Задача A. 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

stack.in	stack.out
push 2 push 3 push 5 back size pop size push 7 pop clear size exit	ok ok ok 5 3 5 2 ok 7 ok 0 bye

Задача B. 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

stack2.in	stack2.out
push 2 back pop size pop push 1 size exit	ok 2 2 0 error ok 1 bye

Задача C. 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

queue2.in	queue2.out
push 5 push 7 size pop size front pop size front exit	ok ok 2 5 1 7 7 0 error 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

deq.in	deq.out
size push_back 1 size push_back 2 size push_front 3 size exit	0 ok 1 ok 2 ok 3 bye

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

deq2.in	deq2.out
size pop_front push_back 1 size push_back 2 size push_front 3 size exit	0 error ok 1 ok 2 ok 3 bye