How to Implement a Queue in C

[Jacob Sorber](https://www.youtube.com/channel/UCwd5VFu4KoJNjkWJZMFJGHQ)

<https://www.youtube.com/watch?v=FcIubL92gaI>

|  |  |
| --- | --- |
| so what is a queue and how do you implement it in C that's our topic for today  so I recently did a video about stacks  one of the first data structures that you learn anytime you start a real study of data structures and today I want to talk about its cousin the queue so quick recap stacks were basically just a list of elements where you can only access them at one end the top you can push stuff on to the stack you can pop stuff off of the stack if this isn't making any sense you should go watch that video  and then come back it may make today's video make more sense but queues are also basically a list there are linear data structure but instead of just accessing at one end we access at both ends but in a very controlled way we have two ends a head and a tail we add new elements to the tail and we remove things from the head so this is a data structure that basically works like a queue in a grocery store or really any scenario where you're waiting in line basically you're adding things to the back of the list and you're taking things off the front of the list and things get serviced in a first-come first-serve order just like with stacks we have two operations instead of push and pop we have n Q and D Q and Q is where we actually add something to the queue DQ is when we remove something from the queue and you're gonna find that we really use cues really everywhere in software anytime you have stuff arriving this can be work this can be network connections whatever anytime you have something that is arriving that you want to process in the order it  arrived you want a queue and like a stack we can implement a queue with an array or with a linked list today I'm a little short on time so I'm going to focus on the linked list implementation and we'll pick up the array base queue in a future video and of course if you've never seen linked lists before and that's a little confusing check out my linked list videos just to catch up again link in the description also as always source code is available through patreon check the description for more information on how to get access but now let's jump into the code if you saw the snacks video this code should look familiar to begin with I just took that code replaced stack with Q and removed the code from push and pop so now we  have this empty NQ and DQ we have our two operations and we have main that calls those functions but which of course won't work right now because our code doesn't do anything yet and like before we're working with intz just because they're simple and we could easily use doubles float structs pointers anything we want really but insert quick and easy so that's what we're going to use today now the first thing I'm going to change is the definition of what we want to call a queue here with a stack everything happens at one end all pushing and popping happens at that one end at the top so it was enough to just store a single pointer and call that our stack but with my queue I want to keep track of two pointers I need to keep track of the head and the tail which will put together in another struct and we're going to call that struct a queue this  is basically just all the data that we're going to need to represent our queue and I'm gonna follow the advice that I recommended to myself in my stacks video and make an initialization function that sets up our queue in an object-oriented language this would be in your constructor but here it's just a function that's going to take a pointer to our struct that's so we can modify it  and then we're going to set both the head and the tail to null meaning that our queue starts out empty and saving some typing let's rename these my queues to just queue I like that a bit better now in NQ we're just going to create a new node by calling malloc we set its value to the value we want to add to the cube and we set its next pointer to the goal that's because this is going at the end of the list so it's not going to have anything after it now we need to connect it to our list so we'll check to see if we have a tail on this list if we do we just connect it to the next pointer to the new node we just created then we set the tail to be the new node so it is now the new tail now one last thing if the list was empty and we just added the first element then head is going to still be null so let's just check to see if that's the case and if it is we can just set the head to point to that new node as well so just to recap we created a new node if there's a tail we connect the old up to this new tail and then we make sure that the head still make sense and that's really it of course we still need to return true if we were successful and let's come back up and return false if Malik failed basically just meaning that we couldn't get any more memory because that's really this functions only failure case unless of course someone passes in a bad pointer now let's look at D Q now anytime you're writing a function it's useful to think about its error cases what can I check now to see if this operation that was requested is even possible now with D Q the main check we want to do up front is the empty case if the queue is empty there's nothing we can do I can't return an integer from an empty queue so we're going to first check to see if the queue is empty and if it is we just returned after that we're