SHORTCUTS AND QUALITY CODE TIPS

SHORTCUTS

ALT + SHIFT + LEFT MOUSE BUTTON – Hold the **ALT** and **SHIFT** key modifiers and drag with your **LEFT MOUSE BUTTON** to select a specific part of your code spread on multiple rows

ALT + SHIFT + ARROW KEYS (←↑↓→) – Hold the **ALT** and **SHIFT** key modifiers and use the **ARROW KEYS** to move the cursor in your code to select a specific part of your code spread on multiple rows (more precise than using the mouse)

ALT + UP DOWN (↑↓) – Hold the **ALT** key modifier and press **UP** or **DOWN** arrow keys to move the selected row(s) of code up or down.

CTRL + SHIFT + LEFT RIGHT (← →) – Hold the **CONTROL** and **SHIFT** key modifiers and press **LEFT** or **RIGHT** arrow keys to select whole words from your code

CTRL + LEFT RIGHT (← →) – Hold the **CONTROL** key modifier and press **LEFT** or **RIGHT** arrow keys to move the cursor faster throughout your code

CTRL + C – Either select some piece of code and hold **CONTROL** key modifier and press **C** to copy the selected code, or just place your cursor on a given row of code without selecting anything and perform the same operation to copy the whole row

CTRL + V – Hold the **CONTROL** key modifier and press **V** to paste the copied text from the previous operation

CTRL + K + C – Hold the **CONTROL** key modifier and press **K** then press **C** (do it fast) to comment the selected rows of code. If no rows are selected, only the row that has your cursor will be commented

CTRL + K + U - Hold the **CONTROL** key modifier and press **K** then press **U** (do it fast) to uncomment the selected rows of code. If no rows are selected, only the row that has your cursor will be uncommented

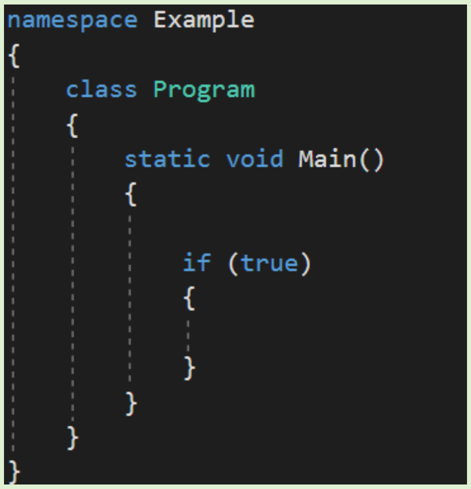
CTRL + R + R – Place your cursor within the variable that you want to rename, hold the CONTROL key modifier and press **R** and then press **R** again (do it fast), after that rename the variable and press **ENTER** to apply the changes

CTRL + K + D – Hold the **CONTROL** key modifier and press **K** and then press **D** (do it fast) to auto format your code. This will correct the spacing within your code, but it will not add or remove empty rows where it is necessary!

QUALITY CODE

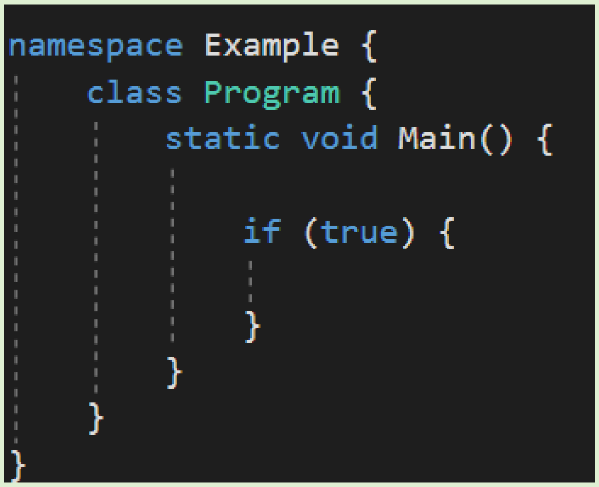
CURLY BRACKETS – Unlike other languages like Java for example, in C# both the opening and closing curly bracket "{" "}" are on new rows with nothing else on the row.

Correct:



```
namespace Example
{
    class Program
    {
        static void Main()
        {

            if (true)
            {

            }
        }
    }
}
```

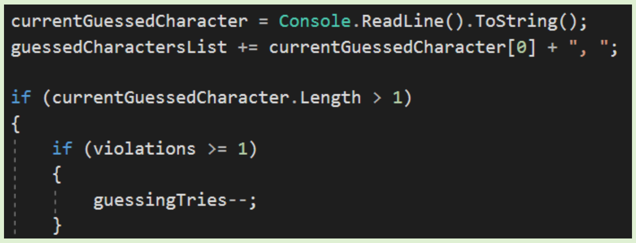
Wrong:



```
namespace Example {
    class Program {
        static void Main() {

            if (true) {

            }
        }
    }
}
```

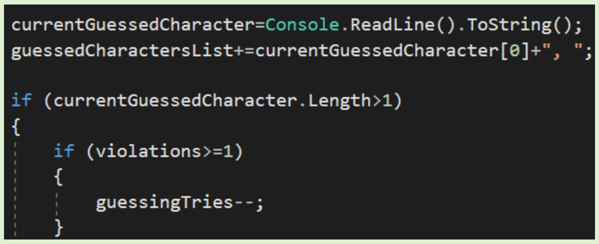
SPACES – To improve the readability of your code, you should put free spaces before and after any operators that you are using.

