Министерство образования и науки 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 УЧЕБНОЙ ДИСЦИПЛИНЕ “ТИПЫ И СТРУКТУРЫ ДАННЫХ” ЛАБОРАТОРНАЯ РАБОТА №4

Студент Ляпина Наталья Викторовна
фамилия, имя, отчество

Группа ИУ7-32Б

Вариант 6

Студент _____ Ляпина Н.В.
подпись, дата *фамилия, и.о.*

Преподаватель _____ Силантьева А.В.
подпись, дата *фамилия, и.о.*

2021 г.

Оглавление

Задание

1. **Общее задание**
   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.

2. **Задание по варианту**

   Используя стек, определить, является ли строка палиндромом

3. **Входные данные**

- Команда для выбора действия (от 0 до 3)
    1. стек, реализованный с помощью массива
    2. стек, реализованный с помощью списка
    3. сравнение времени и памяти двух типов стека

- Количество добавляемых или удаляемых элементов стека
- Непосредственно элементы стека при добавлении

- **Требования к входным данным**

1) Нельзя добавлять больше элементов, чем допустимо его размером
2) Нельзя удалять элементов больше, чем есть в массиве
3) Количество добавляемых и удаляемых элементов должно быть натуральным числом
4) Стек может принимать числа только типа int

4. Выходные данные

- Стек, реализованный массивом
- Стек, реализованный списком
- Массив освобожденных адресов списка

5. Действие программы

Программа выполняет добавление, удаление и обработку элементов стека

5. Обращение к программе

Программа может быть вызвана через консоль компилятора с помощью команды “./app.exe”

6. Аварийные ситуации

В случае аварийной ситуации выводится сообщение об определенной ошибке.

Неверный ввод:

- Ошибка при вводе команды
- Ошибка при вводе кол-ва элементов для добавления
- Ошибка при вводе кол-ва элементов для удаления
- Ошибка при вводе элемента стека

Структура данных

Для реализации данной задачи была создана структура *stack_array_t*, состоящая из указателя на массив *arr* и размера *size*.

Стек из массива

```
typedef struct
{
        int *arr;
        int size;
} stack_array_t;
```

Также для реализации стека из списка используется тип *el_stack_list_t*, который представляет собой элемент списка. В качестве данных передается значение элемента *el* и его индекс в списке *i_el*. Ссылочная часть передает указатель на следующий элемент в списке *next*.

Стек из списка

```
typedef struct el_stack_list
{
        int  el;
        int i_el;
        struct el_stack_list *next;
} el_stack_list_t;
```

Для реализации массива освобожденных адресов используется структура *stack_freed_array_t*. Содержит в себе массив указателей на элементы списка *arr* и его размер *size*.

Массив освобожденных адресов

```
typedef struct el_stack_list
{
        el_stack_list_t **arr;
        int size;
} stack_freed_array_t;
```

Интерфейс модулей

Для обработки типа stack_array_t используются функции:

int count_add_elems(int *num, stack_array_t *stack)
// Функция подсчета количества добавляемых элементов
// num - количество добавляемых элементов, stack - стек

int read_elem(int *el)
// Функция чтение элемента стека с клавиатуры
// el - элемент

int push_array(stack_array_t *stack, int *value)
// Функция добавления одного элемента в стек
// value - значение элемента, stack - стек

int add_elems_array(int elems, stack_array_t *stack)
// Функция добавление некоторого количества элементов в стек
// elems - количество элементов, stack- стек

int add_random_elems_array(int elems, stack_array_t *stack)
// Функция добавления рандомных элементов в стек
// elems - количество элементов, stack - стек

int count_del_elems(int *num, stack_array_t *stack)
// Функция подсчета количества удаляемых элементов
// num - количество элементов, stack - стек

int pop_array(stack_array_t *stack)
// Функция удаления одного элемента
// stack - стек

int del_elems_array(int elems, stack_array_t *stack)
// Функция удаления некоторого количества элементов
// elems - количество элементов, stack - стек

int print_array_stack(stack_array_t *stack)
// Функция печати стека
// stack - стек

int check_palindrom(stack_array_t *stack)
// Функция проверки стека на палиндром
// stack - стек

int get_stack_array(void)
// Функция работы со стеком

Для обработки типа stack_list_t используются функции:

void free_stack(el_stack_list_t **elem_stack_list)
// Функция освобождения списка
// elem_stack_list - стек

int count_add_list_elems(int *num, el_stack_list_t *el_stack)
// Функция подсчета количества добавляемых элементов
// num - количество добавляемых элементов, el_stack - стек

int read_elem_list(int *el)
// Функция чтение элемента стека с клавиатуры
// el - элемент

int push_list(el_stack_list_t **el, int value)
// Функция добавления одного элемента в стек
// value - значение элемента, el - стек

int add_elems_list(el_stack_list_t **el, int n)
// Функция добавление некоторого количества элементов в стек
// n - количество элементов, el- стек

int add_elems_list_random(el_stack_list_t **el, int n)
// Функция добавления рандомных элементов в стек
// n - количество элементов, el - стек

