

## Задача А. Стек

Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 Мб

Реализуйте структуру данных "стек". Напишите программу, содержащую описание стека и моделирующую работу стека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

`push n` Добавить в стек число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`pop` Удалить из стека последний элемент. Программа должна вывести его значение.

`back` Программа должна вывести значение последнего элемента, не удаляя его из стека.

`size` Программа должна вывести количество элементов в стеке.

`clear` Программа должна очистить стек и вывести `ok`.

`exit` Программа должна вывести `bye` и завершить работу.

### Формат входных данных

Во входном файле дана последовательность команд по одной на строке.

Гарантируется, что набор входных команд удовлетворяет следующим требованиям: максимальное количество элементов в стеке в любой момент не превосходит 100, все команды `pop` и `back` корректны, то есть при их исполнении в стеке содержится хотя бы один элемент.

### Формат выходных данных

Выведите результат работы программы.

### Примеры

ВВОД	ВЫВОД
<code>push 2</code>	<code>ok</code>
<code>push 3</code>	<code>ok</code>
<code>push 5</code>	<code>ok</code>
<code>back</code>	<code>5</code>
<code>size</code>	<code>3</code>
<code>pop</code>	<code>5</code>
<code>size</code>	<code>2</code>
<code>push 7</code>	<code>ok</code>
<code>pop</code>	<code>7</code>
<code>clear</code>	<code>ok</code>
<code>size</code>	<code>0</code>
<code>exit</code>	<code>bye</code>

## Задача В. Стек с защитой от ошибок

Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 Мб

Реализуйте структуру данных "стек". Напишите программу, содержащую описание стека и моделирующую работу стека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

`push n` Добавить в стек число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`pop` Удалить из стека последний элемент. Программа должна вывести его значение.

`back` Программа должна вывести значение последнего элемента, не удаляя его из стека.

`size` Программа должна вывести количество элементов в стеке.

`clear` Программа должна очистить стек и вывести `ok`.

`exit` Программа должна вывести `bye` и завершить работу.

Перед исполнением операций `back` и `pop` программа должна проверять, содержится ли в стеке хотя бы один элемент.

### Формат входных данных

Во входном файле дана последовательность команд по одной на строке.

### Формат выходных данных

Выведите результат работы программы. Если во входных данных встречается операция `back` или `pop`, и при этом стек пуст, то программа должна вместо числового значения вывести строку `error`.

### Примеры

ВВОД	ВЫВОД
<code>push 2</code>	<code>ok</code>
<code>back</code>	<code>2</code>
<code>pop</code>	<code>2</code>
<code>size</code>	<code>0</code>
<code>pop</code>	<code>error</code>
<code>push 1</code>	<code>ok</code>
<code>size</code>	<code>1</code>
<code>exit</code>	<code>bye</code>

## Задача С. Очередь с защитой от ошибок

Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 Мб

Реализуйте структуру данных "очередь". Напишите программу, содержащую описание очереди и моделирующую работу очереди, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

**push** *n* Добавить в очередь число *n* (значение *n* задается после команды). Программа должна вывести **ok**.

**pop** Удалить из очереди первый элемент. Программа должна вывести его значение.

**front** Программа должна вывести значение первого элемента, не удаляя его из очереди.

**size** Программа должна вывести количество элементов в очереди.

**clear** Программа должна очистить очередь и вывести **ok**.

**exit** Программа должна вывести **bye** и завершить работу.

Перед исполнением операций **front** и **pop** программа должна проверять, содержится ли в очереди хотя бы один элемент.

### Формат входных данных

Во входном файле дана последовательность команд по одной на строке.

### Формат выходных данных

Выведите результат работы программы. Если во входных данных встречается операция **front** или **pop**, и при этом очередь пуста, то программа должна вместо числового значения вывести строку **error**.

### Примеры

ВВОД	ВЫВОД
push 5	ok
push 7	ok
size	2
pop	5
size	1
front	7
pop	7
size	0
front	error
exit	bye

## Задача D. Дек

Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 Мб

Реализуйте структуру данных "дек". Напишите программу, содержащую описание дека и моделирующую работу дека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

`push_front n` Добавить в начало дека число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`push_back n` Добавить в конец дека число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`pop_front` Удалить из стека первый элемент. Программа должна вывести его значение.

`pop_back` Удалить из стека последний элемент. Программа должна вывести его значение.

`front` Программа должна вывести значение первого элемента, не удаляя его из дека.

`back` Программа должна вывести значение последнего элемента, не удаляя его из дека.

`size` Программа должна вывести количество элементов в деке.

`clear` Программа должна очистить дек и вывести `ok`.

`exit` Программа должна вывести `bye` и завершить работу.

### Формат входных данных

Во входном файле дана последовательность команд по одной на строке. Гарантируется, что количество элементов в деке в любой момент не превосходит 100. Все операции `pop_front`, `pop_back`, `front`, `back` всегда корректны.

### Формат выходных данных

Выведите результат работы программы.

### Примеры

ВВОД	ВЫВОД
<code>size</code>	<code>0</code>
<code>push_back 1</code>	<code>ok</code>
<code>size</code>	<code>1</code>
<code>push_back 2</code>	<code>ok</code>
<code>size</code>	<code>2</code>
<code>push_front 3</code>	<code>ok</code>
<code>size</code>	<code>3</code>
<code>exit</code>	<code>bye</code>

## Задача E. Дек с защитой от ошибок

Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 Мб

Реализуйте структуру данных "дек". Напишите программу, содержащую описание дека и моделирующую работу дека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

`push_front n` Добавить в начало дека число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`push_back n` Добавить в конец дека число  $n$  (значение  $n$  задается после команды). Программа должна вывести `ok`.

`pop_front` Удалить из стека первый элемент. Программа должна вывести его значение.

`pop_back` Удалить из стека последний элемент. Программа должна вывести его значение.

`front` Программа должна вывести значение первого элемента, не удаляя его из дека.

`back` Программа должна вывести значение последнего элемента, не удаляя его из дека.

`size` Программа должна вывести количество элементов в деке.

`clear` Программа должна очистить дек и вывести `ok`.

`exit` Программа должна вывести `bye` и завершить работу.

### Формат входных данных

Во входном файле дана последовательность команд по одной на строке. Гарантируется, что количество элементов в деке в любой момент не превосходит 100.

### Формат выходных данных

Выведите результат работы программы. Перед исполнением операций `pop_front`, `pop_back`, `front`, `back` программа должна проверять, содержится ли в деке хотя бы один элемент. Если во входных данных встречается операция `pop_front`, `pop_back`, `front`, `back`, и при этом дек пуст, то программа должна вместо числового значения вывести строку `error`.

### Примеры

ВВОД	ВЫВОД
<code>size</code>	<code>0</code>
<code>pop_front</code>	<code>error</code>
<code>push_back 1</code>	<code>ok</code>
<code>size</code>	<code>1</code>
<code>push_back 2</code>	<code>ok</code>
<code>size</code>	<code>2</code>
<code>push_front 3</code>	<code>ok</code>
<code>size</code>	<code>3</code>
<code>exit</code>	<code>bye</code>