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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«РАБОТА СО СТЕКОМ»

по курсу «Типы и структуры данных»

Студент: Шимшир Эмирджан Османович

Группа: ИУ7-33Б

Студент _____ Шимшир Э.О.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М.Ю.

подпись, дата *фамилия, и.о.*

Оценка _____

Условие задачи

Вариант 3

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. Распечатайте слова в обратном порядке в перевернутом виде.

Техническое задание

Входные данные

Целое число от 0 до 8 – пункт меню

Слова в виде строк

Выходные данные

Вывод текущего состояние стека на массиве

Вывод текущего состояние стека на списке

Вывод слов в обратном порядке в перевернутом виде

Задачи, реализуемые программой

1. Ввод слов в стек на массиве и списке
2. Удаление слов из стека на массиве и списке
3. Вывод стека на массиве и списке на экран
4. Вывод слов в обратном порядке в перевернутом виде

Допущения

Необходимо по умолчанию вводить только целые числа

Необходимо вводить слова только на английском языке

Максимальный размер стека, который может выбрать пользователь – 1000 элементов

Максимальный размер строки 10 символов

Описание внутренних структур данных

Структура для хранения данных в стеках:

```
typedef struct
{
    int len_word;
    char word[MAX_WORD_LEN + 1];
} data_t;
```

int len_word; - длина слова
char word[MAX_WORD_LEN + 1]; - массив символов слова

В памяти матрица хранится в виде массива указателей на строки

MAX_WORD_LEN = 10

Структура для хранения стека на массиве:

```
typedef struct
{
    data_t array[MAX_ARRAY_STACK_LEN];
    int len;
} array_stack_t;
```

data array[MAX_ARRAY_STACK_LEN]; - массив данных
int len; - длина массива

Память под стек на массиве статическая *MAX_ARRAY_STACK_LEN = 1000*

Структура для хранения стека на списке:

```
typedef struct list_stack_t list_stack_t;

Struct list_stack_t
{
    data_t data;

    int index;
    list_stack_t *next;
};
```

list_stack_t - элемент стека (узел списка)
data_t data - структура с данными
int index - индекс элемента списка
*list_stack_t *next;* - указатель на следующий элемент стека

Максимальный размер стека на списке *MAX_LIST_STACK_LEN = 1000*

Описание меню и функций программы

```
emir_shimshir@192 ~/D/g/B/lab_04 (lab_04)> ./app.exe
Программа для работы со стеком
Шимшир Эмирджан ИУ7-33Б В-24

(по умолчанию необходимо вводить только целые числа)
(слова необходимо вводить только на английском языке)


Пункты меню:
1 - Добавить слова в стек на массиве
2 - Удалить слова из стека на массиве
3 - Вывести текущее состояние стека на массиве
4 - Добавить слова в стек на списке
5 - Удалить слова из стека на списке
6 - Вывести текущее состояние стека на списке
7 - Вывести адреса освобожденных областей
8 - Распечатать слова в обратном порядке в перевернутом виде
0 - выход из программы
Выберете пункт меню:
```

1. Пользователь выбирает пункт меню

2. Пользователь добавляет слова в стек (пункты 1, 4)

3. Пользователь удаляет слова из стека (пункты 2, 5)

4. Пользователь выводит слова из стека на экран (пункты 3, 6)

5. Пользователь выводит адреса освобожденных областей стека на списке (пункт 7)

6. Пользователь печатает слова в обратном порядке в перевернутом виде (пункт 8)

Описание кодов ошибок

Ниже описаны значения ошибок и их коды:

```
#define EXIT_SUCCESS 0
#define ERR_FULL_STACK 1
#define ERR_EMPTY_STACK 2
#define ERR_ACTION 3
#define ERR_READ_STRING 4
#define ERR_STRING_SIZE 5
#define ERR_EMPTY_STRING 6
#define ERR_SPACE 7
#define ERR_WORDS_COUNT 8
#define ERR_FULL_FREE_AREAS 9
#define ERR_EMPTY_FREE_AREAS 10
```

Ограничения на входные данные

Необходимо по умолчанию вводить только целые числа

Необходимо вводить слова только на английском языке

Описание функций

```c
/*
* Функция push для стека на массиве
*
* Принимает указатель на стек на массиве
* Принимает указатель на структуру с данными,
* которую необходимо добавить в стек
*/
int array_push(array_stack_t *stack, data_t data);
```

```c
/*
* Функция pop для стека на массиве
*
* Принимает указатель на стек на массиве
* Принимает указатель на структуру с данными,
* которую необходимо удалить из стека
*/
int array_pop(array_stack_t *stack, data_t *data);
```

```c
/*
* Функция вывода стека на списке
*
* Принимает указатель на начало списка
*/
int array_print(array_stack_t *stack);
```

```c
/*
* Функция push для стека на списке
*
* Принимает указатель на начало списка
* Принимает указатель на структуру с данными,
* которую необходимо добавить в стек
*/
int list_push(list_stack_t **stack, data_t data);
```

```c
/*
* Функция pop для стека на списке
*
* Принимает указатель на начало списка
* Принимает указатель на структуру с данными,
* которую необходимо удалить из стека
*/
int list_pop(list_stack_t **stack, data_t *data);
```

```c
/*
* Функция вывода стека на списке
*
* Принимает указатель на начало списка
*/
int list_print(list_stack_t **stack);
```

Описание алгоритма и исследование полученных результатов

Описание алгоритма

Стек на массиве реализован на основе статического массива. При добавлении элемента в стек, реализованный на статическом массиве, достаточно добавить новый элемент в массив по индексу, равному текущей длине стека и затем увеличить его размер на единицу. При удалении элемента из стека, реализованном на статическом массиве, достаточно переместить «указатель» на предыдущий элемент (уменьшить его длину на единицу). Так в стеке есть возможность работать только с верхним элементом, при просмотре элементы исключаются из стека, поэтому необходимо создавать копию стека перед просмотром и возвращать его в первоначальное состояние для сохранения.

