Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____Информатика и системы управления_____

КАФЕДРА ____Программное обеспечение ЭВМ и информационные технологии_____

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3
“Работа со стеком”

Студент_____Татаринова Дарья Алексеевна_____
фамилия, имя, отчество

Группа_____ИУ7-34Б_____

Выполнил _____ ___Татаринова Д.А.___
подпись, дата *фамилия, и.о.*

Принял _____ ____Барышникова М.Ю.___
подпись, дата *фамилия, и.о.*

2021 г.

Цель работы

Цель работы - реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации: получить представление о механизмах выделения и освобождения памяти при работе со стеком.

Задание

Создать программу работы со стеком, выполняющую операции добавление, удаления элементов и вывод текущего состояния стека. Реализовать стек:

а) массивом;

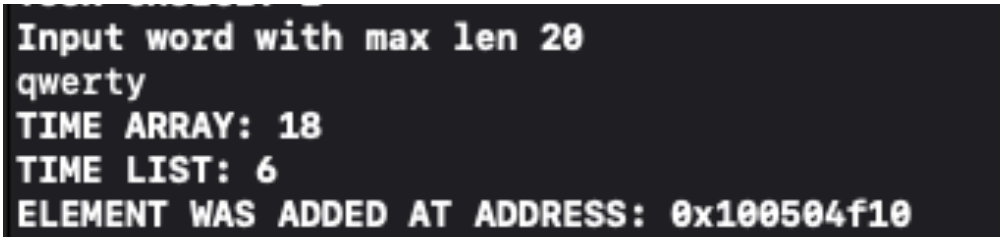
б) списком.

Все стандартные операции со стеком должны быть оформлены подпрограммами. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой список или массив свободных областей (адресов освобождаемых элементов) с выводом его на экран.

Элементами стека являются слова. Распечатать слова в обратном порядке в перевернутом виде.

Входные данные

Слова, длина которых не больше 20 символов.



Выходные данные

При выборе соответствующего пункта меню будет выведено актуальное состояние стека или же все слова в перевернутой форме в обратном порядке.

Способ обращения к программе

Программа может быть вызвана через консоль.

Команда сборки:

gcc -std=c99 -Wall -Werror -Wextra -Wpedantic -Wfloat-conversion -Wvla -Wfloat-equal -c main.c

gcc -std=c99 -Wall -Werror -Wextra -Wpedantic -Wfloat-conversion -Wvla -Wfloat-equal -c stack_list.c

gcc -std=c99 -Wall -Werror -Wextra -Wpedantic -Wfloat-conversion -Wvla -Wfloat-equal -c stack_array.c

gcc -std=c99 -Wall -Werror -Wextra -Wpedantic -Wfloat-conversion -Wvla -Wfloat-equal -c word.c

gcc main.o stack_list.o stack_array.o word.o -o app.exe

Аварийные ситуации

В случае аварийной ситуации выводится сообщение о той или иной ошибке.

Могут быть выведены такие ошибки, как:

- Некорректный ввод пункта меню (введено не число)
- Нехватка памяти при попытке добавить элемент

Структуры данных

Для хранения стека в виде массива и списка использовались структуры.

```
#define WORD_LEN_MAX 21

typedef char word_t[WORD_LEN_MAX];


// стек в виде массива
typedef struct {
    word_t *arr; // сам стек
    int cur_ind; // верхушка стека
    int max_capasity; // максимальный размер стека
```

```
} stack_array_t;


// стек в виде списка
typedef struct node
{
    word_t value; // значение узла списка
    struct node *next; // указатель на следующий узел
} node_t;

typedef struct {
    node_t *head; // указатель на голову списка
} stack_list_t;
```

Описание алгоритма

Данная программа предназначена для работы с стеком и представляет собой консольное приложение со следующими возможными операциями, представленными в меню:

```
1 - Add word
2 - Delete word
3 - Print stack
4 - Show freed addresses
5 - Print all words in reverse order and form
0 - Program exit.
```

Пример добавления слова:

```
YOUR CHOICE: 1
Input word with max len 20
qwerty
TIME ARRAY: 18
TIME LIST: 6
ELEMENT WAS ADDED AT ADDRESS: 0x100605a50
```

Пример вывода стека:

```
YOUR CHOICE: 3
--------ARRAY-------
qwerty
--------------------
TIME ARRAY: 66

--------LIST--------
qwerty
--------------------
TIME LIST: 17
```

Пример вывода стека с перевернутыми словами:

```
YOUR CHOICE: 5
--------ARRAY-------
ytrewq
--------------------
TIME ARRAY: 80

--------LIST--------
ytrewq
--------------------
TIME LIST: 17
```

Пример удаления слова:

```
YOUR CHOICE: 2
TIME ARRAY: 8
TIME LIST: 4
DELETED ELEMENT: milk
```

Если стек пустой:

```
YOUR CHOICE: 2
STACK IS EMPTY
```

Вывод освобожденных адресов:

```
YOUR CHOICE: 4
0x10061fbe0
0x10061fbc0
```

Сравнение памяти (в байтах)

Количество элементов	Массив	Список
10	88	280
50	408	1400
100	808	2800
250	2008	7000
500	4008	14000
750	6008	21000
1000	8008	28000

Сравнение времени (в тактах)

Операция	Массив	Список
Добавление	10	13
Удаление	2	12

Контрольные вопросы

1. Что такое стек?

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: LIFO - последним пришел – первым ушел.

2. Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?

Если хранить стек как список, то память выделяется в куче. Если хранить как массив — либо в куче, либо на стеке (зависит от того, динамически или статический массив используется). Для каждого элемента стека, который хранится как список, выделяется на 4 или 8 байт (если брать современные ПК) больше, чем для элемента стека, который хранится как массив. Данные байты использованы для хранения указателя на следующий элемент списка. (из-за этого либо 4, либо 8 байт)

3. Каким образом освобождается память при удалении элемента стека при различной реализации стека?

Если хранить стек как список, то верхний элемент удаляется при помощи операции освобождения памяти для него и смещением указателя, который указывает на начало стека.

При хранении стека как массива, память удаляется при завершении программы при вызове функции free (если работа идет с динамической памятью).

4. Что происходит с элементами стека при его просмотре?

Элементы стека удаляются, так как каждый раз достается верхний элемент стека, чтобы посмотреть следующий.

5. Каким образом эффективнее реализовывать стек? От чего это зависит?

Стек эффективнее реализовать с помощью массива, если выигрыш по памяти важнее и в виде списка, если важнее время работы программы.

Вывод

Для реализации стека выгодно использовать массив, так как он занимает меньше памяти (примерно в 3 раза), чем список, и работает быстрее на 10-20%.

Однако при использовании массива размер стека сильно ограничен. Если массив был создан статически, то его размер не изменить, если динамически – придется постоянно реаллоцировать память. При использовании же списка такой проблемы не возникает.