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 «РАБОТА СО СТЕКОМ»

Студент Артемьев Илья Олегович

Группа ИУ7 – 33Б

2020 г.

Описание условия задачи

Разработать программу работы со стеком, реализующую операции добавления и удаления элементов из стека и отображения текущего состояния стека. Реализовать стек: а) массивом; б) списком.

Все стандартные операции со стеком должны быть оформлены отдельными подпрограммами. В случае реализации стека в виде списка при отображении текущего состояния стека предусмотреть возможность просмотра адресов элементов стека и создания дополнительного собственного списка свободных областей (адресов освобождаемой памяти при удалении элемента, который можно реализовать как списком, так и массивом) с выводом его на экран. Список свободных областей необходим для того, чтобы проследить, каким образом происходит выделение памяти менеджером памяти при запросах на нее и убедиться в возникновении или отсутствии фрагментации памяти.

Указания к выполнению работы

Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:

- указание формата и диапазона вводимых данных,
- блокирование ввода данных, неверных по типу,
- указание операции, производимой программой:
  - добавление элемента в стек,
  - удаление элемента из стека,
  - вычисление (обработка данных);
- наличие пояснений при выводе результата.

Кроме того, нужно вывести на экран время выполнения программы при реализации стека списком и массивом, а также указать требуемый объем памяти. Необходимо так же выдать на экран список адресов освобождаемых элементов при удалении элементов стека.

При тестировании программы необходимо:

- проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода данных, неверных по типу);
- обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
- проверить правильность выполнения операций;
- обеспечить вывод соответствующих сообщений при попытке удаления элемента из пустого стека;
- отследить переполнение стека.

При реализации стека в виде списка необходимо:

- ограничить доступный объем оперативной памяти путем указания: максимального количества элементов в стеке; максимального адреса памяти, превышение которого будет свидетельствовать о переполнении стека;
- следить за освобождением памяти при удалении элемента из стека.

Описание технического задания

Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные.

Входные данные:

1. **Целое число, представляющее собой номер команды**:

   целое число в диапазоне от 0 до 5.

2. **Командно-зависимые данные:**

   целочисленные значения (элементы стека)

Выходные данные:

Результат выполнения определенной команды.

Функции программы:

1 - Добавить элементы в стеки (количество)

2 - Удалить элементы из стеков (количество)

3 - Вывод текущего состояния стеков

4 - Отсортировать два стека в третий

5 - Вывести время и память, затраченные на сортировку стеков в разных реализациях (массив и список)

1 – Выйти

Обращение к программе:

Запускается через терминал с помощью команды ./app.exe.

Аварийные ситуации:

1. Некорректный ввод номера команды.

   На входе: число, большее чем 5 или меньшее, чем 0.

   На выходе: сообщение «Ошибка: неверно введен номер пункта меню»

2. Совершение сортировки при пустом одном или более стеке.

   На входе: целое число в диапазоне от 4 до 5 (номер команды).

   На выходе: сообщение «Ошибка: пустой стек»

3. Добавление элемента в стек, когда он заполнен полностью.

   На входе: элемент стека.

   На выходе: сообщение «Ошибка: произошло переполнение стека»

ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ

Реализация стека с помощью линейного односвязного списка.

```
typedef struct number_t
{
    int data;

    struct number_t *next;
} number_t;
```

Поля структуры:

data – элемент стека

****next*** – указатель на следующий элемент списка

Реализация стека с помощью массива.

Вспомогательная структура — массив свободных адресов.

```
typedef struct
{
    number_t *free_array_points[MAX_LEN_ARRAY_FREE];

    int len;
} free_points_t;
```

Поля структуры:

****free_array_points*** – указатель на массив свободных адресов

MAX_LEN_ARRAY_FREE = 100 – *м*аксимальное количество элементов в массиве

Структура данных — стек с помощью массива.

```
typedef struct
{
    int *data;

    size_t top;
    size_t size;
} array_stack_t;
```

Поля структуры:

top — количество элементов в стеке

size — максимальное количество элементов в стеке

****data*** – указатель на массив элементов стека

ОПИСАНИЕ АЛГОРИТМА

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.
3. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

НАБОР ТЕСТОВ

	Название теста	**Пользовательский ввод**	**Результат**
1	Некорректный ввод команды	-1	Ошибка: неверно введен номер пункта меню

2	Добавление элемента в заполненный стек	1) Выбрать стек 2) Ввести количество элементов 3) Ввести элементы через пробел	Ошибка: произошло переполнение стека
3	Совершение сортировки при пустом одном или более стеке	Команда 4 или 5	Ошибка: пустой стек
4	Добавление элементов в стек (команда 1)	1) Выбрать стек 2) Ввести количество элементов 3) Ввести элементы через пробел	Элементы успешно добавлены в стек
5	Удаление элементов из стека (команда 2)	1) Выбрать стек 2) Ввести количество удаляемых элементов	Элементы успешно удалены из стека
6	Вывод текущего состояния стеков на экран (команда 3)	Вызов команды (3).	Все стеки выведены на экран + массив свободных указателей
7	Сортировка двух стеков в 3 (команда 4)	Вызов команды (4).	Сортировка прошла успешно
8	Вывод статистики на экран (команда 5)	Вызов команды (5).	Статистика

Оценка эффективности

Измерения эффективности сортировок будут производиться в секундах.

Размер	Массив	Список
10	0.000157	0.000350
100	0.041181	0.181675
250	0.486192	2.998741
500	3.859808	25.609883

Оценка эффективности по памяти

Размер	Массив	Список
10	40	160
100	400	1600
250	1000	4000
500	2000	8000

Из данных таблиц можно сделать вывод, что стек реализованный массивом всегда выигравает по памяти и по времени сортировки.

Ответы на 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 массив, память освобождается при завершении программы.

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, так как он выигрывает в количестве занимаемой памяти и во времени обработки стека.

В стеке, реализованном с помощью массива, легче просматривать элементы, сдвигая указатель, а также не надо хранить указатель на следующий элемент, как в списке.

Вывод

Если стек реализовать массивом, то он будет выигрывать по памяти в ~ 4 раза. Это связанно с тем, что для хранения стека в виде списка требуется память, чтобы хранить указатели.
Так же при реализация стека в виде массива требуется меньше времени на обработку. Это связанно с тем, что при реализации массивом доступ к нужному элементу получить проще, требуется лишь передвинуть указатель. Если реализовать в виде списка, то требуется время для удаления верхнего элемента (верхушки стека) и перестановки указателя.

Можно сделать вывод, что для хранения стека лучше использовать массив, это выгоднее и по памяти и по времени.

Так же можно сделать вывод, что в результате тестирования фрагментации памяти не выявлено.