going to save a pointer to the head and grab the integer value in the head then we just move the current head to point to the node after it this is the same thing we did with our stack and then of course if we run out of nodes we need to make sure to let the tail know and set it to null as well we definitely don't want to leave our tail dangling there and then of course we return our saved result okay so to recap we check for an empty queue just to make sure that we can do what we're being asked to do and then we save the head of the queue oh wait I forgot to free my memory that's not good that would have been a memory leak if I had forgotten to do that so now we save  the result and you're going to return and then we finally remove it from the actual list and that looks pretty good let's make sure it works down here in main we need to clean up some stuff mostly just the initialization now let's call our init Q function and these queues are still called s1 s2 and s3 I told you I just copied the stack code over as a starter so that's going to bug me so let's change these to make them q1 q2 and q3 and we initialize the last two Q's okay so now let's see if it works so into the terminal we go we compile it okay nice I'm actually pleasantly surprised I didn't mess anything up there and we run it and it works it's just like our stack example except that the order is reversed so we're getting values out in first in first out or FIFO order rather than in last in first out order or LIFO order which is what we got from our stack so hopefully at this  point you can see that queues aren't really complicated they're really not and they're used everywhere as I mentioned you can build a queue with an array as well as a linked list it's not complicated except we do have to be a little bit careful about our array indexes and how they wrap around this is usually using the modulus operator but we'll talk about that in a future video also I know a lot of you have been waiting for more information about the upcoming course that I'm offering in July I'm happy to be able to tell you today that the course page is now up at the following URL you can now sign up for the course space is of course limited because my time is limited and this is going to be a hybrid video and live interactive course check out the link for more information and I look forward to seeing you there if on the other hand you're interested in more free content tips and tutorials check out my other videos like these subscribe to the channel so you don't miss new videos on data structures and other topics that you care about and until next time everyone I'll see you later | Итак что же такое очередь и как вы ее реализуете на языке Си это наша сегодняшняя тема  поэтому я недавно сделал видео о стеках  одна из первых структур данных которые вы изучаете в любое время когда вы начинаете реальное изучение структур данных и сегодня я хочу поговорить о ее двоюродном брате очереди так что быстрые стеки резюме были в основном просто списком элементов где вы можете получить к ним доступ только с одного конца сверху вы можете протолкнуть материал в стек вы можете вытащить материал из стека если это не имеет никакого смысла вы должны пойти посмотреть это видео  а потом вернитесь это может сделать сегодняшнее видео более осмысленным но очереди также в основном представляют собой список есть линейная структура данных но вместо того чтобы просто получать доступ на одном конце мы получаем доступ на обоих концах но очень контролируемым образом у нас есть два конца голова и хвост мы добавляем новые элементы в хвост и удаляем вещи из головы так что это структура данных которая в основном работает как очередь в продуктовом магазине или действительно любой сценарий где вы ждете в очереди в основном вы добавляете вещи в заднюю часть списка и снимаете вещи с передней части списка и вещи обслуживаются в как и в случае со стеками, у нас есть две операции вместо push и pop, у нас есть n Q и D Q и Q-это когда мы на самом деле добавляем что-то в очередь DQ-это когда мы удаляем что-то из очереди, и вы обнаружите, что мы действительно используем сигналы действительно везде в программном обеспечении в любое время, когда у вас есть вещи, прибывающие это может быть работа это может быть сетевое подключение что угодно в любое время, когда у вас есть что-то, что прибывает, что вы хотите обработать в порядке  если вы хотите очередь и как стек мы можем реализовать очередь с массивом или со связанным списком сегодня мне немного не хватает времени поэтому я собираюсь сосредоточиться на реализации связанного списка и мы возьмем базовую очередь массива в будущем видео и конечно если вы никогда не видели связанные списки раньше и это немного сбивает с толку проверьте мои видео со связанным списком просто чтобы наверстать упущенное ссылка в описании также как всегда исходный код доступен через patreon проверьте описание для получения дополнительной информации о том как получить доступ но теперь давайте перейдем к коду если вы видели видео с закусками это код должен выглядеть знакомым для начала я просто взял этот код заменил стек на Q и удалил код из push и pop так что теперь мы  у нас есть эти пустые NQ и DQ у нас есть наши две операции и у нас есть main который вызывает эти функции но которые конечно не будут работать прямо сейчас потому что наш код еще ничего не делает и как и раньше мы работаем с intz просто потому что они просты и мы могли бы легко использовать двойные плавающие структуры указатели все что мы хотим на самом деле но вставить