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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
СТЕК
ДИСЦИПЛИНА: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент, группа **Боренко А. ИУ7-32Б**

2020 г.

Цель работы

Реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного списка, оценить преимущества и недостатки каждой реализации, получить представление о механизмах выделения и освобождения памяти при работе с динамическими структурами данных.

Описание условия задачи

Вариант 8

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.

Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные.

Описание Технического Задания

Описание исходных данных и результатов

Исходные данные:

- Ввод пользователя (Стек 1, Стек 2)

Реультаты:

- В зависимости от выбора пользователя
  - Работа со стеком 1 (стека 2 нет) Вывод сохраненных адресов
  - Сортировка стеков 1, 2 — стек 3
  - Тестирование — сравнение реализаций сортировки стека по памяти и по времени

Описание задачи, реализуемой программой

Вывод меню, для реализации выбора пользователя. Ввод стека. Сортировка двух стеков с использованием третьего. Тестирование.

Способ обращения к программе

Обращение к программе происходит через терминал. В командную строку вводится «./app.exe».

Описание возможных аварийных ситуаций и ошибок пользователя

1. Ввод буквы вместо пункта меню — (Расценивается как выход)
2. Ввод буквы при вводе стека — (Расценивается как выход в главное меню)
3. Ввод цифры не из меню (тоже что и ввод буквы)

4. Ошибка при вводе стека (тоже что и ввод буквы)

Описание внутренних структур данных

```
typedef struct array_stack_t array_stack_t;
typedef struct adress_stack_node_t adress_stack_node_t;
typedef struct list_stack_node_t list_stack_node_t;

struct array_stack_t
{
	int *alb; //нижняя граница
	int *ps;  //вершина стека
	int *aub; //верхняя граница
};

struct list_stack_node_t
{
	int element;
	list_stack_node_t *next;
};

struct adress_stack_node_t
{
	long int element;
	adress_stack_node_t *next;
};
```

Описание алгоритма

Общий алгоритм

1. Вывести меню
2. Получить введенное число
3. Если это 1
    1. Вывести меню работы со стеком
    2. Получить число — выбор пользователя
    3. Если 1
        1. Добавить элемент
    4. Если 2
        1. Удалить
    5. Если 3
        1. Вывести стеки
    6. Если 4
        1. Вывести адреса
    7. Если 5 или ошибка
        1. Вывести адреса использованных ячеек памяти

        2. Освободить все списки
    4. Если это 2
        1. Ввести стек 1
        2. Ввести стек 2
        3. Вывести стек 1
        4. Вывести стек 2
        5. Создать стек 3
        6. Сортировать стеки — результат в стеке 3
        7. Вывести стек 1
        8. Вывести стек 2
        9. Вывести стек 3
    5. Если это 3
        1. Протестировать
        2. Вывести результаты
    6. Иначе
        1. Закончить работу

Алгоритм сортировки:

1. На входе стек1 и стек2
2. Пока хотя бы один из стеков не пуст
    1. Если стек 2 пуст → записать в него элемент из первого стека
    2. Пока стек 1 не пуст
        1. Увеличить счетчик
        2. Взять элемент из стека 1 сравнить с элементом из стека 2
        3. Сравнить с элементом стека 2
        4. Меньший оставить в стеке 2
        5. Другой записать в стек 3
    3. Пока счетчик не опустеет
        1. Уменьшить счетчик
        2. Переместить элемент из стека 3 в стек 1
    4. Если стек 1 пуст → записать в него элемент из первого стека
    5. Пока стек 2 не пуст

1. Увеличить счетчик
2. Взять элемент из стека 2 сравнить с элементом из стека 1
3. Сравнить с элементом стека 1
4. Меньший оставить в стеке 1
5. Другой записать в стек 3

6. Пока счетчик не опустеет
    1. Уменьшить счетчик
    2. Переместить элемент из стека 3 в стек 2

