Fixing compile-time errors

While syntax errors like those on page 99 can be detected by the **Code Editor** in real-time, other errors that employ correct syntax cannot be detected until the code is compiled. Compile errors are typically errors of logic, and they cause the execution to halt when an "exception" occurs. For example, when incompatible data types appear in an expression, an **InvalidCastException** occurs and execution stops immediately:

The **IConvertible** interface provides methods that convert a value to a CLR type, but it cannot meaningfully convert a **bool** to **char**.

1. Type the following lines into the **Code Editor**

```
bool flag = true ;
IConvertible convertible = flag ;
char letter = convertible.ToChar( null ) ;
```

2. Press **Start** or **F5** to run the application and see execution is soon halted. The line causing the exception becomes highlighted in the **Code Editor** and an **Exception Assistant** pop-up window appears with a list of possible solutions

```
static void Main(string[] args)
{

    bool flag = true;
    IConvertible convertible = flag;
    char letter = convertible.ToChar(null);
}
```

Exception Unhandled

System.InvalidCastException: 'Invalid cast from 'Boolean' to 'Char'.'

View Details | Copy Details

Exception Settings

You can click on the **View Detail** link for more error information.

To fix this **InvalidCastException**, the code would need amendment so both values are of compatible data types.

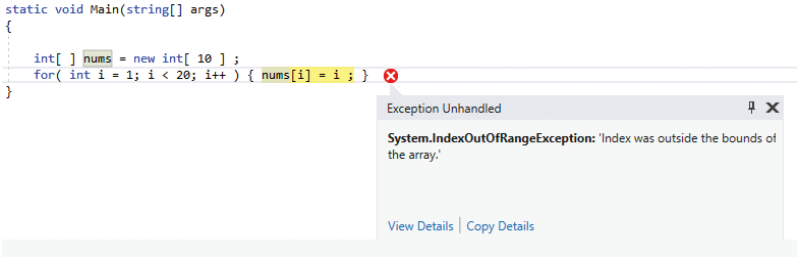
The cause of other compile errors may be less obvious without some further investigation. For example, when a loop that is reading array elements attempts to address an element index that does not exist, causing an **IndexOutOfRangeException**.

Execution halts immediately, so it is useful to examine the counter value to identify the precise iteration causing the compile error.

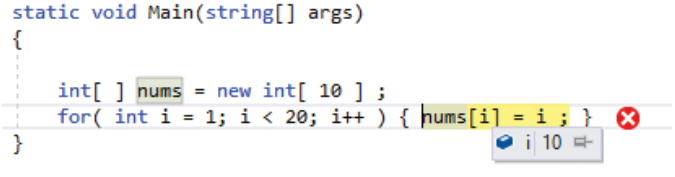
1. In the **Code Editor** type the following variable array declaration of ten elements (0-9), and a loop

```
int [ ] nums = new int [ 10 ] ;
for ( int i = 1 ; i < 20 ; i++ ) { nums [ i ] = i ; }
```

2. Press **Start** or **F5** to run the application and see execution is soon halted. The code causing the exception becomes highlighted in the **Code Editor** and an **Exception Assistant** pop-up window appears with a list of possible solutions



3. Place the cursor over the assignment to the array variable to see a pop-up appear displaying its current value



It's now clear that execution halted when the loop attempted to address **nums[10]** – beyond the bounds of last element **nums[9]**. To fix this **IndexOutOfRangeException**, the code would need amendment to end the loop after 10 iterations.

Another common compile error is the **FileNotFoundException** that occurs when a file is missing or its path name is incorrect.