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 №4 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент **Звягин Даниил Олегович**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Звягин Д.О.**

Преподаватель _____ **Барышникова М. Ю.**

Преподаватель _____ **Никульшина Т.А.**

Оценка _____

2023 г.

Условие задачи
Создать программу работы для работы с типом данных «стек». Используя созданный тип данных, распечатать убывающие серии последовательности целых чисел в обратном порядке.

Техническое задание
Реализовать стек: двумя способами – с помощью массива и списка. Оформить операции работы со стеком в виде подпрограмм. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой массив свободных областей (адресов освобождаемых элементов) с выводом его на экран. Реализовать операции работы со стеком, который представлен в виде массива и в виде списка, оценить преимущества и недостатки каждой реализации. Найти путь в лабиринте с помощью стека. Используя созданный тип данных, распечатать убывающие серии последовательности целых чисел в обратном порядке.

Входные данные:
Можно передать имя файла как аргумент командной строки, тогда программа попытается считать содержимое стека из файла.
При дальнейшей работе программы (или если файл не передан), пользователю будет предложено меню.
Пункты меню:

0. Выход
1. Вывести стеки (содержимое)
2. Вывести адреса элементов стека в виде списка
3. Добавить элемент в стек (push)
4. Удалить последний элемент (pop)
5. Вывести серии (по заданию)
6. Провести замеры

При окончании работы программы, перед очисткой стеков, будут выведены адреса всех элементов стека через список, а также отдельно адреса, расстояние в памяти между которыми больше, чем размер узла списка (то есть фрагментированные данные)
Все пункты меню всегда работают для обоих стеков кроме пункта 2 (указано в названии пункта)

Выходные данные:
В зависимости от пункта меню: Содержимое стеков, адреса элементов, статус выполнения команд, серии последовательности, результаты профилирования, сообщения об ошибках.

Возможные аварийные ситуации:
Невозможность открыть файл для ввода стека

Нехватка памяти

Способ обращения к программе

Собрать программу с помощью make release;
Запустить программу с помощью файла app.exe;
Можно передать имя файла с исходными данными.

Структуры данных

```
// перечисление пунктов меню
enum menu_items
{
    EXIT,
    PRINT,
    PRINT_ADRESSES,
    PUSH,
    POP,
    PRINT_SERIES,
    PROFILING,
};

// структура статического массива со счётчиком элементов
typedef struct {
  int pb[10000];
  size_t el_count;
} my_arr_t;

// перечисление возможных ошибок при работе со стеком через массив
enum stack_errors
{
    DA_STACK_EMPTY = 2,
};

// Структура стека через массив
typedef struct
{
    my_arr_t arr;
    int *PS;
} dynarr_stack_t;

// перечисление возможных ошибок при работе со списками
enum list_errors
{
    L_NOT_ENOUGH_MEMORY = 1,
};

// Структура одного узла списка
typedef struct my_list_node
{
    int value;
```

```
    struct my_list_node *next;
} my_list_node_t;

// Перечисление ошибок при работе со стеком через список
enum list_stack_errors
{
    L_STACK_EMPTY = 3,
};

// структура стека через список
typedef struct
{
    my_list_t list;
} list_stack_t;
```

Алгоритм определения убывающей последовательности

Записываем последовательность в стек.
(необязательно для алгоритма, но для удобства) создаём копию стека
Удаляем последний элемент из стека и выводим его.
Сохраняем два последних элемента стека, удаляя элементы из стека циклически.
Если последний элемент больше предыдущего, значит в изначальной последовательности они были частями убывающей серии => просто выводим его
Иначе выводим элемент с новой строки – он будет началом (а изначально - концом) новой серии чисел

Пример работы

```
❯ ./app.exe ./data/test.txt
Считываем: 3 2 1 6 5 4 5 6
Из файла ./data/test.txt считано 8 элементов
Меню:
0: Выход
1: Вывести стеки
2: Вывести адреса элементов стека через список
3: Добавить элемент в стек
4: Удалить последний введённый элемент
5: Вывести серии
6: Провести замеры
>5
dynarr_stack series:
6
5
4 5 6
1 2 3

list_stack series:
6
5
4 5 6
1 2 3
```

Замеры

Замеры проводятся по N раз (отдельная колонка в таблице). Все замеры времени усредняются по количеству замеров. Затраты памяти для каждого замера одинаковы.

Время и память вывода убывающих серий в наносекундах и байтах соответственно:

Кол-во эл-тов	N	Время		Память		Преимущество массива по t, %	Преимущество массива по mem, %
		arr	list	arr	list		
8	1000	71	166	40016	128	233,80%	0,32%
20	1000	142	390	40016	320	274,65%	0,80%
50	1000	339	1081	40016	800	318,88%	2,00%
100	1000	609	2177	40016	1600	357,47%	4,00%
250	1000	1579	5770	40016	4000	365,42%	10,00%
500	1000	3440	12156	40016	8000	353,37%	19,99%
1000	1000	8188	23546	40016	16000	287,57%	39,98%
5000	1000	58515	114985	40016	80000	196,51%	199,92%
10000	1000	103606	231242	40016	160000	223,19%	399,84%

Выводы по проделанной работе

Стек, реализованный связанным списком, стабильно проигрывает стеку, выполненному в виде статического массива по скорости от 2х до 3.5 раз, однако при количестве элементов, меньшем, чем 2500, стек, выполненный в качестве статического массива проигрывает стеку, выполненному в виде списка по памяти. Таким образом, можно сделать вывод, что если нужно реализовать такую структуру данных как стек, то нужно, ориентируясь на объём доступной памяти и ожидаемое количество элементов выбрать либо стек в виде массива – для прироста в скорости, а также меньшему объёму занимаемой памяти при кол-ве элементов, большем, чем 2500. Однако если вы рассчитываете обрабатывать меньшее количество элементов и вам критически важен объём занимаемой памяти, то можно выбрать список (или задать меньший размер массива (или выполнить стек в виде динамического массива)).

Контрольные вопросы

Что такое стек?

Стек – структура данных, работающая по правилу Last In First Out, в которой можно обрабатывать только последний добавленный элемент.

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

При хранении стека с помощью списка, мы выделяем блок динамической памяти для каждого узла. При хранении с помощью статического массива, память выделяется на стеке. Для каждого элемента стека, реализованного списком, выделяется значительно большее количество памяти, чем для очередного элемента массива. Эти дополнительные байты занимает указатель на следующий элемент списка. Размер указателя (4 или 8 байт) зависит от архитектуры.

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

При хранении стека связанным списком, верхний элемент удаляется путем освобождения памяти для него и смещения указателя, указывающего на начало стека. При удалении из стека, реализованного массивом, смещается лишь указатель на вершину стека, а сами данные на самом деле остаются в памяти, но считаются «стёртыми».

Что происходит с элементами стека при его просмотре?

Элементы стека уничтожаются, так как каждый раз достается верхний элемент стека.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Реализовывать стек эффективнее с помощью массива. Он выигрывает как во времени обработки, так и в количестве занимаемой памяти. Но, если не известен размер стека, то в таком случае стоит использовать списки, так как ограничением занимаемой ими памяти является только объём памяти в компьютере.