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 Маслова Марина Дмитриевна
фамилия, имя, отчество

Группа ИУ7-33Б

2020 г.

Оглавление

Техническое задание

Условие задачи

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.

Входные данные

Целое число, задающее выбор пункта меню.

Целое число, задающие количество удаляемых элементов.

Символы, элементы стека.

Выходные данные

Текущее состояние стека на массиве и на списке, массив свободных областей, время работы подпрограмм операций, информация о том, является ли слово палиндромом.

Задачи, реализуемые программой

1. Добавление элементов в стек на массиве и списке.
2. Удаление элементов из стека на массиве и списке.
3. Вывод текущего состояния стека на экран.
4. Вывод массива свободных областей.
5. Определение того, является ли слово палиндромом или нет.

Возможные аварийные ситуации и ошибки пользователя

Некорректный ввод пункта меню.

Некорректный ввод целочисленных данных.

Переполнение стека.

Попытка вывода пустого стека или удаления из него элементов.

Описание внутренних структур данных

wint_t – тип для хранения длинных символов, объявленный в заголовочном файле *wchar.h* из стандартной библиотеки.

Для хранения стека на массиве используется структура **arr_stack_t**:

```
typedef struct
{
    wint_t array[MAX_ARR_LEN]; //массив элементов стека
    int length;                //кол-во элементов стека
} arr_stack_t;
```

Для хранения узлов стека на списке используется структура **list_stack_t**:

```
typedef struct list_stack_t list_stack_t;
struct list_stack_t
{
    wint_t data;         //символ
    int index;           //индекс символа
    list_stack_t *prev; //указатель на предыдущий элемент
};
```

Для хранения массива освобожденных областей используется структура **free_areas_t**:

```
typedef struct
{
     //массив освобожденных областей
    list_stack_t *arr[MAX_AREAS_NUM];
    int len;                          //длина
} free_areas_t;
```

Описание функций

```
/**
 * Удаляет элемент стека на массиве
 * stack – указатель на стек;
 * element – указатель, куда записывается удаленный элемент;
 * Возвращает код ошибки
 */
int as_pop(arr_stack_t *stack, wint_t *const element);
```

```
/**
 * Добавляет элемент в стек на массиве
 * stack – указатель на стек;
 * element – добавляемый элемент;
 * Возвращает код ошибки
 */
int as_push(arr_stack_t *stack, const wint_t element);

/**
 * Выводит содержимое стека
 * stack – указатель на стек;
 * Возвращает код ошибки
 */
int as_print(arr_stack_t *stack);

/**
 * Удаляет элемент стека на списке
 * stack – указатель на вершину стека;
 * element – указатель, куда записывается удаленный элемент;
 * Возвращает код ошибки
 */
int ls_pop(list_stack_t **stack, wint_t *const element);

/**
 * Добавляет элемент в стек на массиве
 * stack – указатель на вершину стека;
 * element – добавляемый элемент;
 * Возвращает код ошибки
 */
int ls_push(list_stack_t **stack, const wint_t element);

/**
 * Выводит содержимое стека
```

```
 * stack – указатель на вершину стека;
 * Возвращает код ошибки
 */
int ls_print(list_stack_t **stack);

/**
 * Проверяет, является ли строка палиндромом (стек на массиве)
 * word – указатель на стек с элементами строки;
 * Возвращает 0, если не является, и 1 в обратном случае
 */
bool a_is_palindrome(arr_stack_t *word);

/**
 * Проверяет, является ли строка палиндромом (стек на списке)
 * word – указатель на вершину стека с элементами строки;
 * Возвращает 0, если не является, и 1 в обратном случае
 */
bool l_is_palindrome(list_stack_t **word);

/**
 * Добавляет освобожденную область в список
 * array – указатель на массив освобожденных областей;
 * ptr – адрес добавляемой свободной области;
 * Возвращает код ошибки
 */
int add_area(free_areas_t *array, list_stack_t *const ptr);

/**
 * Удаляет область из списка освобожденных областей
 * array – указатель на массив освобожденных областей;
 * ptr – адрес удаляемой области;
 * Возвращает код ошибки
 */
int delete_area(free_areas_t *array, list_stack_t *const ptr);

/**
 * Выводит массив освобожденных областей на экран
```

```
 * array – указатель на массив освобожденных областей;
 * Возвращает код ошибки
 */
int print_free_areas(free_areas_t *array);
```

Описание алгоритма

При добавлении элемента в стек на массиве новый символ записывается в следующий свободный элемент массива, значение длины массива увеличивается на 1, при удалении – длина уменьшается на 1, при выводе происходит последовательное удаление символов из стека с их сохранением в другом стеке для последующего восстановления состояния стека на массиве.

При добавлении элемента в стек на списке выделяется память под новый элемент и адрес предыдущего, которому присваивается указатель на вершину, указатель на вершину в свою очередь переставляется на новый элемент, при удалении – очищается память, выделенная под верхний элемент, указатель вершины переставляется на предыдущий элемент. Вывод аналогичен выводу стека на массиве.

Для проверки строки на то, является она палиндромом или нет, она сохраняется в стек, далее её половина переставляется в другой стек, если длина строки нечетная «центральный» элемент удаляется без обработки. Далее из каждого стека поочередно достаются элементы и сравниваются, если хотя бы в одной паре элементы не равны, слово не палиндром, если во всех парах элементы равны, слово палиндром.

