Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Отчет по дисциплине «Типы и структуры данных»
Лабораторная работа №4

Вариант 3

Студент Голикова С.М.
Группа ИУ7-35Б

Москва, 2021

1. Условие 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. 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.

2. Техническое задание

Используя стек, определить, является ли строка палиндромом.

3. Внешняя спецификация

а) Исходные данные и результаты

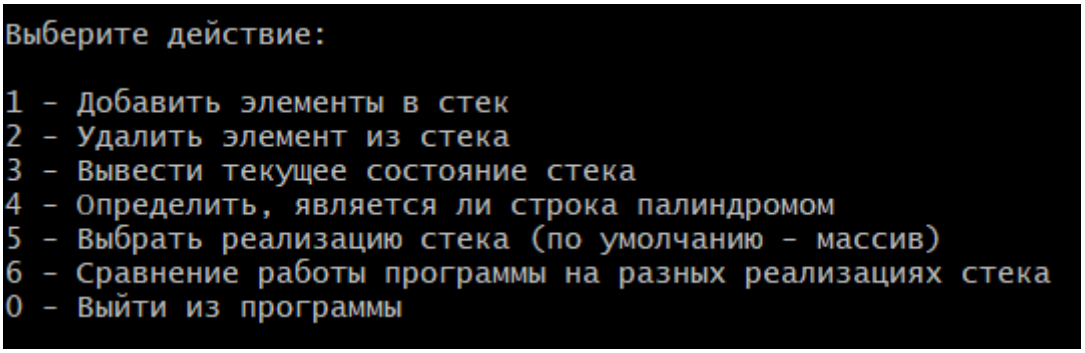
Входные данные:

- Целое число — номер пункта меню, который вызывает описанное в пункте действие
- Строка символов — последовательность элементов стека при его создании или добавлении в него

Допущения:

1) Максимальный размер стека - 100
2) Символ переноса строки - признак окончания ввода последовательности

Выходные данные:



В зависимости от пункта меню (см.выше):

- Символы - элементы стека
- Адреса - адреса текущих элементов стека или адреса свободных областей
- Ответ, является ли строка палиндромом
- Результаты анализа эффективности различных реализаций стека (время и объем памяти)

b) Задачи, реализуемые программой

Операции работы со стеком — добавление элемента, удаление элемента, вывод текущего состояния и адресов элементов, вывод свободных областей, проверка строки на палиндром при помощи стека.

c) Способ обращения к программе

Программа запускается из терминала в директории с проектом при помощи команды «./app.exe».

d) Возможные аварийные ситуации и ошибки пользователя

Аварийные ситуации и ошибки:

- некорректный выбор пункта меню
- переполнение стека
- удаление элемента из пустого стека
- вывод пустого стека
- проверка на палиндром пустого стека
- некорректный выбор пункта меню

В случае аварийной ситуации пользователю выдается сообщение об ошибке и происходит возвращение в меню.

4. Внутренние структуры данных

```
#define MAX_STACK_SIZE 100 // максимальный размер стека

// реализация стека в виде массива
typedef struct
{
    char data[MAX_STACK_SIZE];  // элементы стека
    int cur_size;               // текущий размер стека
} arr_stack_t;

// реализация стека в виде списка
typedef struct list_stack
{
    int value;                  // значение элемента
    int index;                  // индекс элемента
    struct list_stack *prev;    // указатель на предыдущий
                                // элемент стека
} list_stack_t;

// массив свободных областей
typedef struct
{
    size_t *arr;                // массив адресов
    int len;                    // текущее число адресов
} free_areas_t;
```

5. Описание алгоритма

Работа всей программы основана на взаимодействии пользователя с меню. Каждый пункт меню (от 0 до 6) вызывает описанное в пункте действие.

Алгоритм добавления элементов в стек реализуется при помощи операции push (вставки элемента) в соответствии с каждой структурой данных.

Алгоритмы удаления элемента и вывода текущего состояния стека и массива свободных областей реализуются при помощи операции pop (извлечения последнего элемента стека) в соответствии с каждой структурой данных.

Алгоритм проверки строки на палиндром реализован так: из стека удаляется size / 2 элементов и записывается в дополнительный стек. Затем извлекается один элемент, если size - нечетный. Далее элементы одновременно извлекаются из каждого стека и сравниваются: если все элементы совпали, строка является палиндромом, иначе - не является. Данный алгоритм работает с копией стека при реализации его в виде массива и с самим стеком (удаляя все элементы) при реализации его в виде списка.