Стек на списке реализован на основе односвязного списка. При добавлении элемента в стек, реализованный на основе односвязного списка, достаточно выделить память для нового узла, сделать так, чтобы он указывал на вершину стека, а затем сделать новый узел вершиной стека. При удалении элемента из стека на списке память, отведенная под данный элемент, освобождается, а указатель на вершину стека перемещается на следующий элемент. Так в стеке есть возможность работать только с верхним элементом, при просмотре элементы исключаются из стека, поэтому необходимо создавать копию стека перед просмотром и возвращать его в первоначальное состояние для сохранения.

Полученные результаты

Результаты измерения времени и памяти при различном размере списков. Для каждого случая проводилось 5 экспериментов. При проведении экспериментов программа была скомпилирована без оптимизаций (**-O0**), внешние задачи отсутствовали.

Время в микросекундах

Количество элементов	Добавление элементов в массив	Удаление элементов в массиве	Вывод слов в обратном порядке в массиве	Добавление элементов в список	Удаление элементов в списке	Вывод слов в обратном порядке в списке
10	4 (100%)	1 (100%)	14 (100%)	18 (450%)	1 (100%)	23 (165%)
100	15 (100%)	7 (100%)	121 (100%)	39 (260%)	12 (170%)	137 (115%)
1000	61 (100%)	55 (100%)	16403 (100%)	270 (450%)	140 (255%)	20536 (125%)

Память в байтах

Количество элементов	Массив	Список
10	16004	320
100	16004	3200
1000	16004	32000

Тестирование

Позитивные тесты

Входные данные	Действия программы	Выходные данные
Пункт 1 Корректный ввод элементов стека на массиве	Добавление элементов в стек на массиве	Ожидание следующего ключа
Пункт 2 Корректный ввод количества удаляемых элементов стека на массиве	Удаление элементов из стека на массиве	Ожидание следующего ключа
Пункт 3 Наличие непустого стека на массиве	Вывод стека на массиве на экран	Ожидание следующего ключа
Пункт 4 Корректный ввод элементов стека на списке	Добавление элементов в стек на списке	Ожидание следующего ключа
Пункт 5 Корректный ввод количества удаляемых элементов стека на списке	Удаление элементов из стека на списке	Ожидание следующего ключа
Пункт 6 Наличие непустого стека на списке	Вывод стека на списке на экран	Ожидание следующего ключа
Пункт 7 Наличие очищенных областей стека на списке	Вывод очищенных адресов на экран	Ожидание следующего ключа
Пункт 8 Наличие непустых стеков	Вывод слов в обратном порядке в перевернутом виде	Ожидание следующего ключа
Пункт 0	Завершение программы	Выход с кодом 0

Негативные тесты

Входные данные	Действия программы	Выходные данные
Пункт 1 Полный стек на массиве	Ошибка, стек полный	Ожидание следующего ключа
Пункт 2 Пустой стек на массиве	Ошибка, стек пустой	Ожидание следующего ключа
Пункт 3 Пустой стек на массиве	Ошибка, стек пустой	Ожидание следующего ключа
Пункт 4 Полный стек на списке	Ошибка, стек полный	Ожидание следующего ключа
Пункт 5 Пустой стек на списке	Ошибка, стек пустой	Ожидание следующего ключа
Пункт 6 Пустой стек на списке	Ошибка, стек пустой	Ожидание следующего ключа
Пункт 7 Наличие очищенных областей стека на списке	Ошибка, массив очищенных областей пустой	Ожидание следующего ключа
Пункт 8 Пустой стек на списке и на массиве	Стек на массиве Пустой стек Стек на списке Пустой стек	Ожидание следующего ключа

Контрольные вопросы

Что такое стек?

Стек – структура данных, в которой можно обрабатывать только последний добавленный элемент (верхний элемент). На стек действует правило LIFO — last in first out - последним пришел, первым вышел.

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

При реализации стека на статическом массиве память выделяется на стеке, размер зависит от выбранного программистом максимального количества элементов в стеке. При реализации стека на списке память выделяется отдельно под каждый элемент при его добавлении, размер памяти под конкретный элемент в моем случае составляет 32 байт.

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

При удалении элемента из стека, реализованном на статическом массиве, достаточно переместить «указатель» на предыдущий элемент. При удалении элемента из стека на списке память, отведенная под данный элемент, освобождается, а указатель на вершину стека перемещается на следующий элемент.

Что происходит с элементами стека при его просмотре?

Так в стеке есть возможность работать только с верхним элементом, при просмотре элементы исключаются из стека, поэтому необходимо создавать копию стека перед просмотром и возвращать его в первоначальное состояние для сохранения.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Реализовывать стек эффективнее по времени с помощью статического массива. Он выигрывает как во времени обработки, так и в количестве занимаемой памяти при большом количестве элементов. Вариант хранения списка может быть предпочтительнее только в том случае, если нам нужно занять всю оперативную память элементами в программе.

Вывод

На основе полученных значений времени обработки стеков на массиве и списке и используемой при этом памяти делаем вывод, что стек, реализованный связанным списком, проигрывает по времени обработки. Стек на статическом массиве особенно выгодно использовать при заполненности стека близкой к максимальной. По памяти стек на списке будет выгоднее при количестве элементов менее половины от максимального количества, так как память под элементы выделяется динамически.

Таким образом, можно сделать вывод, что статический массив использовать лучше, чем список для реализации стека, если скорость работы важнее памяти или если стек при использовании программы всегда будет заполнен не менее чем на половину.