In C#, a weak reference is a way to reference an object without preventing it from being garbage collected.

This is useful in situations where you want to be able to access an object if it still exists, but you don't want to prevent it from being cleaned up by the garbage collector if it's no longer needed.

To create a weak reference in C#, you can use the **WeakReference** class.

Here's an example:

```
var myObject = new MyObject();
var weakReference = new WeakReference(myObject);
```

To access the object that a weak reference refers to, you can use the **Target** property of the **WeakReference** class.

Here's an example:

```
var targetObject = weakReference.Target as MyObject;
if (targetObject != null) { // Do something with targetObject }
else { // targetObject has been garbage collected }
```

Weak references can be useful in a variety of situations. For example, you might use them in a cache to store objects that can be cleaned up if memory is running low. Or you might use them in an event listener to listen for events from an object without preventing that object from being garbage collected.

It's important to note that weak references have some limitations. Because they don't prevent objects from being garbage collected, there's no guarantee that the object will still exist when you try to access it through a weak reference.

Additionally, the garbage collector might not immediately clean up an object even if there are no strong references to it, so it's possible for a weak reference to still refer to an object that's no longer needed.

In conclusion, weak references are a useful tool in C# for referencing objects without preventing them from being garbage collected.

They can be helpful in a variety of situations, but they do have some limitations that need to be considered.