

Bài: Queue trong C#

Xem bài học trên website để ủng hộ Kteam: [Queue trong C#](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

Dẫn nhập

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về [STACK TRONG C#](#). Hôm nay chúng ta sẽ cùng tìm hiểu về **Queue trong C#**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- [BIẾN](#) , [KIỂU DỮ LIỆU](#), [TOÁN TỬ](#) trong C#
- [CÂU ĐIỀU KIỆN](#) trong C#
- Cấu trúc cơ bản của [VÒNG LẶP](#), [HÀM](#) trong C#
- [MẢNG](#) trong C#
- [LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#](#)

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Queue là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Queue.

Queue là gì?

Queue (hay còn gọi là **hàng đợi**) là một cấu trúc dữ liệu hoạt động theo nguyên lý **FIFO (First In First Out)**. Người ta hay gọi nó là hàng đợi bởi vì nó hoạt động giống như việc xếp hàng đợi chờ gì đó trong thực tế vậy.

Giả sử ta đến xếp hàng để mua vé như hình dưới:



- Khi đó người xếp hàng đầu tiên sẽ là người mua được vé đầu tiên. Đây chính là nguyên lý **First In First Out** (vào trước ra trước) của **Queue**.
- Hãy tưởng tượng những người đứng xếp hàng đó chính là 1 danh sách (**Queue**), mỗi người là 1 phần tử của danh sách. Thế là ta đã có cấu trúc dữ liệu **Queue** trong lập trình rồi đó.

Trong C#, [Queue](#) là một Collections đại diện cho một danh sách hoạt động theo nguyên lý **FIFO** đã trình bày ở trên.

Vì C# đã hỗ trợ sẵn cấu trúc dữ liệu [Queue](#) rồi nên chúng ta chỉ tìm hiểu cách sử dụng nó thôi. Còn cách tổ chức nó như thế nào sẽ được trình bày trong series về CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT.

Đặc điểm của Queue

- Là một danh sách lưu trữ các đối tượng nhưng không thể truy cập các phần tử thông qua chỉ số phần tử được.
- Hành động thêm phần tử vào [Queue](#) được gọi là **Enqueue** (xếp hàng).
- Hành động lấy phần tử ra khỏi [Queue](#) được gọi là **Dequeue** (ra khỏi hàng). Và luôn luôn lấy ra phần tử được thêm vào đầu tiên.

[Queue](#) rất giống [Stack](#) chỉ khác ở nguyên lý hoạt động thôi nên [Stack](#) có gì [Queue](#) sẽ có cái tương tự như vậy.

Do [Queue](#) cũng là 1 Collections nên để sử dụng ta cần thêm thư viện [System.Collections](#) bằng câu lệnh:

```
using System.Collections;
```

Trước khi sử dụng ta cần khởi tạo vùng nhớ bằng toán tử [new](#):

C#:

```
// khởi tạo 1 Queue rỗng  
Queue MyQueue = new Queue();
```

Bạn cũng có chỉ định sức chứa (Capacity) ngay lúc khởi tạo bằng cách thông qua **constructor** được hỗ trợ sẵn:

C#:

```
// khởi tạo 1 Queue và chỉ định sức chứa ban đầu là 5  
Queue MyQueue2 = new Queue(5);
```

Bạn cũng có thể khởi tạo 1 **Queue** chứa các phần tử được sao chép từ một danh sách khác:

C#:

```
// khởi tạo 1 mảng bất kỳ  
ArrayList MyArray = new ArrayList();  
MyArray.Add(5);  
MyArray.Add(9);  
MyArray.Add(10);  
  
// Khởi tạo 1 Queue và sao chép giá trị của các phần tử từ MyArray vào Queue.  
Queue MyQueue3 = new Queue(MyArray);
```

Một số thuộc tính và phương thức hỗ trợ sẵn trong Queue

Một số thuộc tính thông dụng trong [Queue](#):

TÊN THUỘC TÍNH	Ý NGHĨA
Count	Trả về 1 số nguyên là số phần tử hiện có trong Queue .

