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)

Отчет

по лабораторной работе №4
по теме
«Работа со стеком»
Вариант 1.

Дисциплина: Типы и структуры данных

Студент ИУ7-31Б:
Косарев Алексей
Проверила:
Никульшина Т. А.

Москва, 2021

1. Описание условия задачи

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.

Реализовать:

- добавление элемента в стек;
- удаление элемента из стека;
- распечатать убывающие серии последовательности целых чисел в обратном порядке;

2. Описание ТЗ

- Описание исходных данных

Исходными данными являются числовые данные, введенные пользователем, для выбора пунктов меню.

Меню:

1. Добавление нового элемента в стек (массив)
2. Добавление нового элемента в стек (список)
3. Удаление элемента из стека (массив)
4. Удаление элемента из стека (список)
5. Вывод стека (в виде массива)
6. Вывод стека (в виде списка)
7. Вывод стека с адресами (в виде списка)
8. Вывод убывающих серий последовательностей в обратном порядке (стек массивом)
9. Вывод убывающих серий последовательностей в обратном порядке (стек списком)
10. Анализ обработки стеков по времени и по памяти
0. Выход из программы

В программе необходимо создать стек (путем добавления новых элементов).

- Описание выходных данных

- состояние стека
- адреса удаленных элементов стека
- убывающие последовательности стека в обратном порядке

- анализ работы с разными реализациями стека по времени и памяти

3. Описание задачи, реализуемой в программе

Программа реализует создание стека двумя способами (массивом и списком); добавление новых элементов в стек и удаление их оттуда; вывод текущего состояния стека. Также с помощью стека решается задача по выводу убывающих серий последовательностей в обратном порядке.

4. Способ обращения к программе

Обращение к программе происходит через консоль, путём запуска файла с расширением .exe (./main.exe).

5. Описание возможных аварийных ситуаций и ошибок пользователя

```
#define OK 0                        // Нет ошибок
#define STACK_OVERFLOW -1           // Переполнение стека
#define STACK_EMPTY -2              // Пустой стек
```

Данные ошибки не вызывают завершения программы, а лишь препятствуют выполнению некоторых пунктов меню и выдают соответствующие сообщения об ошибках.

Также при некорректном вводе (например, вводе буквы вместо числа при добавлении нового элемента в стек) программа будет запрашивать ввод у пользователя, пока тот не введет корректное значение.

6. Описание внутренних структур данных

```
#define MAX_STACK_LEN 500               // Максимальная длина стека

// Стек в виде массива
typedef struct
{
   int data[MAX_STACK_LEN];
   int *pointer;
   int *start;
   int *end;
} array_stack_t;

// Стек в виде списка
struct list_stack_t
{
   int number;
   struct list_stack_t *prev;
};
```

7. Алгоритмы

Алгоритм нахождения убывающих последовательностей в обратном порядке:

1. Начинаем движение с конца стека к началу
2. Смотрим, является ли предыдущий элемент стека больше, чем текущий. Если да, то выводим текущий элемент. Иначе смотрим, является ли текущий элемент концом возрастающей последовательности (если смотреть с конца). Если да, то выводим текущий элемент.
3. Удаляем текущий элемент.

Алгоритм добавления элемента в стек:

1. Считываем значение для нового элемента
2. Выделяем память под новый элемент
3. Добавляем значение нового элемента в выделенную память
4. В указатель предыдущего элемента добавляем указатель на новый

Алгоритм удаления элемента из стека:

1. Переносим указатель стека на предыдущий элемент
2. Освобождаем память из под удаляемого элемента

8. Тесты

Положительные тесты:

1. Ввод стека, в котором все элементы образуют убывающую последовательность.
2. Ввод стека, в котором элементы образуют две раздельные убывающие последовательности
3. Ввод стека, в котором элементы образуют несколько раздельных убывающих последовательностей
4. Удаление любого (не первого) элемента из стека
5. Удаление самого первого элемента из стека

Негативные тесты:

1. Некорректный выбор пункта меню (ввод буквы)
2. Некорректный выбор пункта меню (ввод несуществующего номера пункта)
3. Некорректный ввод значения элемента стека (ввод буквы)
4. Переполнение стека при вводе нового элемента
5. Попытка удаления элемента из пустого стека

9. Временная эффективность и затраты памяти

Анализ процесса:

N	time		memory	
	array	list	array	list
20	0	0	824	320
40	0	2	824	640
60	0	3	824	960
80	0	6	824	1280
100	1	7	824	1600
120	2	8	824	1920
140	2	11	824	2240
160	3	13	824	2560
180	4	17	824	2880
200	6	23	824	3200

Анализ добавления\удаления элементов из стека:

	list		array	
N	add	delete	add	delete
10	0	0	0	0
30	2	0	0	0
50	5	0	0	0
70	8	0	0	1
90	11	1	0	2
110	13	5	0	3
130	18	7	1	3
150	22	11	1	4
170	27	15	2	4
190	36	17	3	4

Из результатов анализа использования разных типов представления стека (в виде массива и в виде списка) видно, что по времени стек в виде массива быстрее, чем стек в виде односвязного списка.

По памяти выгодней использовать односвязный список, только если длина стека намного меньше длины массива (при проценте заполнения массива ~25%), иначе - выгодней массив.

10. Вывод

В процессе выполнения данной лабораторной работы я изучил два разных варианта реализации стека (с помощью массива и списка).

Проведя анализ обработки двух реализации стека, я понял, что по памяти стек выгодней реализовывать через односвязный список, если количество элементов в стеке намного меньше чем длина массива (при проценте заполнения массива ~25%), в иных случаях лучше пользоваться массивом.

По времени же массив абсолютно эффективнее для реализации стека, чем односвязный список.

Если в задаче заранее известно количество элементов стека, то лучше использовать массив для его реализации. Если же не понятно, сколько будет в стеке будет элементов, то рациональней использовать список.

11. 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: последним пришел – первым ушел, Last In – First Out (LIFO).

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

   Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу, которая выделяется в начале программы сразу под все элементы массива.

   Если стек реализован в виде статического односвязного линейного списка, то для его хранения отводится указатель на структуру, содержащую указатель на такую же структуру и само значение элемента int. При каждом добавлении элемента выделяется новая область памяти, адрес которой записывается в указатель стека.

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

   При реализации стека с помощью статического массива память выделяется при компиляции и не меняется во время работы программы. В конце работы программы освобождается память из под массива.

   При реализации стека с помощью линейного односвязного списка при удалении элемента сначала по указателю стека считывается информация об исключаемом элементе, а затем указатель смещается к предыдущему элементу. После чего освобождается память, выделенная под элемент.

4. Что происходит с элементами стека при его просмотре?

   При просмотре стека, чтобы обратиться ко всем элементам стека, надо удалить предыдущий элемент, так как принцип стека LIFO мы не можем обратиться к элементу стека из середины.

   Таким образом при просмотре стека все его элементы поочередно удаляются.

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

   При полностью заполненном массиве эффективнее реализовывать стек с помощью массива.

   Эффективность зависит от того, насколько заполнен массив. Если размер массива намного превышает количество введённых элементов стека, то эффективнее по памяти использовать односвязный список.