Тесты

№	Что проверяется	Входные данные	Результат
1	Некорректный ввод пункта меню	jk	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
2	Некорректный ввод пункта меню (длинная строка)	hjkl	Некорректный ввод кода действия. Попробуйте ещё раз.

3	Некорректный ввод пункта меню (< 0)	-1	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
4	Некорректный ввод пункта меню (> 8)	9	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
5	Неверный ввод количества удаляемых элементов (не числовые данные, пункты 2 и 5)	2 или 5 j	Введенные данные нецелочисленные!
6	Неверный ввод количества удаляемых элементов (неположительное число, пункты 2 и 5)	2 или 5 -7	Введенное количество выходит за допустимый диапазон!
7	Неверный ввод количества удаляемых элементов (количество больше количества элементов в стеке, пункты 2 и 5)	2 или 5 4 (при трех элементах в стеке)	Введенное количество выходит за допустимый диапазон!
8	Переполнение стека при добавлении элементов (пункты 1 и 4)	1 или 4 asdfgasdfgh	Элемент 'a' успешно добавлен! Элемент 's' успешно добавлен! Элемент 'd' успешно добавлен! Элемент 'f' успешно добавлен! Элемент 'g' успешно добавлен! Элемент 'a' успешно добавлен!

			Элемент 's' успешно добавлен! Элемент 'd' успешно добавлен! Элемент 'f' успешно добавлен! Элемент 'g' успешно добавлен! Переполнение стека!
9	Строка, длина которой больше размера стека (пункт 8)	8 фывапфывапр	Переполнение стека!
10	Отсутствие освобожденных областей	7 (сразу после запуска программы)	Память ещё не выделялась, или все освобожденные области были снова использованы!
11	Добавление элементов в стек	1 или 4 asd	Элемент 'a' успешно добавлен! Элемент 's' успешно добавлен! Элемент 'd' успешно добавлен!
12	Удаление элементов из стека	2 или 5 1 (при стеке из теста 11)	Элемент 'd' успешно удален!
13	Слово-палиндром	8 око	СТЕК НА МАССИВЕ Строка является палиндромом! СТЕК НА СПИСКЕ Строка является палиндромом!
14	Слово, не являющееся палиндром	8 окно	СТЕК НА МАССИВЕ Строка не является палиндромом! СТЕК НА СПИСКЕ Строка не является палиндромом!

Оценка эффективности

Добавление элементов (такты):

Количество элементов	Массив	Список
10	22	28
100	184	193
1000	749	910

Удаление элементов (такты):

Количество элементов	Массив	Список
10	20	24
100	205	212
1000	1007	1183

Палиндром (такты):

Количество элементов	Массив	Список
10	27	6
100	16	16
1000	26	64

Используемая память (байт):

Количество элементов	Массив	Список
10	4004	160
100	4004	1600
1000	4004	16000

Контрольные вопросы

Что такое стек?

Стек – это структура данных, в которой есть возможность обрабатывать только последний добавленный элемент; работает по принципу LIFO – последним пришел, первым вышел.

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

При реализации стека на статическом массиве память выделяется на стеке, размер зависит от выбранного программистом максимального количества элементов в стеке. При реализации стека на списке память выделяется отдельно под каждый элемент при его добавлении, размер памяти под конкретный элемент в моем случае составляет 16 байт.

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

При удалении элемента из стека, реализованном на статическом массиве, достаточно переместить «указатель» на предыдущий элемент. При удалении элемента из стека на списке память, отведенная под данный элемент, освобождается, а указатель на вершину стека перемещается на предыдущий элемент.

Что происходит с элементами стека при его просмотре?

Так в стеке есть возможность работать только с верхним элементом, при просмотре элементы исключаются из стека, поэтому необходимо создавать копию стека перед просмотром или возвращать его в первоначальное состояние для сохранения.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Ответ на этот вопрос представлен в выводе.

Вывод

На основе полученных значений времени обработки стеков на массиве и списке и используемой при этом памяти делаем вывод, что операции добавления и удаления элементов работают быстрее при реализации стека на массиве в 1.2 раза (в списке затрачивается время на выделение и освобождение памяти), однако при реализации стека на статическом массиве при количестве

элементов близких к 1000 и ниже стек на списке выигрывает по памяти в 2.5-25 раз в зависимости от количества элементов.

Интереснее дело обстоит с определением, является ли слово палиндромом, при небольших значениях стек на массиве проигрывает стеку на списке в 4.5 раза, при значениях близких к 100 они работают примерно одинаковое время, а при больших значениях стек на списке начинает проигрывать стеку на массиве (в 2.5 раза). Связано это с тем, что в алгоритме необходимо узнавать, пуст ли на данный момент стек или нет, при обработке на массиве происходит обращение к полю структуры, в то время как при обработке на списке достаточно проверить равенство указателя NULL, второе происходит быстрее, что и влияет на значения времени при малых значениях, при больших значениях данные обращения составляют малую долю от всех операций, поэтому не влияют на результат.

Использование массива освобожденных областей показало, что фрагментации памяти не происходит.