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 Дьяченко Артём Александрович

Группа ИУ7 – 33Б

Преподаватель Барышникова М. Ю.

2022 г.

Оглавление

Описание условия задачи

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.

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывода 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.

Используя стек, определить, является ли строка палиндромом.

Описание технического задания

Входные данные:

Числовое значение номера пункта меню, кол-ва удаляемых элементов на стеке и символы, добавляемые на стек.

Выходные данные:

Текущее состояние стека на массиве и списке, адреса освобождённых областей памяти, результат проверки слова на паллиндром.

Обращение к программе:

Запускается через терминал командой: ./app.exe.

Аварийные ситуации:

1. Ввод некорректного пункта меню.
2. Удаление элементов из пустого стека.
3. Кол-во добавляемых на стек элементов превышает максимальное кол-во элементов на стеке.
4. Ввод пустой строки в качестве паллиндрома.
5. Ввод некорректных целочисленных данных.

НАБОР ТЕСТОВ

№	Название теста	Пользовательский ввод	Вывод
1	Некорректный пункт меню	10	Введенный код не соответствует ни одному пункту меню. Попробуйте ещё раз.
2	Некорректный пункт меню	abacaba	Ошибка ввода кода действия. Попробуйте ещё раз.
3	Удаление элементов из пустого стека на массиве	2	Стек пуст!
4	Удаление элементов из пустого стека на списке	5	Стек пуст!
5	Переполнение стека на массиве или списке	1 или 4 12345678901	Переполнение стека!
6	Переполнение стека при проверки слова на паллиндром	8 12345678901	Переполнение стека!
7	Корректная проверка слова на паллиндром	8 123454321	Строка является палиндромом!

8	Отсутсвие освобождённых областей	7	Память не выделялась, или все освобожденные области были повторно использованы!
9	Вывод освобождённых областей памяти	7	Адреса освобожденных областей: 0x55abc2d861a0 0x55abc2d86180 0x55abc2d861c0

Описание структуры данных

Структура стека на массиве. Состоит из статического массива размера MAX_ARR_LEN (по умолчанию – 10) длинных чисел (для работы с расширенной кодировкой UTF-8, иначе русские символы не будут считываться) и размера элементов на стеке.

```
typedef struct
{
    wint_t array[MAX_ARR_LEN];  // массив "длинных" символов в стеке
    int length;                 // кол-во элементов в стеке
} arr_stack_t;
```

Структура стека на списке. Состоит из символа, индекса текущего элемента и указателя на предыдущий элемент. Корневой элемент указывает на «предка» NULL.

```
struct list_stack_t
{
    wint_t data;            // "длинный" символ
    int index;              // индекс элемента
    list_stack_t *prev;     // указатель на предыдущий элемент
};
```

Структура массива освобождённых областей стека. Состоит из массива размера MAX_AREAS_NUM (по умолчанию – 10) освобождённых областей и кол-ва элементов этого массива.

```
typedef struct
{
    list_stack_t *arr[MAX_AREAS_NUM];   // массив освобожденных областей
    int len;                            // кол-во элементов в массиве
} free_areas_t;
```

Оценка эффективности (такты)

Кол-во элементов	Операция (время в тактах)	Стек на массиве	Стек на списке
10	Добавление	7	11
	Удаление	7	7
	Палиндром	3	3
	Память (байт)	44	160
100	Добавление	148	174
	Удаление	57	90
	Палиндром	13	17
	Память (байт)	404	1600
1000	Добавление	384	633
	Удаление	474	494
	Палиндром	34	58
	Память (байт)	4004	16000

Описание алгоритма

1. После запуска программы пользователю предлагается ввести пункт меню.
2. Пользователь вводит целочисленные или символьные данные, в зависимости от пункта меню.
3. При добавлении элемента на стек, если он на массиве, то он записывается в первую свободную ячейку массива и увеличивает длину массива на 1. Если элемент добавляется в стек на списке – выделяется память под новый узел списка и он создаётся: в узел записывается значение символа а индекс инициализируется бОльшим на единицу значением, чем то, что находится в предыдущем узле (на который указывает ссылка).
4. При удалении элемента со стека на
   а) Массиве: его длина уменьшается на 1, но сам элемент никуда из ячейки памяти не уходит – он будет перезаписан следующим добавленным элементом.
   б) Списке: адрес удалённого элемента помещается в массив освобождённых адресов памяти, а узел списка стирается из памяти.
5. При проверке слова на палиндром его помещают в два стека на списке: в первом хранится первая половина слова, во втором – вторая. Центральный элемент (например, в слове abacaba, это будет c) затирается, и далее оба стека сравнивают два «первых на выходе» символа – если хотя бы одна пара не совпадает – слово не палиндром.

Ответы на контрольные вопросы

1. Что такое стек?

Стек – структура данных, работающая по принципу «последний пришёл – первый вышел». Чтобы добраться до элемента посередине стека, придётся «выкинуть» половину элементов.

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

При реализации стека на (статическом) массиве его размер зависит от установленного макс. кол-ва элементов. В моём случае стек на массиве занимает (MAX MAX_ARR_LEN + 4) байт. При реализации стека на списке его размер динамичен – для каждого элемента выделяется новая область памяти. В моём случае – по 16 байт на элемент.

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

При удалении элемента со стека на

а) Массиве: его длина уменьшается на 1, но сам элемент никуда из ячейки памяти не уходит – он будет перезаписан следующим добавленным элементом.

б) Списке: адрес удалённого элемента помещается в массив освобождённых адресов памяти, а узел списка стирается из памяти.

4. Что происходит с элементами стека при его просмотре?

Создаётся копия стека, куда помещаются все элементы. Элементы стека поочерёдно «вытаскиваются» из стека и выводятся на экран. Далее в стек возвращаются элементы в исходном порядке.

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

Разница подходов минимальна на малых данных. При данных, кол-во которых больше нескольких десятков, реализация стека на массиве выигрывает по всем параметрам, кроме удаления элементов (и только при больших данных больше 1000). Если Вы знаете, сколько максимум данных должно храниться в стеке – используйте массив. Если вам нужен динамический стек – используйте связный список.

Вывод

Реализация стека на массиве выигрывает в подавляющем количество случаев и по скорости, и по памяти. Поэтому, если максимальное кол-во записей известно заранее, стоит использовать именно эту реализацию. Для динамического стека подойдёт односвязный список. Но за это придётся жертвовать в первую очередь памятью, т.к. память выделяется под каждый узел. Так, при одинаковом кол-ве элементов, стек на массиве будет занимать в 4 раза меньше места.

Скорость добавления элементов в стек выше в 1.65 раза у стека на массиве, а удаления – на незначительные 5% при данных, больших тысячи. Проверка на палиндром у стека на массиве проходит быстрее в 1.7 раза. Это связано с тем, что для обработки стека на списке необходимо иметь дополнительную переменную для предыдущего узла.