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 Хамзина Регина Ренатовна
фамилия, имя, отчество

Студент, группа ИУ7-33Б Хамзина Р.Р.,

2020 г.

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.
Используя стек, определить, является ли строка палиндромом.

Техническое задание

Входные данные:

1. *Целое число* — номер пункта меню, который вызывает описанное в пункте действие.
2. *Строка символов* — последовательность элементов стека при его создании. Последовательность может состоять из любых символов кроме символов кириллицы и символа «.». Символ «.» - признак окончания ввода последовательности. Максимальное число элементов равно 100.
3. *Символ* — элемент стека при добавлении в него. Любой символ кроме кириллицы.

Выходные данные:

1. *Символы* — текущие элементы стека.
2. *Адреса* — адреса текущих элементов стека или адреса свободных областей.
3. *Временная характеристика* — время работы функций.
4. *Объемная характеристика* — объем структуры данных.

Функция программы:

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

Обращение к программе:

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

Аварийные ситуации

1. Некорректный ввод номера пункта меню
Входные данные : не целое число или целое число, большее 10, или целое число, меньшее 0.
Выходные данные : сообщение «Команда введена неверно».

2. Переполнение стека — при добавлении элемента в стек, он заполнен 100 элементами или при создании вводится последовательность более, чем из 100 элементов.
Входные данные : символ элемента или последовательность символов более, чем из 100 элементов.
Выходные данные : сообщение «Стек переполнен».

3. Удаление элемента из пустого стека
Входные данные : пустой стек.
Выходные данные : сообщение «Стек пуст».

4. Вывод пустого стека
Входные данные : пустой стек.
Выходные данные : сообщение «Стек пуст».

5. Проверка на палиндром пустого стека
Входные данные : пустой стек.
Выходные данные : сообщение «Стек пуст».

6. Ввод уже созданного стека
Входные данные : непустой стек.
Выходные данные : сообщение «Стек уже создан».

Внутренняя структура данных

Для реализации стека при помощи массива используется структура:

```
typedef struct
{
   int count;
   char array[MAX_CAPACITY];
} stack_array_t;
```

Её поля:
int count — текущее число элементов;
char array[MAX_CAPACITY] — стек, где *MAX_CAPACITY* — его максимальный размер, равный 100.

Для реализации стека при помощи односвязного списка используется структура:

```
typedef struct stack_list
{
    int count;
    char symbol;
    struct stack_list *next;
} stack_list_t;
```

Её поля:
int count — индекс элемента;
char symbol — символ элемента;
*struct stack_list *next* — указатель на следующий элемент;

Для массива свободных областей используется структура:

```
typedef struct
{
    size_t *array;
    int count;
} free_address_t;
```

Её поля:
*size_t *array* — массив адресов;
int count — текущее число адресов;

Алгоритм

Работа всей программы основана на работе пользователя с меню. Каждый пункт меню вызывает описанное в пункте действие. 0 — Выход из программы.
Алгоритм заполнения стека и добавления в него элементов реализуется при помощи операции *push* — вставка элемента - в соответствии с каждой структурой данных.
Алгоритм удаления элемента и вывода текущего состояния стека и массива свободных областей реализуется при помощи операции *pop* — получить из стека последний элемент — в соответствии с каждой структурой данных. Чтобы не повредить стек при печати, он копируется.
Алгоритм проверки строки на палиндром реализован так: из стека удаляется count / 2 элементов и записываются в дополнительный стек. Затем извлекаются count % 2 элементов. После одновременно извлекаются из каждого стека элементы и сравниваются. Если все элементы совпали — строка является

палиндромом, иначе — не является. Алгоритм также производится с копией стека.

Функции программы

Функции для работы со стеком, реализованном при помощи массива:

*int push_array(stack_array_t *stack, const char symbol)* — добавление элемента в стек;

*char pop_array(stack_array_t *stack)* — извлечение элемента из стека;

*int input_array_stack(stack_array_t *stack)* — ввод элементов*;*

*int print_array_stack(stack_array_t *stack)* — печать текущего состояния стека;

*int stack_array_is_palindrome(stack_array_t *stack, uint64_t *time)* — проверка строки на палиндром;

Функции для работы со стеком, реализованном при помощи односвязного списка, и с массивом свободных областей:

*int push_list(stack_list_t **head, const char symbol)* — добавление элемента в стек*;*

