Queue

quarta-feira, 16 de outubro de 2024 13:48

Also store elements in a sequential & specific order but following the FIFO order.

↳ Not associated by index values (unlike vectors)

↳ Can only access elements in the front or back

↳ Never in the middle !!!

Create a Queue

To create a queue, use the `queue` keyword, and specify the **type** of values it should store within angle brackets `<>` and then the name of the queue, like: `queue<type> queueName`.

```
// Create a queue of strings called cars
queue<string> cars;
```

Note: The type of the queue (string in our example) cannot be changed after its been declared.

Note: You cannot add elements to the queue at the time of declaration, like you can with vectors:

```
queue<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

Add Elements

To add elements to the queue, you can use the `.push()` function after declaring the queue.

The `.push()` function adds an element at the end of the queue:

Example

```
// Create a queue of strings
queue<string> cars;

// Add elements to the queue
cars.push("Volvo");
cars.push("BMW");
cars.push("Ford");
cars.push("Mazda");
```

The queue will look like this:

```
Volvo (front (first) element)
BMW
Ford
Mazda (back (last) element)
```

Access Queue Elements

You cannot access queue elements by referring to index numbers, like you would with arrays and vectors.

In a queue, you can only access the element at the front or the back, using `.front()` and `.back()` respectively:

Example

```
// Access the front element (first and oldest)
cout << cars.front();  // Outputs "Volvo"

// Access the back element (last and newest)
cout << cars.back();  // Outputs "Mazda"
```

Try it Yourself »

Change Front and Back Elements

You can also use `.front` and `.back` to change the value of the front and back elements:

Example

```
// Change the value of the front element
cars.front() = "Tesla";

// Change the value of the back element
cars.back() = "VW";

// Access the front element
cout << cars.front();  // Now outputs "Tesla" instead of "Volvo"

// Access the back element
cout << cars.back();  // Now outputs "VW" instead of "Mazda"
```

Try it Yourself »

Remove Elements

You can use the `.pop()` function to remove an element from the queue.

This will remove the front element (the first and oldest element that was added to the queue):

Example

```
// Create a queue of strings
queue<string> cars;

// Add elements to the queue
cars.push("Volvo");
cars.push("BMW");
cars.push("Ford");
cars.push("Mazda");

// Remove the front element (Volvo)
cars.pop();

// Access the front element (Now BMW)
cout << cars.front();
```

Try it Yourself »

Get the Size of a Queue

To find out how many elements there are in a queue, use the `.size()` function:

Example

```
cout << cars.size();
```

Try it Yourself »

Check if the Queue is Empty

Use the `.empty()` function to find out if the queue is empty or not.

The `.empty()` function returns `1` (*true*) if the queue is empty and `0` (*false*) otherwise:

Example

```
queue<string> cars;
cout << cars.empty(); // Outputs 1 (The queue is empty)
```

Try it Yourself »

C++ Deque

In the previous page, your learned that elements in a queue are added at the end and removed from the front.

A deque (stands for **d**ouble-**e**nded **queue**) however, is more flexible, as elements can be added and removed from both ends (at the front and the back). You can also access elements by index numbers.