Jump Statements

In this lesson jump statements such as break, continue, return and throw will be discussed in detail

WE'LL COVER THE FOLLOWING

- break
- continue
- return
- throw

A **jump** *statement* can be used to transfer program control using keywords such as `break`, `continue`, `return`, and `throw`.

break

A `break` *statement* is used to **exit** from a case in a `switch` statement and also used to **exit** from

- `for`
- `foreach`
- `while`
- `do-while`

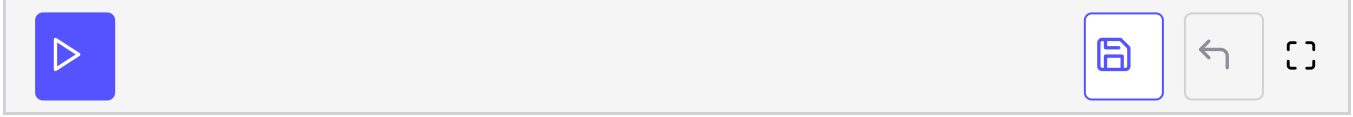
loops that will *switch* the control to the statement immediately after the **end** of the loop.

```
using System;

public class BreakExample
{
    static void Main()
    {
        int i;

        for (i = 0; i < 10; i++) // see the comparison, i < 10
        {
```

```
            if (i >= 3)
            {
                break;
                // Not run over the code, and get out of loop.
                // Note: The rest of code will not be executed,
                // & it leaves the loop instantly
            }
        }
        // Here check the value of i, it will be 3, not 10.
        Console.WriteLine("The value of i is: {0}", i);
    }
}
```



Break Example

continue

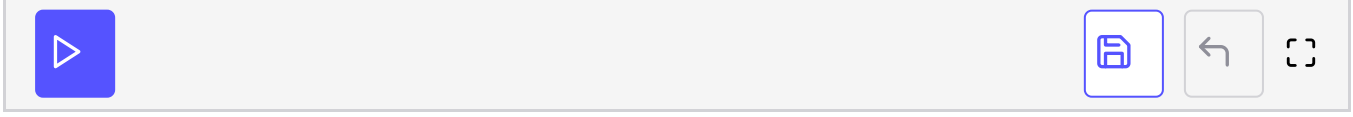
The `continue` keyword transfers program control just **before** the *end* of a *loop*.

- The **condition** for the loop is then checked
- If it is met, the *loop* performs another iteration

```
using System;

class ContinueExample
    {
        static void Main()
        {
            int counter = 0;

            for (int i = 0; i < 10; i++)
            {
                if (i >= 5)
                {
                    continue;   // Not run over the code, and return to the beginning
                                // of the scope as if it had completed the loop
                }
                counter += 1;
            }
            // Here check the value of counter, it will be 5, not 10.
            Console.WriteLine("The value of counter is: {0}", counter);
        }
}
```



Continue Example

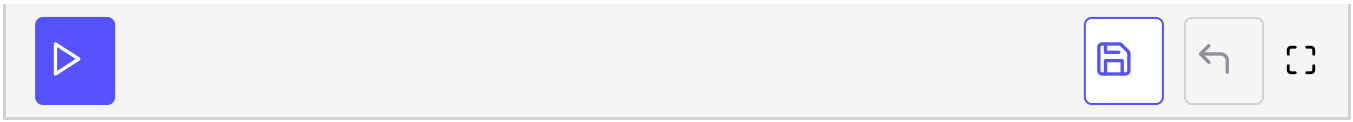
return

The `return` *keyword* identifies the **return** value for the *function* or *method* (if any), and transfers control to the end of the *function*.

> **Note:** Run the code below first. See the output, after that uncomment **line 13** in code widget below and run the code again.

```
using System;

class returnExample
{
  static int Main()
  {
    int num1 = 2;
    int num2 = 3;
    int answer = num1+num2; //computing sum of num1 and num2
    Console.WriteLine("value of answer is: {0}",answer);
    return answer; // the code terminates here from this function
    //when you uncomment the line below and run the code you'll get an "unreachable code" err
    //answer = 9; // here is a block that will not be executed
    }
}
```



When you run the code above:

- it will display the value of `answer` as **5** in the console.

When you run the code above after **uncommenting line 13**:

- it will display the value of `answer` as **5** in the console.
- the code will also give an `Error`, "**Unreachable code detected**", because the function is calling `return` statement before line **13** hence it stops executing after the `answer` is returned.

throw

The `throw` keyword throws an **exception**.

- If it is located within a `try` block, it will transfer the control to a `catch` block that matches the **exception**.

- Otherwise, it will check if any *calling* functions are contained within the matching `catch` block and transfer execution there.
- If no functions contain a `catch` block, the program may terminate because of an *unhandled* exception.

```
using System;

class throwExample
{
    static void Main()
    {
        int num1=10;
        int num2 =0;
        int result=0;
        try
        {
            result = num1/num2; //divinding by 0
        }
        catch(DivideByZeroException e)
        {
          Console.WriteLine("Exception caught: {0}", e); //exception will be caught
        }
    }
}
```

throw Example

This marks the end of this *chapter*. In the next, we will learn about **methods** in **C#**.