	Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
«Работа со стеком"

Студент, группы ИУ7-31Б
Костев Дмитрий

Москва 2020

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.

Описание условия задачи

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

- массивом;
- 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.

При реализации стека массивом располагать два стека в одном массиве. Один стек располагается в начале массива и растет к концу, а другой располагается в конце массива и растет к началу. Заполнять и освобождать стеки произвольным образом с экрана. Элементами стека являются вещественные числа. Списком реализовать один стек.

Техническое задание:

Входные данные:

1. Целое число, представляющее собой номер команды: целое число в диапазоне от 0 до 7.
2. Командно-зависимые данные:
- вещественные числа

Выходные данные:

1. Состояние стеков
2. Количественная характеристика сравнения вариантов обработки стека.

Интерфейс программы:

===Массив===

1. Распечатать весь стек
2. Добавить элемент в стек
    1. Идущий с начала
    2. Идущий с конца
3. Вытащить элемент из стека
    1. Идущий с начала
    2. Идущий с конца

===Список===

4. Распечатать весь стек
5. Добавить элемент в стек

6. Вытащить элемент из стека
7. Провести анализ

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 7 или меньшее, чем 0.
   На выходе: сообщение об ошибке

2. Попытка извлечь элемент из пустого стека.
   На входе: пустой стек.
   На выходе: сообщение об ошибке

3. Некорректный ввод числа.
   На входе: символ или буква.
   На выходе: сообщение об ошибке

Обращение к программе

Исполняемый файл app.exe. Запускается из командной строки, аргументы строки не требуются.

Используемые структуры

Структура для хранения масиива:

```
typedef struct array_stack {

    double array[ARRAY_CAPACITY];

    double *ptr_start;
    double *ptr_end;

    double *cur_ptr_start;
    double *cur_ptr_end;

} array_stack_t;
```

Структура для хранения списка:

```
typedef struct node {
    struct node* next;
    double value;
} node_t;
```

Алгоритм

1. Пользователю выводится меню
2. Пользователь вводит номер команды
3. Выполняется действие согласно номеру команды

- Стек - массив

1. Распечатать весь стек с двух сторон
   Проходится по всем элементам массива и выводит их
2. Добавить элемент в стек
   Смещение указателя
   Присваивание значения
3. Вытащить элемент из стека
   Выдача значения
   Смещение указателя

- Стек - список

4. Распечатать весь стек
   Пробегается по всем элементам списка и выводит их, при этом печатает их
   Выводит свободные области оперативной памяти
5. Добавить элемент в стек
   Создается узел, указывающий на голову списка
   В дескрипторе указатель на голову перенаправляют на новосозданный узел
6. Вытащить элемент из стека
   Выводится значение головы
   Запоминается адрес головы
   В дескрипторе указатель на голову перенаправляют на следующий (второй) элемент списка
   Высвобождается старая голова

- Сравнение

7. Провести анализ
   Перебираются различные длинны стека
   Создаются новые стеки для каждого измерения
   Выводится таблица с численными и объёмными характеристиками вставки, удаления элементов в стеках разных видов.

Основные функции

```
// Добавление элемента в стек массива
int add_array_item(array_stack_t *array);

// Получение элемента стека массива
int get_array_item(array_stack_t *array);

// Добавление элемента в стек списка
int push_list_stack(list_stack_t *stack);

// Получение элемента стека списка
int pop_list_stack(list_stack_t *stack, double *element);

// Сравнение методов
int compare_methods();
```

Тесты

Описание	Входные данные	Выходные данные
Неправильный номер команды	-1	Сообщение об ошибке
Неправильный номер команды	20	Сообщение об ошибке
Успешное добавление элемента в стек массива с начала	2 1 11.35	
Успешное добавление элемента в стек массива с конца	2 2 11.35	
Вывод стека	1 или 4	
Попытка вытащить элемент из пустого стека	3 или 6 и пустой стек	Сообщение об ошибке

Оценка эффективности

- Время работы указано в тактах процессора.
- Объём матриц указан в байтах.

Stack's size	Time of adding		Time of deletion		Memory	
	array	list	array	list	array	list
1	1	0	0	0	40	32
2	1	1	0	1	48	48
4	1	1	0	1	64	80
8	1	1	0	1	96	144
16	1	2	0	2	160	272
32	1	3	0	3	288	528
64	1	5	1	6	544	1040
128	2	10	2	12	1056	2064
256	4	21	4	25	2080	4112
512	6	42	7	49	4128	8208
1024	10	76	13	93	8224	16400
2048	20	149	26	181	16416	32784
4096	39	295	50	366	32800	65552
8192	74	593	99	720	65568	131088
16384	149	1181	198	1433	131104	262160
32768	297	2351	392	2882	262176	524304
65536	591	4653	786	6791	524320	1048592

*В массиве память посчитана только из количества заполненных элементов.

Контрольные вопросы

1. Что такое стек?

Стек – структура данных, в которой предусмотрена только обрабатка последнего элемент (верхний элемент). На стек действует правило - последний вошел, первым вышел.

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

- Для каждого элемента стека, реализованного списком, выделяется память для хранения указателя и содержания элемента.
- Для каждого элемента стека, реализованного массивом, выделяется память только для хранения содержания элемента.

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

- При хранении стека связанным списком, верхний элемент удаляется освобождением памяти для него и смещением указателя, указывающего на начало стека.

- При удалении из стека, реализованного массивом, смещается лишь указатель на вершину стека.

4. Что происходит с элементами стека при его просмотре?

Все элементы стека удаляются, так как каждый раз достается верхний элемент стека.

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

- Если колличество элементов заранее известно, и размер узла списка не превышат в два раза размер содержимого, то эффективнее использовать массив. Иначе при нехватке пямяти целесообразно использовать список.

Вывод

При длинне стека 8192 стек-массив быстрее в шесть раза и менее затратен в памяти в два раза. Но мною был реализован стек через статический массив, значит при использовании динамического, выигрыш времени был бы в примерно два раза меньше, так как нужно время на выделение и освобождение памяти.

Очевидно, что использование стека-списка, целесообразно, когда размер указателя на следующий элемент намного меньше, чем размер содержимого узла.
Также возможно использование стека-списка для экономии оперативной памяти в случаях, когда размер стека заранее не известен.

В остальных случаях, т.е. когда максимальный размер стека заранее известен, целесообразней использовать стек, построенный на массиве.