

	<p>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)</p>
---	--

ФАКУЛЬТЕТ _____ Информатика и системы управления (ИУ) _____

КАФЕДРА _____ Программное обеспечение ЭВМ и информационные технологии (ИУ7) _____

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 **«РАБОТА СО СТЕКОМ»**

Студент, группа

Кононенко С., ИУ7-33Б

2019 г.

Описание условия задачи

Создать программу работы со стеком, выполняющую операции добавления, 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.

Распечатать убывающие серии последовательности целых чисел в обратном порядке.

Техническое задание

Входные данные:

1. **Целое число, представляющее собой номер команды:** целое число в диапазоне от 0 до 5.
2. **Командно-зависимые данные:**
 - целочисленные значения (количество элементов стека, значения элементов стека, адреса).

Выходные данные:

1. Результат выполнения индивидуального задания – убывающие подпоследовательности в исходной последовательности.
2. Количественная характеристика сравнения вариантов обработки стека.

Функция программы: программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Ввести элементы стека.
2. Добавить элемент в стек.
3. Удалить элемент из стека.
4. Вывести массив освободившихся адресов.
5. Вывести убывающие подпоследовательности в обратном порядке и вывести количественную характеристику обработки.

Обращение к программе: запускается из терминала.

Аварийные ситуации:

1. Некорректный ввод номера команды.
На входе: число, большее чем 6 или меньшее, чем 0.
На выходе: сообщение «Введена недопустимая команда! Повторите попытку.»
2. Некорректный ввод количества элементов стека.
На входе: отрицательное целое число или буква.
На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
3. Некорректный ввод лимитирующего адреса для стека.
На входе: неположительное целое число или буква.
На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
4. Некорректный ввод элемента стека.
На входе: буква или любой другой нечисловой символ.
На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
5. Попытка создать новый стек, при имеющемся в программе.
На входе: попытка создания нового стека.
На выходе: сообщение «Стек уже существует. Выход из программы...»

Структуры данных

Реализация стека на основе массива:

```
typedef struct
{
    int top;
    int capacity;
    int *arr;
} arrstack_t;
```

Поля структуры:

- **int top** – верхушка стека;
- **int capacity** – максимальная ёмкость стека;
- **int *arr** – указатель на массив.

Реализация стека на основе линейного односвязного списка:

```
typedef struct liststack
{
    int data;
    struct liststack *next;
} liststack_t;
```

Поля структуры:

- **int data** – значение элемента стека;
- **struct liststack *next** – указатель на нижестоящий элемент.

Реализация массива свободных областей:

```
typedef struct
{
    size_t *arr;
    int capacity;
    int ind;
} arr_t;
```

Поля структуры:

- **size_t *arr** – указатель на массив адресов;
- **int capacity** – максимальная ёмкость массива;
- **int ind** – текущий незанятый элемент.

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с матрицами.
3. При вводе стека, стек сразу хранится двумя способами представления (на основе массива и на основе линейного односвязного списка).
4. При выборе команды добавить/удалить элемент из стека, элемент добавляется/удаляется сразу из двух реализаций.
5. При выборе команды вывода массива свободных областей, выводится массив свободных областей в том случае, если какие-либо элементы были удалены из стека.
6. При выборе команды нахождения убывающих подпоследовательностей, решение реализуется сразу на обоих представлениях и выводится количественная характеристика выполнения задачи.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод команды	6	Введена недопустимая команда! Повторите попытку.
2	Некорректный ввод количества элементов стека	-1	Введено недопустимое значение! Повторите попытку
3	Некорректный ввод количества элементов стека	A	Введено недопустимое значение! Повторите попытку
4	Некорректный ввод элемента стека	B	Введено недопустимое значение! Повторите попытку
5	Некорректный ввод элемента стека	2.5	Введено недопустимое значение! Повторите попытку
6	Попытка создания стека при уже существующем	Ввод нового стека	Стек уже существует! Выход из программы...
7	Некорректный ввод лимитирующего адреса	asda	Введено недопустимое значение! Повторите попытку
8	Добавление элемента в полный стек	5 (стек полный)	Размер стека достиг максимального значения!
9	Извлечение элемента из пустого стека	Пустой стек	Стек пуст!
10	Корректный ввод элементов стека	5 1 2 3 4 5	Стек успешно заполнен.
11	Добавление элемента в стек при неполном стеке	4	Значение успешно помещено в стек.
12	Удаление элемента при непустом стеке	Удаление	Значение успешно извлечено: значение пика стека
13	Вывод массива освободившихся адресов	Вывод массива	Освободившиеся адреса: массив свободных адресов
14	Отсутствие решения	Нахождение подпоследовательности	Подпоследовательности не найдены.
15	Наличие решения	3 2 1 1 1	1 2 3 Время: время

Оценка эффективности

Измерения эффективности реализаций стека будут производиться в единицах измерения – тактах процессора. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

Время добавления элемента (в тактах процессора):

Количество элементов	Массив	Список
10	42060	122610
100	429900	1216500
1000	4482000	12278000

Время удаления элемента (в тактах процессора):

Количество элементов	Массив	Список
10	214800	845280
100	2029200	7548200
1000	20034000	79856000

Объём занимаемой памяти (в байтах):

Количество элементов	Массив	Список
10	160	160
100	1600	1600
1000	16000	16000

Контрольные вопросы

1. Что такое стек?

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

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

При хранении стека с помощью списка, то память всегда выделяется в куче. При хранении с помощью массива, память выделяется либо в куче, либо на стеке (в зависимости от того, динамический массив или статический). Для каждого элемента стека, реализованного списком, выделяется на 4 или 8 байт (на большинстве современных ПК) больше, чем для элемента массива. Эти дополнительные байты занимает указатель на следующий элемент списка. Размер указателя (4 или 8 байт) зависит от архитектуры. В данной лабораторной работе количество выделяемой памяти равно, так как при реализации стека массивом было введено еще одно поле, указывающее на конечную величину стека.

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?

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

5. 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 (в классическом случае). Вариант хранения списка может выигрывать только в том случае, если стек реализован статическим массивом. В этом случае, память для списка ограничена размером оперативной памяти (так как память выделяется в куче), а память для статического массива ограничена размером стека.

Вывод

Стек, реализованный связанным списком, внушительно (~400% от времени, достигнутом реализацией на основе массива) проигрывает по времени, но не проигрывает по памяти в данной реализации. Таким образом, можно сделать вывод, что если нужно реализовать такую структуру данных как стек, то лучше использовать массив, а не связанный список.