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» (ИУ)

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«РАБОТА СО СТЕКОМ»

по курсу:

«ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Вариант: 8

Студент:

Авдейкина Валерия Павловна, группа ИУ7-33Б _____

(подпись, дата)

Руководители:

Преподаватель ИУ7

Силантьева Александра Васильевна _____

(подпись, дата)

Преподаватель ИУ7

Барышникова Марина Юрьевна _____

(подпись, дата)

Проверяющий:

_____

(подпись, дата)

Оценка: _____

2022 г.

Оглавление

Описание условий 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.

Вариантное задание (8): Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные.

Техническое задание

1 Входные данные

- Входные данные при вводе номера выбранной опции меню:
  - Целые беззнаковые числа [0; 14]
- Входные данные для различных опций меню:

Номер опции	Название опции	Входные данные
0	Выход	-
1	Заполнить стеки из файла	Имя файла1, Имя файла2
2	Показать содержимое стеков	-
3	Добавить элемент...	Номер выбора, значение элемента
4	Удалить элемент...	Номер выбора
5	Выполнить сортировку с помощью 3-го стека	-
6	Выполнить сортировку с анализом	-
7	Очистить все	-
8	Вывести меню	-
9	Вывести правила ввода	-
10 -15	Работа с тестовыми стеками	Различные

Таблица 1: Входные данные опций меню

- Имя файла: символы, не более 255
- Номер выбора: целое неотрицательное число [0;1]
- Значение элемента: целое

2 Выходные данные

- Выходные данные при вводе номера выбранной опции меню:
  - Выходные данные команды в случае корректного ввода
- Выходные данные для различных опций меню:

Номер опции	Название опции	Выходные данные
0	Выход	Сообщение об успехе / Сообщение об ошибке
1	Заполнить стеки из файла	Сообщение об успехе / Сообщение об ошибке
2	Показать содержимое стеков	Данные / Сообщение об ошибке
3	Добавить элемент...	Сообщение об успехе / Сообщение об ошибке
4	Удалить элемент...	Сообщение об успехе / Сообщение об ошибке
5	Выполнить сортировку с помощью 3-го стека	Данные / Сообщение об ошибке
6	Выполнить сортировку с анализом	Данные / Сообщение об ошибке
7	Очистить все	Сообщение об успехе / Сообщение об ошибке
8	Вывести меню	Меню
9	Вывести правила ввода	Правила ввода
10 -15	Работа с тестовыми стеками	Различные

Таблица 2: Выходные данные опций меню

3 Задача, реализуемая программой

Программа реализует обработку стеков с различными реализациями, включая удаление, добавление и сортировку элементов.

4 Способ обращения к программе

Программа вызывается в командной строке без каких-либо аргументов. Ввод данных производится с клавиатуры после соответствующего приглашения к вводу («`Команда...`» / «`Введите..`»).

5 Возможные аварийные ситуации и ошибки со стороны пользователя

1. Некорректный ввод имени файла:
    1.1. Пустой ввод
    1.2. Недопустимые символы
    1.3. Превышение допустимого количества символов *(поведение программы не определено)*

1.4. Имя несуществующего файла или директории

2. Некорректные данные в обрабатываемом файле / вводе с клавиатуры

3. Некорректный ввод номера опции:

3.1. Пустой ввод

3.2. Недопустимые символы (*поведение программы не определено*)

3.3. Недопустимый номер

Описание внутренних структур данных

В ходе работы были составлены следующие структуры (АТД):

1. Стек на основе массива:
   - data — массив элементов стека
   - size — размер массива
   - top — позиция вершины стека

```
struct arr_stack
{
   int *data;
   size_t size;
   int top;
};
```

Итоговый размер структуры: 20 байт.

2. Стек на основе линейного односвязного списка:
   - size — количество элементов стека
   - max_size — максимально допустимый размер стека
   - top — указатель на вершину стека — элемент структурного типа stack_elem:
     - next — указатель на следующий элемент стека
     - value — значение элемента стека

```
typedef struct stack_elem *stack_elem_t;

struct stack_elem
{
   stack_elem_t next;
   int value;
};

struct stack
{
   stack_elem_t top;
   size_t size;
   size_t max_size;
};
```

Итоговый размер структуры:

Алгоритм

Для объединения и сортировки двух стеков с помощью третьего стека используется следующий алгоритм:

- два стека последовательно объединяются в один
- пока объединенный стек не пуст:
  - считываем вершину объединенного стека
  - пока считанная ранее вершина больше, чем вершина объединенного стека, и ее индекс не равен -1
    - добавляем в объединенный стек вершину отсортированного
  - когда вершина отсортированного стека стала меньше, чем считанная ранее, добавляем считанну вершину в отсортированный стек

Для работы со стеком были реализованы следующие функции:

ДЛЯ РАБОТЫ СО СТЕКОМ НА ОСНОВЕ МАССИВА:
`size_t get_arr_size(arr_stack_t stack)`
`int get_arr_top(arr_stack_t stack)`
`int get_arr_top_value(arr_stack_t stack)`
`int arr_pop(arr_stack_t stack)`
`void arr_push(arr_stack_t stack, const int value)`
`arr_stack_t create_arr_stack(const size_t size)`
`void delete_arr_stack(arr_stack_t *stack)`
ДЛЯ РАБОТЫ СО СТЕКОМ НА ОСНОВЕ ЛИНЕЙНОГО ОДНОСВЯЗНОГО СПИСКА:
`int get_top_value(stack_t stack)`
`void free_elem(stack_elem_t elem)`
`stack_elem_t create_elem(const int value)`
`int pop(stack_t stack)`
`void push(stack_t stack, const int value)`
`stack_t create_stack(const size_t max_size)`
`size_t get_size(stack_t stack)`
`size_t get_max_size(stack_t stack)`
`void delete_stack(stack_t *stack)`