*char pop_list(stack_list_t **head, size_t *address)* — извлечение элемента из стека*;*

*int input_list_stack(stack_list_t **head)* — ввод элементов*;*

*int print_list_stack(stack_list_t **head, free_address_t array)* — печать текущего состояние стека*;*

*int stack_list_is_palindrome(stack_list_t **head, uint64_t *time)* — проверка строки на палиндром*;*

stack_list_t symbol_create(const char symbol)* — создание узла списка*;*

*void symbol_free(stack_list_t *node)* — освобождение узла списка*;*

*free_address_t *create_array(void)* — создание массива свободных областей*;*

void output_array(const free_address_t array) — печать массива свободных областей*;*

*int free_array(free_address_t *array)* — освобождение массива свободных областей;

Тесты

	Тест	Ввод	Вывод
1	Неверный пункт меню: больше 10	12	Команда введена неверно
2	Неверный пункт меню: меньше 0	-1	Команда введена неверно
3	Неверный пункт меню: не целое число	a	Команда введена неверно
4	Ввод больше 100 элементов	1/6, Строка, больше, чем из 100 символов	Стек переполнен
5	Добавление в стек, в котором уже 100 элементов	2/7, a	Стек переполнен
6	Удаление элемента из пустого стека	3/8	Стек пуст
7	Печать пустого стека	5/10	Стек пуст
8	Проверка строки на палиндром пустого стека	4/9	Стек пуст
9	Ввод уже непустого стека	1,6, aaa.	Стек уже создан
10	Валидный ввод стека	1,6, aaa.	Стек создан
11	Валидное добавление элемента в стек	2/7, h	Элемент добавлен
12	Валидное удаление элемента	3/8	Элемент удален
13	Проверка строки, которая является палиндромом, на палиндром	4/9, abcba.	Строка - палиндром
14	Проверка строки, которая не является палиндромом, на палиндром	4/9, abcab.	Строка не палиндром
15	Печать стека, как массив	5, в стеке: 4a)	)

			a 4
16	Печать стека, как односвязного списка, и массива свободных областей	10, в стеке: 4a)	Элемент :) Адрес: 55e4c0822160 Элемент : a Адрес: 55e4c22e5710 Элемент : 4 Адрес: 55e4c22e56f0 55e4c22e5730

Оценка эффективности

Время добавления элемента в стек (тики):

Число элементов	Массив	Список
10	2652	9282
100	14560	46592
100	130494	430630

Время удаления элемента из стека (тики):

Число элементов	Массив	Список
10	6500	8060
100	67600	132600
1000	754000	1118000

Время проверки строки на палиндром (тики):

Число элементов	Массив	Список
10	2756	6890
100	7020	18226
1000	46670	222872

Объем памяти (байты):

Число элементов	Массив	Список
10	1004	160
100	1004	1600
1000	1004	16000

Ответы на контрольные вопросы

Что такое стек?
Стек – это структура данных - последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины.

Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?
При реализации стека при помощи связанного списка: *(sizeof(type) + sizeof(*type_t))*count*, где *count* — число элементов, *type* — тип элементов, *type_t* — тип узла.
При реализации стека при помощи массива: *sizeof(type)*count*, где *count* — число элементов, *type* — тип элементов.

Каким образом освобождается память при удалении элемента стека при различной реализации стека?
При реализации стека при помощи связанного списка: освобождается память для верхнего элемента и смещается указатель, который указывает на начало стека.
При реализации стека при помощи динамического массива: смещается указатель, который указывает на вершину стека.

Что происходит с элементами стека при его просмотре?
Элементы стека извлекаются из стека — уничтожаются.

Каким образом эффективнее реализовывать стек? От чего это зависит?
Реализовывать стек при помощи списка эффективнее в том, что память для него выделяется в куче и ограничена размером оперативной памяти, в то время как для статического массива память ограничена размером стека.
По времени работы реализация стека при помощи массива эффективнее.

Выводы

Стек, реализованный при помощи статического массива, эффективнее по времени в 3-4 раза, чем стек, реализованный при помощи связанного списка, так как при работе с массивом нужно работать только с указателем, в случае односвязного списка необходимо работать с указателем и освобождать память для последнего элемента. Реализация стека при помощи массива эффективнее и по памяти, так как в случае односвязного списка хранится указатель на следующий узел списка. 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 размером оперативной памяти.