Очередь — структура данных, которая, как и стек, имеет ограничения, по добавлению и удалению элементов. Чтобы понять очереди, представьте  очередь в  магазин: человек в начале очереди (тот, кто пришел первый) будет обслуживаться первым, вновь пришедшие люди будут добавляться в конец очереди. Таким образом, первый человек в очереди обслуживается первым, последний в очереди, обслуживается последним.Сокращенно очереди обозначаются как: FIFO — First In, First Out (первым пришел, первым ушел).

Очереди часто используются в программировании сетей, операционных систем и других ситуациях, в которых различные процессы должны разделять ресурсы, такие как процессорное время.

Немного терминологии:

* **enqueue** — добавление элемента в очередь;
* **dequeue** — удаления элемента из очереди.

Хотя концепция очереди может быть простой, в программировании очередь не так проста, как стек. Давайте вернемся к примеру с очередью в магазин. Скажем, один человек покидает очередь. Тогда что? Все в очереди должны пройти вперед, верно? Теперь, представьте себе, что только один человек может двигаться одновременно. Таким образом, второй человек делает шаг вперед, чтобы заполнить место, оставленное от первого лица, а затем третьим лицом делается шаг вперед, чтобы заполнить место, оставленное после второго человека, и так далее. Теперь представьте, что никто не может уйти или быть добавленным в очередь, пока все не шагнул вперед. Как вы уже поняли, очередь будет двигаться очень медленно. Конечно, не трудно  запрограммировать очередь, и она будет работать, но очень не просто сделать очередь, которая бы работала очень быстро!

Есть несколько основных способов реализации очереди. Во-первых, просто создать массив и переложить все элементы, чтобы поместить элемент в очередь или извлечь его. Это тот случай, о котором мы говорили, он является очень медленным.

Медлительность предыдущего метода заключается в том, что чем больше элементов в очереди, тем больше времени занимает перемещение. Есть еще один метод, не смещая элементы в очереди, выполняются функции постановки и удаления элементов из очереди. Представьте себе, снова, очередь в магазин. Представим, что не очередь движется к продавцу, а продавец движется к очереди, таким образом позиция начала очереди постоянно изменяется, стремится к концу очереди. Таким образом люди, которые стоят в очереди не делают шаг вперед или назад, что экономит время.

К сожалению, этот метод является гораздо более сложной задачей, чем очередь из массива. Вместо того, чтобы отслеживать только  «конец» очереди, мы также должны следить за началом очереди. Все это еще более усложняется, когда мы понимаем, что после того, как в очередь добавляется  и извлекается элемент, очередь будет нужно обернуть вокруг конца массива. Подумайте об очереди в магазин. Пока люди входят и выходят, очередь движется все дальше и дальше назад, и в конце концов очередь окружит магазин и в конечном итоге вернется на прежнее место.