3. Закончить сортировку

Основные функции

adress_work.h
- `int add_adresses(adress_stack_node_t **head, long int adress);`
- `void output_adresses(adress_stack_node_t *head);`
- `void free_adresses(adress_stack_node_t **head);`

help.h
- `void list_pop(list_stack_node_t **head);`
- `int list_push(list_stack_node_t **head, int new_el);`
- `void list_free(list_stack_node_t **stack);`
- `void list_output_stack(list_stack_node_t *stack);`
- `void list_output_tree_stacks(list_stack_node_t **stack1, list_stack_node_t **stack2, list_stack_node_t **stack3);`
- `int array_create_stack(array_stack_t *stack, int len);`
- `int array_push(array_stack_t *stack, int element);`
- `int array_pop(array_stack_t *stack);`
- `void array_free(array_stack_t *stack);`
- `void array_output_stack(array_stack_t stack);`
- `void array_output_tree_stacks(array_stack_t *stack1, array_stack_t *stack2, array_stack_t *stack3);`
- `void clearInputBuf(void);`
- `void output_line(int size);`
-

input.h
- `int input_array_stack(array_stack_t *stack);`
- `int input_list_stack(list_stack_node_t **head);`
- `int input_type_stack();`

menu.h
- `void menu();`

sort.h
- `int list_sort(list_stack_node_t **stack1, list_stack_node_t **stack2, list_stack_node_t **stack3);`
- `int array_sort(array_stack_t *stack1, array_stack_t *stack2, array_stack_t *stack3);`
- `void choise_sort();`

stack_work.h

- `void choise_stack_work();`

`testing.h`

- `void testing();`

Тесты

Главное Меню

- *Негативные (Результат для всех — завершение программы)*
  - *Ввод буквы*
  - *Ввод числа не из приведенного списка*
- *Позитивные*
  - *Ввод 1 — действие — работа со стеком*
  - *Ввод 2 — действие — сортировка стеков*
  - *Ввод 3 — действие — Тестирование*

Меню Работа со стеком

- *Негативные (Результат для всех — выход в главное меню)*
  - *Ввод буквы*
  - *Ввод числа не из приведенного списка*
- *Позитивные*
  - *Ввод 1 — действие — Добавление элемента*
  - *Ввод 2 — действие — Удаление элемента*
  - *Ввод 3 — действие — Вывод стека*
  - *Ввод 4 — действие — Вывод списка адресов*
  - *Ввод 5 — действие — Освобождение стека, вывод списка адресов и егое освобождение*

Сортировка

- *Позитивные*
  - *Тест 1 (Отсортированные — надо просто склеить)*
    - *Стек 1: 0, 1, 2*
    - *Стек 2: 3, 4, 5*
    - *Результат:*
      - *Стеки 1 и 2 пусты,*
      - *Стек 3: 0, 1, 2, 3, 4, 5*
  - *Тест 2 (Отсортированные одинаковые)*

        - *Стек 1: 1, 2, 3*
        - *Стек 2: 1, 2, 3*
        - *Результат:*
            - *Стеки 1 и 2 пусты,*
            - *Стек 3: 1, 1, 2, 2, 3, 3*
    - *Тест 3 (Перевернутый тест 1)*
        - *Стек 1: 7, 6, 5*
        - *Стек 2: 4, 3, 2*
        - *Результат:*
            - *Стеки 1 и 2 пусты,*
            - *Стек 3: 2, 3, 4, 5, 6, 7*
    - *Тест 4 (Перевернутый тест 2)*
        - *Стек 1: 7, 6, 5*
        - *Стек 2: 7, 6, 5*
        - *Результат:*
            - *Стеки 1 и 2 пусты,*
            - *Стек 3: 5, 6, 7*
    - *Тест 5 (Разной длины Случайные с отрицательными числами)*
        - *Стек 1: -1, 2, -3, 4, 2*
        - *Стек 2: -2, 2, 3, 10*
        - *Результат:*
            - *Стеки 1 и 2 пусты,*
            - *Стек 3: -3, -2, -1, 2, 2, 2, 3, 10*

Выводы о проделанной работе

Результаты тестирования

Таблица с результатами тестирования (кол-во повторов для вычисления времени = 100)

N	type	size	procent	time
1	array	10	10	664
1	list	10	10	2282
2	array	10	20	625
2	list	10	20	8478
3	array	10	30	653
3	list	10	30	20347
4	array	10	50	780

