Модуль 4. Структуры данных.

4. Решение задач для самостоятельной работы

4.1 Задача №56. Стек неограниченного размера

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

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.

Размер стека должен быть ограничен только размером доступной оперативной памяти. Перед исполнением операций back и pop программа должна проверять, содержится ли в стеке хотя бы один элемент. Если во входных данных встречается операция back или pop, и при этом стек пуст, то программа должна вместо числового значения вывести строку error.

Входные данные

Команды управления стеком вводятся в описанном ранее формате по 1 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.

Выходные данные

Требуется вывести протокол работы со стеком, по 1 сообщению в строке

Пример

Входные данные	Выходные данные
push 3 push 14 size clear push 1 back push 2 back pop size pop size exit	ok ok 2 ok ok 1 ok 2 2 1 1 0 bye

Решение

Для реализации стека неограниченного размера можно воспользоваться разобранным в теоретической части примером, изменив структуру данных таким образом, чтобы она поддерживала произвольное увеличение размера.

Для этого можно воспользоваться вектором или списками, причём во втором случае будет достаточно однонаправленного списка.

Также в программу понадобится добавить обработку ошибочной ситуации – доступ к элементам стека при отсутствии элементов.

Программа на языке C++ представлена ниже.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main () {
     unsigned int last=0;
     int k;
     vector <int> stack;
     string s;

     for (;;) {
          cin >> s;
          if (s == "push") {
```

```
            cin >> k;
            stack.push_back(k);
            last++;
            cout << "ok" << endl;
        }
        else if (s == "pop") {
            if (last > 0) {
                last--;
                cout << stack[last] << endl;
                stack.pop_back();
            }
            else
                cout << "error" << endl;
        }
        else if (s == "back") {
            if (last > 0) {
                cout << stack[last-1] << endl;
            }
            else
                cout << "error" << endl;
        }
        else if (s == "size") {
            cout << last << endl;
        }
        else if (s == "clear") {
            last=0;
            stack.clear();
            cout << "ok" << endl;
        }
        else if (s == "exit") {
            cout << "bye" << endl;
            return 0;
        }
    }
    return 0;
}
```

4.2 №59. Очередь неограниченного размера

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

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 последний 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.

Размер очереди должен быть ограничен только размером доступной оперативной памяти. Перед исполнением операций front и pop программа должна проверять, содержится ли в очереди хотя бы один элемент. Если во входных данных встречается операция front или pop, и при этом очередь пуста, то программа должна вместо числового значения вывести строку error.

Входные данные

Команды управления очередью вводятся в описанном ранее формате по 1 на строке.

Выходные данные

Требуется вывести протокол работы с очередью, по 1 сообщению в строке

Пример

Входные данные	Выходные данные
push 1 front exit	ok 1 bye
size push 1 size push 2 size push 3 size exit	0 ok 1 ok 2 ok 3 bye

Решение

Также как и в предыдущей задаче нужно воспользоваться структурами данных с возможностью неограниченного роста: очередью или вектором.

Также обработаем ситуацию выбора элемента при пустой очереди, хотя эта операция в данной работе не требуется.

При реализации с помощью списков можно воспользоваться решением предыдущей задачи, зафиксировав в ней указатели на начало и конец очереди.

При решении с помощью вектора необходимо дополнить решение операциями перемещения очереди, которая возникнет если в начале очереди освободится много ячеек, а выделенный размер буфера подойдёт к концу.

Эта операция может произойти при добавлении очередного элемента, причём можно передвинуть очередь сразу, а можно отложить её выполнение до заполнения всей очереди.

Также в программе реализован свой счётчик числа элементов очереди, хотя в стандартном векторе он реализован. Использование своего счётчика потребовало слежения за их синхронным изменением.

Программа на языке C++, реализующая второй способ, представлена ниже.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main () {
     unsigned int first=0, last=0;
     int k;
     vector <int> queue;
     string s;
     for (;;) {
          cin >> s;
          if (s == "push") {
               cin >> k;
               int i;
               if (first > 0) { // если очередь можно сдвинуть в
начало
                    for (i=0; first+i<last; i++)
                         queue [i]=queue[first+i];
                    first = 0;
                    last = i;
                    // обрезать массив
                    while (queue.size() > last)
                         queue.pop_back();
               }
               queue.push_back(k);
               last++;
               cout << "ok" << endl;
          }
          else if (s == "pop") {
```

```
                if (first < last)
                    cout << queue[first++] << endl;
                else
                    cout << "error" << endl;
            }
            else if (s == "front") {
                if (first < last)
                    cout << queue[first] << endl;
                else
                    cout << "error" << endl;
            }
            else if (s == "size") {
                cout << last-first << endl;
            }
            else if (s == "clear") {
                first = 0;
                last=0;
                queue.clear();
                cout << "ok" << endl;
            }
            else if (s == "exit") {
                cout << "bye" << endl;
                return 0;
            }
        }
        return 0;
}
```

4.3 №50. Игра в пьяницу

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

В игре в пьяницу карточная колода раздается поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт – проигрывает.

Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту ("шестерка берет туза").

Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

Напишите программу, которая моделирует игру в пьяницу и определяет, кто выигрывает. В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9.

Входные данные

Программа получает на вход две строки: первая строка содержит 5 чисел, разделенных пробелами — номера карт первого игрока, вторая – аналогично 5 карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой.

Выходные данные

Программа должна определить, кто выигрывает при данной раздаче, и вывести слово first или second, после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении 10^6 ходов игра не заканчивается, программа должна вывести слово botva.

Примеры

Входные данные

1 3 5 7 9

2 4 6 8 0

Выходные данные

second 5

Решение

В данной задаче необходимо реализовать две очереди. Для этого можно воспользоваться своими наработками из предыдущей задачи или, что заметно проще и полезнее ☺, использовать стандартные реализации.

Программа должна последовательно моделировать ситуацию взятия двух карт с вершины каждой колоды, затем проводить сравнение, как описано в условии, и возвращать обе карты в одну из двух очередей с конца в нужном порядке.

Первая карта выиграет если её значение равно 0, а у второй карты 9 или если её значение больше, чем у второй карты, но при этом её значение не равно 9, при значении второй карты, равное 0. Такие сложные условия лучше ограничивать дополнительными скобками.

В программе после проверки какая карта выигрывает фиксируется указатель на соответствующую очередь.

Программа на языке C++ представлена ниже.

```
#include <iostream>
#include <deque>
using namespace std;
int main () {
	deque <int> f, s, *t;
	int i, k, m;
	for (i=0; i<5; i++) {
		cin >> k;
		f.push_back(k);
	}
	for (i=0; i<5; i++) {
		cin >> k;
		s.push_back(k);
	}
	for (i=0; i<1000000; i++) {
		if (f.empty() || s.empty())
			break;
		k=f.front();
		f.pop_front();
		m=s.front();
		s.pop_front();
		if (((k == 0) && (m == 9)) ||
			((k > m) && !((k == 9) && (m == 0))))
			t=&f;
		else
			t=&s;
		(*t).push_back(k);
		(*t).push_back(m);
	}
	if (s.empty())
		cout << "first " << i;
	else if (f.empty())
		cout << "second " << i;
	else
		cout << "botva";
	return 0;
}
```