Một số phương thức thông dụng trong [Queue](#):

TÊN PHƯƠNG THỨC	Ý NGHĨA
-----------------	---------

Clear()	Xoá tất cả các phần tử trong Queue .
Clone()	Tạo 1 bản sao từ Queue hiện tại.
Contains (object Value)	Kiểm tra đối tượng Value có tồn tại trong Queue hay không.
CopyTo(Array array, int Index)	Thực hiện sao chép tất cả phần tử trong Queue sang mảng một chiều array từ vị trí Index của array.
Enqueue ()	Trả về giá trị của đối tượng tại vị trí đầu trong Queue (phần tử thêm vào đầu tiên) nhưng không xoá phần tử khỏi Queue .
Dequeue()	Trả về giá trị của đối tượng tại vị trí đầu trong Queue (phần tử thêm vào đầu tiên) đồng thời xoá phần tử khỏi Queue .
Push(object Value)	Thêm một phần tử có giá trị Value vào đầu Queue .
ToArray()	Tạo ra 1 mảng các object chứa tất cả các phần tử trong Queue và trả về mảng đó.

Một ví dụ đơn giản về sử dụng Queue

Cách sử dụng Queue hoàn toàn tương tự cách sử dụng Stack. Một ví dụ đơn giản về sử dụng Queue:

C#:

```
// Tạo 1 Queue rỗng
Queue MyQueue4 = new Queue();

// Thực hiện thêm vài phần tử vào Queue thông qua hàm Enqueue.
MyQueue4.Enqueue("HowKteam");
MyQueue4.Enqueue("Free");
MyQueue4.Enqueue("Education");

// Thử sử dụng các phương thức của Queue.
Console.WriteLine(" Sophan tu hien tai cua Queue la: {0}", MyQueue4.Count);

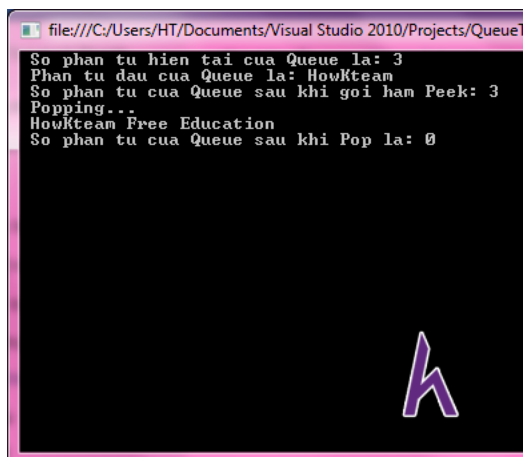
// Lưu ý ở đây ta chỉ muốn xem giá trị mà không muốn nó khỏi Queue thì ta sẽ dùng Peek.
Console.WriteLine(" Phan tu dau cua Queue la: {0}", MyQueue4.Peek());

// Thử kiểm tra lại số phần tử để chắc chắn rằng hàm Peek không xoá phần tử ra khỏi Queue.
Console.WriteLine(" Sophan tu cua Queue sau khi goi ham Peek: {0}", MyQueue4.Count);


// Thực hiện xoá các phần tử ra khỏi Queue thông qua hàm Dequeue.
Console.WriteLine(" Popping...");
int Length = MyQueue4.Count;
for (int i = 0; i < Length; i++)
{
    Console.Write(" " + MyQueue4.Dequeue());
}
Console.WriteLine();

// Kiểm tra lại số phần tử của Queue sau khi Pop
Console.WriteLine(" Sophan tu cua Queue sau khi Pop la: {0}", MyQueue4.Count);
```

Kết quả: khi chạy đoạn chương trình trên:



```
file:///C:/Users/HT/Documents/Visual Studio 2010/Projects/Queue1
Số phần tử hiện tại của Queue là: 3
Phần tử đầu của Queue là: HowKteam
Số phần tử của Queue sau khi gọi hàm Peek: 3
Popping...
HowKteam Free Education
Số phần tử của Queue sau khi Pop là: 0
```



Kết luận

Nội dung bài này giúp các bạn nắm được:

- Queue là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Queue.

Bài học sau chúng ta sẽ cùng tìm hiểu về [BITARRAY TRONG C#](#).

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.