**ПРАКТИЧЕСКОЕ ЗАДАНИЕ: РАБОТА С КОЛЬЦЕВЫМИ ОЧЕРЕДЯМИ**

**Очередь** – это линейный список, доступ к элементам которого происходит по принципу FIFO (First In - First Out – первым пришел и первым ушел). Элемент, который был первым поставлен в очередь, будет первым получен при поиске. Элемент, поставленный в очередь вторым, при поиске будет получен также вторым и т.д. Этот способ является единственным при постановке элементов в очередь и при поиске элементов в очереди.

Для очереди характерны две операции – занесение элемента в очередь и извлечение (считывание) элемента из очереди. В простой очереди для работы с данными доступны две позиции – начало (из этой позиции происходит извлечение) и конец (в эту позицию заносится входящий элемент) или "голова" и "хвост". Произвольный доступ к элементам, в отличии от массивов, формально не допускается. Операция извлечения (считывания) формально является разрушающей. Это означает, что считанные данные становятся недоступными. Возможно, явного разрушения (уничтожения) данных и не происходит, но к ним нет доступа, используя стандартные операции работы с очередью.

**Классификация очередей**

* По архитектуре очереди делятся на **линейные** и **кольцевые** (циклические).
* По количеству позиций записи и считывания – на **простые** и **приоритетные**.

Абстрактная структура данных, очередь реализовывается несколькими способами:

* одномерные массивы;
* связные списки;
* классы объектно-ориентированного программирования.

Если реализовывать очередь по аналогии с реальной, где её участники (элементы) продвигаются к кассе, то извлечение элемента будет иметь сложность O(n), (где n – количество элементов в очереди), т.к. элементы-очередники будут приближаться к кассе. При программной реализации эффективнее постоянно смещать указатель на начало очереди (смещать кассу). При этом возникает другая проблема: количество элементов, одновременно находящихся в очереди следует ограничить. Для этого очередь нужно закольцевать: последний (*новый*, поступивший последним) элемент будет указывать на первый (самый *старый*, поступивший раньше всех), а использовать будем два указателя: на элемент, который следует извлекать, и элемент, который был помещен.

Кольцевая очередь становится заполненной только в том случае, когда указатель свободного места совпадает с указателем выборки следующего элемента (иначе, очередь будет иметь свободное место для нового элемента). Однако, это значит, что в начале программы индекс выборки должен устанавливаться не в нулевое значение, а на значение максимального числа событий. В противном случае первое обращение к процедуре постановки в очередь приведет к появлению сообщения о заполнении списка. Следует помнить, что очередь может содержать только на один элемент меньше, чем значение максимального числа событий, поскольку указатели выборки и постановки в очередь всегда должны отличаться хотя бы на единицу (иначе, нельзя будет понять заполнена ли очередь или она пустая).

**Пример реализации кольцевой очереди**

Наиболее широко кольцевые очереди применяются в ОС при буферизации данных (Lazy write, Read ahead).

Рассмотрим пример использования очередей в организации ввода с клавиатуры в режиме реального времени. Так работают многие текстовые процессоры, когда изменяется формат параграфа или выравнивается строка, Имеется короткий промежуток времени, когда набранная на клавиатуре информация не выводится на экран до окончания другого процесса. Для достижения такого эффекта в программе должна предусматриваться постоянная проверка ввода с клавиатуры в ходе выполнения другого процесса. При вводе некоторого символа его значение должно быстро ставиться в очередь и процесс должен продолжаться. После завершения процесса набранные символы выбираются из очереди и обрабатываются обычным образом.

Для того, чтобы понять, как это можно сделать с помощью циклической очереди, рассмотрим следующую простую программу, состоящую из двух процессов:

Первый процесс запускает счетчик 1 ÷ 32000.

Второй процесс помещает вводимые с клавиатуры символы в циклическую очередь. При этом вводимые символы не будут выводиться на экран до тех пор, пока не будет введена точка с запятой.