Таблица 3. Описание основных функций

Тестирование

- **Команда 1**

```
Команда (8 - меню, 9 - правила): 1
Заполнение 1-го стека...
Введите имя файла: ./data/rnd_9.txt
Стек был успешно заполнен

Заполнение 2-го стека...
Введите имя файла: ./data/rnd_16.txt
Стек был успешно заполнен
```

- **Команда 2**

```
Команда (8 - меню, 9 - правила): 2
        СТЕК 1:
Максимальный размер: 5000
Количество элементов: 16
Содержимое:
<- [ 10 -5 -3 2 1 2 9 798 6 4 1 3 7 54 2 3 ]

        СТЕК 2:
Максимальный размер: 5000
Количество элементов: 9
Содержимое:
<- [ 13 2 0 67 1 44 -3 32 -1 ]
```

- **Команда 3**

```
Команда (8 - меню, 9 - правила): 3
1) в 1-й стек
2) в 2-й стек
Введите номер варианта: 1
Вставка в 1-й стек...
Введите целое число: 10
Элемент был успешно добавлен
```

- **Команда 4**

```
Команда (8 - меню, 9 - правила): 4
1) из 1-го стека
2) из 2-го стека
Введите номер варианта: 1
Удаление из 1-го стека...
Элемент 0 был успешно удален
```

- **Команда 5**

```
Команда (8 - меню, 9 - правила): 5
Выполняется сортировка стеков...
Слияние прошло успешно
Выполняется сортировка стека...
Сортировка прошла успешно, 1-й стек перезаписан
        РЕЗУЛЬТАТ СОРТИРОВКИ
Максимальный размер: 5000
Количество элементов: 25
Содержимое:
<- [ 798 67 54 44 32 13 9 7 6 4 3 3 2 2 2 2 1 1 1 0 0 -1 -3 -3 -5 ]
```

- **Команда 6**

```
                Время (мкс)     Размер (Б)
        ./data/500.txt
linked list:             4331            8016
vector     :               16            2012
        ./data/1000.txt
linked list:            17157           16016
vector     :               33            4012
        ./data/2000.txt
linked list:            69827           32016
vector     :               91            8012
        ./data/500_1.txt
linked list:             8491            8016
vector     :               19            2012
        ./data/500__1.txt
linked list:               17            8016
vector     :                3            2012
```

- **Команда 7**

```
Команда (8 - меню, 9 - правила): 7
Очистка 1-го стека...
Ошибка: Структура(-ы) пуста(-ы)
Очистка 2-го стека...
Ошибка: Структура(-ы) пуста(-ы)
```

Оценка эффективности

Измерение времени проводилось в двух плоскостях — сравнивались две разные реализации при сортировке данных различного размера. Количество итераций: 20.

```
                Время (мкс)    Размер (Б)
        ./data/500.txt
linked list:            4345           8016
vector     :              16           2012
        ./data/1000.txt
linked list:           17208          16016
vector     :              31           4012
        ./data/2000.txt
linked list:           70136          32016
vector     :              87           8012
        ./data/500_1.txt
linked list:            8550           8016
vector     :              16           2012
        ./data/500__1.txt
linked list:              16           8016
vector     :               3           2012
```

Из данных, полученных в ходе обработки данных, можно сделать следующие выводы:

- При сортировке небольших данных сортировка при реализации стека на основе списка является более эффективной по времени
- При сортировке больших данных с помощью стека на основе списка тратится больше времени, чем при помощи стека на основе массива — это происходит из-за того, что в списке происходит динамическое выделение памяти, которое занимает много времени
- При увеличении размера данных сортировка при обеих реализациях становится дольше
- Под списочный стек выделяется больший объем памяти
- Списочный стек очень выгоден при сортировке данных, отсортированных в обратном порядке

Контрольные вопросы

1 Что такое стек?

Структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека.

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

Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу.

В случае реализации линейным односвязным списком: до начала работы указатель стека показывает на нулевой, физически отсутствующий адрес (т. е. указатель - пустой). При включении элемента в стек сначала происходит выделение области памяти, адрес которой записывается в указатель стека, а затем по значению этого указателя в стек помещается информация.

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

При реализации на основе вектора память при удалении стека не освобождается — она освобождается тогда, когда стек полностью опустошается или опустошается некоторый блок.

При реализации на основе линейного односвязного списка память освобождается из-под каждого элемента стека при его удалении.

4 Что происходит с элементами стека при его просмотре?

При просмотре стека его элементы удаляются.

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

Эффективность реализации стека зависит от отсортированности данных, их количества, планируемой деятельности и ограничения по памяти. При больших ограничениях выгоднее использовать списочный стек.

Выводы

В ходе работы были изучены детали обработки стека при его различных реализациях и способ сортировки стека с помощью второго.