быстро и легко так что это то что мы собираемся использовать сегодня теперь первое что я собираюсь изменить это определение того что мы хотим назвать очередью здесь со стеком все происходит на одном конце все толкает и выскакивает происходит на этом одном конце вверху так что это было достаточно просто сохранить один указатель и назвать его нашим стеком но с моей очередью я хочу отслеживать два указателя мне нужно отслеживать голову и хвост которые будут объединены в другую структуру и мы будем называть эту структуру очередью это так  это в основном просто все данные, которые нам понадобятся для представления нашей очереди, и я собираюсь последовать совету, который я рекомендовал себе в своем видео stacks, и сделать функцию инициализации, которая настраивает нашу очередь на объектно-ориентированном языке, это будет в вашем конструкторе, но здесь это просто функция, которая будет принимать указатель на нашу структуру, чтобы мы могли ее изменить.  а потом мы собираемся установить и голова, и хвост в null, что означает наша очередь начинается с записями и сохранение некоторых набрав давайте переименуем эти мои очереди, только очередь мне нравится это немного лучше, теперь в ПД мы собираемся создать новый узел с помощью вызова malloc, мы установили его значение на значение, которое мы хотим добавить в куб и мы поставили ее рядом указатель к цели, потому что это будет в конце списка, так что он не будет ничего после теперь нам нужно подключить его в наш список, поэтому мы будем проверять, если у нас будет хвост этого списка, если мы делаем, мы просто подключите его к следующему указатель новый узел, который мы только что создали, а затем положил хвост должен быть новый узел, так что теперь новый хвост теперь одна последняя вещь, если список был пуст, и мы только что добавили первый элемент потом голова будет по-прежнему иметь значение null, так что давай просто проверить, чтобы увидеть, если это так и если мы можем просто установить голову, чтобы указать, что новый узел, а так, только чтобы резюмировать, мы создали новый узел, если есть хвост, то подключите до этого новый хвост и затем мы убеждаемся, что голова по-прежнему имеет смысл и что на самом деле это, конечно, нам еще нужно вернуть true, если мы были успешны и пусть вернется и вернет false, если Малик потерпел неудачу в основном просто имея в виду что мы не смогли получить больше памяти потому что это действительно случай отказа этой функции если конечно кто то не передаст плохой указатель теперь давайте посмотрим на D Q теперь в любое время когда вы пишете функцию полезно думать о ее случаях ошибок что я могу проверить сейчас чтобы увидеть возможна ли эта операция которая была запрошена даже сейчас С D Q основная проверка которую мы хотим сделать впереди это пустой случай если очередь пуста мы ничего не можем сделать я не могу вернуть целое число из пустой очереди поэтому мы сначала проверим это мы только что вернувшийся после этого мы собираемся сохранить указатель на головку и захватить целое значение в голове затем мы просто перемещаем текущую головку чтобы указать на узел после него это то же самое что мы сделали с нашим стеком а затем конечно если у нас закончатся узлы нам нужно обязательно сообщить хвосту и установить его равным нулю мы определенно не хотим оставлять наш хвост болтающимся там а затем конечно же мы возвращаем наш сохраненный результат хорошо так что резюмируя мы проверяем пустую очередь просто чтобы убедиться что мы можем сделать то что нас просят сделать а затем мы сохраняем голову из очереди Ой подождите я забыл освободи мою память это нехорошо это была бы утечка памяти если бы я забыл это сделать так что теперь мы экономим  результат и вы собираетесь вернуться а затем мы наконец удалим его из фактического списка и это выглядит довольно хорошо давайте убедимся что он работает здесь в main нам нужно очистить некоторые вещи в основном только инициализацию теперь давайте вызовем нашу функцию init Q и эти очереди все еще называются s1 s2 и s3 я сказал вам что я просто скопировал код стека в качестве стартера так что это будет меня беспокоить так что давайте изменим их чтобы сделать их q1 q2 и q3 и мы инициализируем последние два Q это хорошо мило на самом деле я приятно удивлен что не сделал этого испортим там что нибудь и мы запустим его и он работает это точно так же как наш пример стека за исключением того что порядок перевернут поэтому мы получаем значения в порядке first in first out или FIFO а не в порядке last in first out или LIFO который мы получили из нашего стека так что будем надеяться на это  как я уже упоминал вы можете построить очередь с массивом а также связанным списком это не сложно за исключением того что мы должны быть немного осторожны с нашими индексами массива и тем как они обернуты вокруг этого обычно используется оператор модуля но мы поговорим об этом в будущем видео также я знаю что многие из вас ждали больше информации о предстоящем курсе который я предлагаю в июле я рад сообщить вам сегодня что страница курса теперь находится по следующему адресу подпишитесь на курс пространство конечно ограничено потому что мое время ограничено и это будет гибридное видео и живой интерактивный курс проверьте ссылку для получения дополнительной информации и я с нетерпением жду встречи с вами там если с другой стороны вы заинтересованы в более бесплатном контенте советы и учебники проверьте мои другие видео как эти подпишитесь на канал чтобы не пропустить новые видео по структурам данных и другим темам которые вас волнуют и до следующего раза все я увижу вас позже |