Вводимые символы не будут выводиться на экран, поскольку первый процесс будет иметь более высокий приоритет до тех пор, пока вы не введете точку с запятой или пока счетчик не достигнет максимального значения. Затем из очереди будут выбраны введенные символы и выведены на экран.

{ программа, иллюстрирующая применение циклической очереди }

program KeyBuffer;

uses Crt, Dos;

const

MAX\_EVENT = 10;

type

EvtType = char;

var

event: array[1..MAX\_EVENT] of EvtType;

spos, rpos, t: integer;

ch: char;

procedure Qstore(q:EvtType); { поместить объект в очередь }

begin

{ убедиться, что в очереди имеется свободное место}

if ((spos+1=rpos) or ((spos=MAX\_EVENT) AND (rpos=0))then

[WriteLn](http://valera.asf.ru/delphi/help/name.php?name=Writeln)('List full')

else

begin

event[spos]:=q; spos:=spos+1;

if spos=MAX\_EVENT then spos:=1; { вернуться в начало очереди }

end;

end; { конец процедуры постановки в очередь }

function Qretrieve:EvtType; { выборка объекта из очереди }

begin

if rpos=MAX\_EVENT then rpos:=1; { вернуться в начало очереди }

else rpos:=rpos+1;

if rpos=spos then

begin

[WriteLn](http://valera.asf.ru/delphi/help/name.php?name=Writeln)('Queue empty');

Qretrieve:=';';

end else

begin

Qretrieve:= event[rpos-1];

end;

end; { конец функции выборки объекта из очереди }

begin { буфер набранных с помощью клавиатуры символов }

spos := 0;

rpos := MAX\_EVENT;

{ установить ch на начальное значение, отличное от точки с запятой }

ch:=' ';

t := 1;

repeat

if KeyPressed then

begin

ch := ReadKey;

Qstore(ch);

end;

t:=t+1;

[write](http://valera.asf.ru/delphi/help/name.php?name=write)(t); write(' ');

until (t=32000) or (ch=';');

{ вывести содержимое буфера введенных с клавиатуры символов }

repeat

ch:=Qretrieve;

if ch<>';' then [Write](http://valera.asf.ru/delphi/help/name.php?name=write)(ch);

until ch = ';';

end.

**Задания**

1. Даны два кольцевых списка, содержащих фамилии студентов и номера экзаменационных билетов. Число пересчета для билетов - Е, для учащихся - К. Сформировать список студентов и номера билетов.
2. Даны два кольцевых списка, содержащие фамилии студентов двух групп. Перевести L студентов из первой группы во вторую. Число пересчета - К. Вывести списки групп.
3. Дан кольцевой список, содержащий 20 фамилий футболистов. Разбить игроков на две команды по 10 человек; во вторую команду попадает каждый N-й человек.
4. Даны два кольцевых списка, содержащие фамилии спортсменов двух команд по айкидо. Произвести жеребьевку. В первой команде выбирается каждый N-й игрок, а во второй - каждый M-й.
5. Даны два кольцевых списка, содержащие фамилии баскетболистов основного состава команды и запасного. Произвести К замен, число для пересчета - T . Вывести списки составов.
6. Даны два кольцевых списка, содержащие фамилии курсантов двух взводов. Во время учений М человек из первого взвода получили травмы, организовать его пополнение курсантами второго взвода.
7. Даны два кольцевых списка, содержащие фамилии участников лотереи и наименования призов. Выиграет N человек (каждый К-й); число для пересчета призов - T.
8. Даны два списка, содержащие перечень товаров и фамилии покупателей. Каждый N-й покупатель покупает М-й товар. Вывести список покупок.
9. Даны два списка, содержащие перечень товаров, производимых APPLE и SAMSUNG. Сформировать список конкурирующих товаров (выпускаемых как одной так и другой фирмой).