6. Основные функции, используемые в программе

Имя	`int choose_action(void);`
Функция	Выбор действия в меню
Имя	`int array_input_element(arr_stack_t *stack);` `int list_input_element(list_stack_t **stack, free_areas_t *areas);`
Функция	Ввод элементов с добавлением их в стек (1 - массив, 2 - список)
Имя	`int array_push(arr_stack_t *stack, char element);` `int list_push(list_stack_t **head, char element, int first_node);`
Функция	Добавление одного элемента в стек (1 - массив, 2 - список)
Имя	`int array_delete_element(arr_stack_t *stack);` `int list_delete_element(list_stack_t **stack, free_areas_t *areas);`
Функция	Удаление последнего элемента из стека (1 - массив, 2 - список)
Имя	`int array_pop(arr_stack_t *stack, char *element);` `int list_pop(list_stack_t **stack, char *element, size_t *address);`
Функция	Извлечение последнего элемента из стека (1 - массив, 2 - список)
Имя	`void array_print(arr_stack_t *stack);` `void list_print(list_stack_t *stack, free_areas_t *areas);`
Функция	Вывод текущего состояния стека (1 - массив, 2 - список)
Имя	`int array_is_palindrome(arr_stack_t *word);` `int list_is_palindrome(list_stack_t **head, free_areas_t *areas);`
Функция	Проверка на палиндром (1 - массив, 2 - список)

Имя	`void create_areas(free_areas_t *areas);` `void print_free_areas(free_areas_t *areas);`
Функция	Работа с освобождаемыми областями памяти
Имя	`void free_list(list_stack_t **head);`
Функция	Освобождение списка

7. Тесты

№ теста	Ввод	Действие	Вывод
1	7	Ввод неверного пункта меню	Ошибка ввода: необходимо ввести номер одного из предложенных вариантов
2	1 строка более чем из 100 элементов	Ввод больше 100 элементов (в стек добавляются 100 элементов, для остальных выводится сообщение)	Элемент 'a' успешно добавлен … Элемент 'a' успешно добавлен Невозможно добавить элемент 'a': стек заполнен
3	1 a	Добавление в стек, в котором уже 100 элементов	Невозможно добавить элемент 'a': стек заполнен
4	2	Удаление элемента из пустого стека	Невозможно удалить элемент: стек пуст
5	3	Вывод пустого стека	Текущее состояние стека, реализованного в виде списка: Стек пуст
6	4	Проверка на палиндром пустого стека	Стек пуст
7	1 abc	Валидное добавление элементов в стек	Элемент 'a' успешно добавлен Элемент 'b' успешно добавлен Элемент 'c' успешно добавлен

8	2	Валидное удаление элемента	Элемент 'c' успешно удален
10	4 abcba	Проверка строки, которая является палиндромом, на палиндром	Строка является палиндромом
11	4 abc	Проверка строки, которая не является палиндромом, на палиндром	Строка не является палиндромом
12	3	Вывод стека в виде массива	Текущее состояние стека, реализованного в виде массива: -> c b a
13	3	Вывод стека в виде односвязного списка	Текущее состояние стека, реализованного в виде списка: -> c : 000002bd569580a0 b : 000002bd56958060 a : 000002bd569581c0 Массив освободившихся областей: Массив освободившихся областей пуст
14	3	Вывод стека в виде односвязного списка с наличием освободившихся областей	Текущее состояние стека, реализованного в виде списка: -> c : 000002705b658120 b : 000002705b6581e0 a : 000002705b6580c0 Массив освободившихся областей: 2705b658220

8. Анализ эффективности

Время добавления элемента в стек (такты):

Количество элементов	Массив	Список
10	519	8218
50	1465	28084
100	3657	49591

Время удаления элемента из стека (такты):

Количество элементов	Массив	Список
10	334	398
50	1592	1861
100	2833	2299

Время проверки на палиндром (такты):

Количество элементов	Массив	Список
10	1377	28662
50	3743	55375
100	5027	76599

Объем памяти (байты)

Количество элементов	Массив	Список
10	104	160
50	104	800
100	104	1600

9. Выводы

Стек, реализованный при помощи статического массива, в несколько раз эффективнее по времени, чем стек, реализованный при помощи односвязного списка, так как при работе с массивом нужно работать только с указателем, а в случае односвязного списка необходимо работать с указателем и еще освобождать память для последнего элемента.

На моих измерениях реализация стека при помощи массива оказалась эффективнее по памяти, однако такое возможно не всегда: при очень большом максимально допустимом размере стека или при других типах данных, хранящихся в стеке (например, double), список может выигрывать по памяти.

Минусом реализации стека при помощи массива в случае статического массива является ограничение памяти размером стека, в то время как при реализации односвязного списка память выделяется в куче и ограничена лишь размером оперативной памяти.

10. 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?**

При реализации стека при помощи связанного списка память выделяется по мере необходимости:
(sizeof(type) + sizeof(type_t*)) * count,
где count — число элементов, type — тип элементов, type_t — тип узла.

При реализации стека при помощи массива память выделяется в начале работы программы:
sizeof(type) * count,
где count — число элементов, type — тип элементов.

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

При реализации стека при помощи связанного списка: освобождается память, ранее занимаемая верхним элементом (по мере необходимости), и смещается указатель, который указывает на начало стека.

При реализации стека при помощи статического массива: смещается указатель, который указывает на вершину стека. Память освобождается при завершении работы программы.

4. **Что происходит с элементами стека при его просмотре?**

Элементы извлекаются из стека (уничтожаются).

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

Реализовывать стек при помощи списка эффективнее тем, что память для него ограничена размером оперативной памяти, в то время как для статического массива память ограничена размером стека. По времени работы реализация стека при помощи массива эффективнее.