int count_del_list_elems(int *num, el_stack_list_t *el_stack)
// Функция подсчета количества удаляемых элементов

```
// num - количество элементов, el_stack - стек

int get_arr_del_elems(stack_freed_array_t *arr, el_stack_list_t *el, int n)
// Функция создания массива освобожденных адресов
// arr - массив, el - стек, n - размер массива

int pop_list(el_stack_list_t **el)
// Функция удаления одного элемента
// el - стек

int del_elems_list(el_stack_list_t **el, int n)
// Функция удаления некоторого количества элементов
// n - количество элементов, el - стек

int print_list(el_stack_list_t **el)
// Функция печати стека
// el - стек

void print_list_pointer_array(stack_freed_array_t *arr)
// Функция печати массива освобожденных адресов
// arr - массив

int check_palindrom_list(el_stack_list_t **stack_cur)
// Функция проверки стека на палиндром
// stack_cur - стек

int get_stack_list(void)
// Функция работы со стеком
```

Описание алгоритма

1. Вводится команда для выполнения определенной функции программы.
2. Выполняется введенная функция
    - Добавить элементы в стек
    - Добавить случайные элементы в стек
    - Удалить элементы из стека
    - Вывести текущее состояние стека
    - Используя стек, определить является ли строка палиндромом
    - Выйти из программы.
3. При ошибке выполнения функции выводится сообщение об ошибке и программа завершается с ненулевым кодом возврата.

Тесты

Негативные тесты

№	Описание	Вводимая структура	Результат
1	Недопустимый символ команде меню	a2	Ошибка
2	Несуществующая команда меню	6	Ошибка
3	Недопустимое количество добавляемых элементов	-2	Ошибка
4	Недопустимый количество удаляемых элементов	-10	Ошибка
5	Недопустимое значение элемента стека	1.3	Ошибка

Позитивные тесты

№	Описание	Входные данные	Результат
1	Правильный пункт меню	comand = 1	Выбор действия
2	Добавление элемента в стек	size = 2 1 1	Успешное чтение стека
3	Добавление рандомных элементов в стек	size = 3 1 40 3	Успешное заполнение стека
4	Удаление элемента из стека	size = 2 40 3	Успешное удаление из стека
5	Проверка является ли стек палиндромом	size = 3 1 2 1	Стек является палиндромом
6	Проверка является	size = 4	Стек не является

	ли стек палиндромом	1 2 3 1	палиндромом

Оценка эффективности

Сравнение эффективности

Добавление

Кол-во элементов	Массив	Список
10	2474	9900
100	22174	126830
250	56870	255074
500	115926	480448
1000	229087	906722

Удаление

Кол-во элементов	Массив	Список
10	726	4565
100	5365	46761
250	12739	137013
500	25939	223365
1000	51443	426326

Обработка

Кол-во элементов	Массив	Список
10	1087	15000
100	8104	177435

250	18952	993083
500	36978	3473887
1000	81396	14582548

Память

Кол-во элементов	Массив	Список
10	20016	160
100	20016	1760
250	20016	4000
500	20016	9760
1000	20016	16000

Вывод

Если стек реализован статическим массивом, то добавление в него новых элементов будет происходить быстрее (~ в 4-6 раз), чем в стек, реализованный списком. Это связано с тем, что для хранения стека в виде списка требуется выделить память для указателей на следующие элементы списка.

Удаление элементов из массива происходит в 13-20 раз быстрее, чем из списка, так как при этом затрачивается время на очищение памяти, динамически выделенной под элемент стека.

При проверке стека на палиндром стек-массив справился быстрее (~ на 1000%) списка массива. Это также связано с тем, что тратится время на выделение памяти под стек-буфер.

Вариант хранения стека в виде списка может выигрывать в том случае, если стек реализован статическим массивом (массив должен быть заполнен менее, чем на 25%). Так же, если не известен размер стека, то в таком случае стоит использовать списки (или динамические массивы)

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: LIFO - последним пришел – первым ушел,

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

Если хранить стек как список, то память выделяется в куче. Если хранить как массив — либо в куче, либо на стеке (зависит от того, динамически или статический массив используется). Для каждого элемента стека, который хранится как список, выделяется на 4 или 8 байт (если брать современные ПК) больше, чем для элемента стека, который хранится как массив. Данные байты использованы для хранения указателя на следующий элемент списка. (из-за этого либо 4 либо 8 байт)

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

Если хранить стек как список, то верхний элемент удаляется при помощи операции освобождения памяти для него и смещением указателя, который указывает на начало стека. Если хранить стек как массив, то смещается только указатель на начало стека.

4. Что происходит с элементами стека при его просмотре?

Элементы стека удаляются, так как каждый раз достается верхний элемент стека, чтобы посмотреть следующий.

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

Стек эффективнее реализовать с помощью массива, так как он выигрывает в количестве занимаемой памяти (если массив динамический) и во времени обработки стека (добавлении и удалении элементов). Хранение с помощью списка может выигрывать, если только стек реализован с помощью статического массива, так как в данном случае размер памяти под список ограничен размером оперативной памяти (хранится в куче), а для статического массива — ограничена размером стека.