\_lr2

1. Сделан коммит в локальный репозитрий

git commit

1. Изменения отправлены в удаленный репозиторий

git push origin kutsenok\_leonid\_sem2\_lr2

1. Сделан *pull request* в ветку *master*

**Выводы.**

В ходе лабораторной работы изучены возможности реализации стека на языке Си. Полученные знания закреплены в ходе разработки программы для управление стеком с помощью команд, вводимых с терминала.