4	list	10	50	52591
5	array	10	70	766
5	list	10	70	100607
6	array	10	100	876
6	list	10	100	210225
7	array	20	10	604
7	list	20	10	2318
8	array	20	20	2217
8	list	20	20	9181
9	array	20	30	2361
9	list	20	30	18670
10	array	20	50	2897
10	list	20	50	53821
11	array	20	70	2338
11	list	20	70	107713
12	array	20	100	2455
12	list	20	100	207023
13	array	30	10	661
13	list	30	10	2242
14	array	30	20	2208
14	list	30	20	9217
15	array	30	30	4717
15	list	30	30	19550
16	array	30	50	4844
16	list	30	50	52453
17	array	30	70	5124
17	list	30	70	102074
18	array	30	100	4968
18	list	30	100	200334
19	array	50	10	597
19	list	50	10	2234
20	array	50	20	2195
20	list	50	20	8962
21	array	50	30	5161
21	list	50	30	18747
22	array	50	50	12528
22	list	50	50	50343
23	array	50	70	12785
23	list	50	70	97451
24	array	50	100	13635
24	list	50	100	212631
25	array	70	10	791
25	list	70	10	2273
26	array	70	20	2228
26	list	70	20	12372
27	array	70	30	5512
27	list	70	30	18848
28	array	70	50	15794
28	list	70	50	56586
29	array	70	70	24729
29	list	70	70	104659
30	array	70	100	25187
30	list	70	100	204418

```
|  31 |    array |      100 |       10 |      662 |
|  31 |     list |      100 |       10 |     2515 |
-------------------------------------------------
|  32 |    array |      100 |       20 |     2213 |
|  32 |     list |      100 |       20 |     9578 |
-------------------------------------------------
|  33 |    array |      100 |       30 |     4863 |
|  33 |     list |      100 |       30 |    18580 |
-------------------------------------------------
|  34 |    array |      100 |       50 |    12772 |
|  34 |     list |      100 |       50 |    51539 |
-------------------------------------------------
|  35 |    array |      100 |       70 |    24868 |
|  35 |     list |      100 |       70 |    97928 |
-------------------------------------------------
|  36 |    array |      100 |      100 |    50114 |
|  36 |     list |      100 |      100 |   198082 |
-------------------------------------------------
```

Очевидно, что прireализации стека массивом, можно значительно выйграть в скорости, по сравнению со списком. Однако проведем расчеты памяти:
(10 100) где 10 элементов максимальная необходимая возможность стека, 10 элементов заполнено.
Указатель 8 байт, int — 4 байта.
Список : (4 + 8) * 10 = 120
Массив : 4*10 = 40
Массив выгоднее по памяти.
(10 50) где 10 элементов максимальная необходимая возможность стека, 5 элементов заполнено.
Указатель 8 байт, int — 4 байта.
Список : (4 + 8) * 5 = 60
Массив : 4*10 = 40
Массив выгоднее по памяти.
(10 30) где 10 элементов максимальная необходимая возможность стека, 3 элементов заполнено.
Указатель 8 байт, int — 4 байта.
Список : (4 + 8) * 3 = 36
Массив : 4*10 = 40
Список выгоднее по памяти.
(10 10) где 10 элементов максимальная необходимая возможность стека, 1 элемент заполнен.
Указатель 8 байт, int — 4 байта.
Список : (4 + 8)
Массив : 4*10
Список выгоднее по памяти.
Можно сделать вывод, что по памяти выгоден массив, только если количество элементов хотя бы приблизительно известно. Иначе в стеке останется очень много выделенной, но не использованной памяти.

Вывод

Стек выгодно реализовывать массивом, когда есть ограничения по времени и хотя бы приблизительно известен размер стека, В других случаях лучше использовать список, потому что он не накладывает ограниченй на длину стека, что удобней и пользователю и программисту.

Ответы на вопросы

1. Что такое стек?

Стек — это абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (последний пришел, первый вышел).

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

При хранении стека массивом — запрашивается макс. кол-во элементов в стеке, и под это кол-во выделяется память.

При хранении стека списком — память выделяется под каждый элемент по мере добавления.

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

При хранении стека массивом — память заполняется нулем, передвигается указатель (фактически ничего не освобождается)

При хранении стека списком — память освобождается при удалении каждого элемента.

4. Что происходит с элементами стека при его просмотре?

Они все вынимаются, потом складываются назад.

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

Реализацию стека надо выбирать по ситуации.

Если стек точно будет маленький, и необходимо, чтобы программа была эффективвной по времени и памяти, то надо реализовывать массивом.

Если размер стека заранее не известен (даже примерно). Требований особых к эффективности не стоит, то выгодней использовать реализацию списком.