Correct:



```
currentGuessedCharacter = Console.ReadLine().ToString();
guessedCharactersList += currentGuessedCharacter[0] + ", ";

if (currentGuessedCharacter.Length > 1)
{
    if (violations >= 1)
    {
        guessingTries--;
    }
```

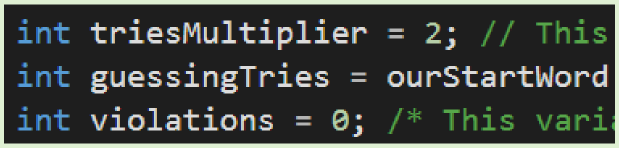
Wrong:



```
currentGuessedCharacter=Console.ReadLine().ToString();
guessedCharactersList+=currentGuessedCharacter[0]+", ";

if (currentGuessedCharacter.Length>1)
{
    if (violations>=1)
    {
        guessingTries--;
    }
```

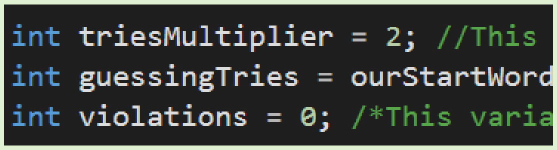
COMMENTS – You should always put one empty space after the "//" double slash comment or after the "/*" slash star comment, again to improve readability.

Correct:



```
int triesMultiplier = 2; // This
int guessingTries = ourStartWord
int violations = 0; /* This vari
```

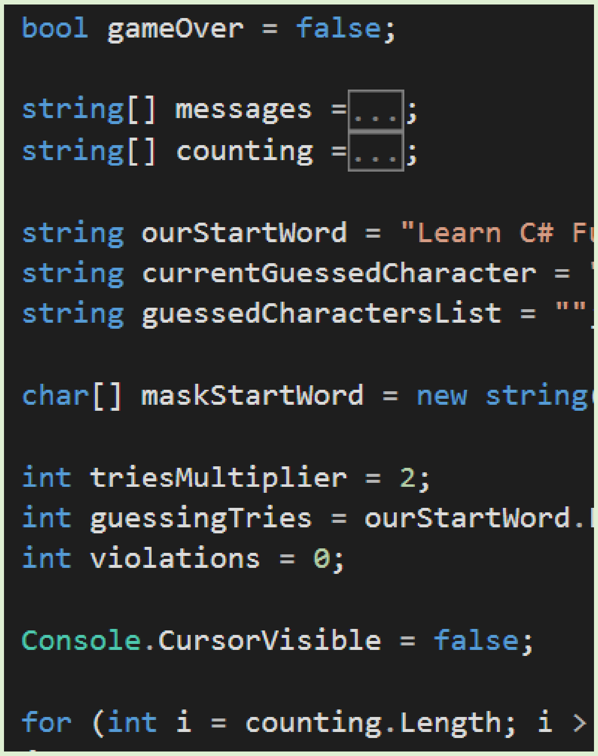
Wrong:



```
int triesMultiplier = 2; //This
int guessingTries = ourStartWord
int violations = 0; /*This varia
```

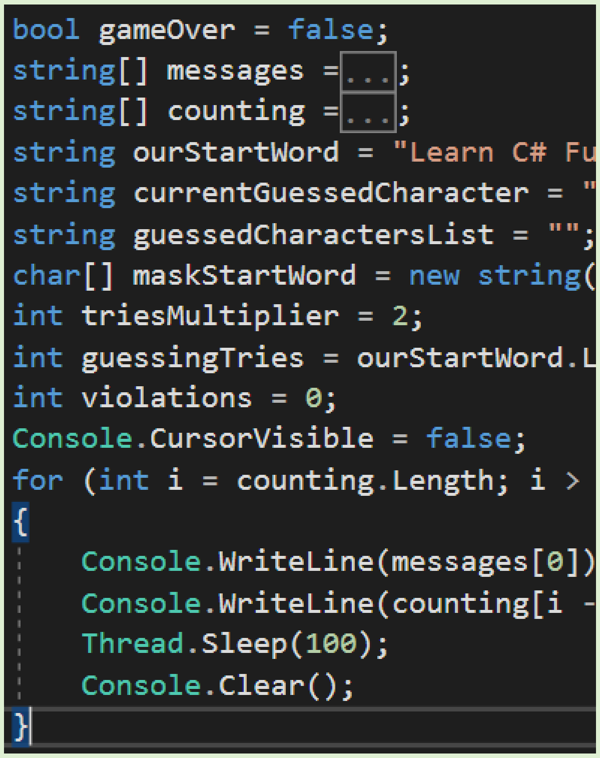
FREE ROWS – Another way to improve readability is to add empty rows between different blocks of code or between variables with different data types.

Correct:



```
bool gameOver = false;

string[] messages =...;
string[] counting =...;

string ourStartWord = "Learn C# F
string currentGuessedCharacter = 
string guessedCharactersList = ""

char[] maskStartWord = new string

int triesMultiplier = 2;
int guessingTries = ourStartWord.
int violations = 0;

Console.CursorVisible = false;

for (int i = counting.Length; i >
```

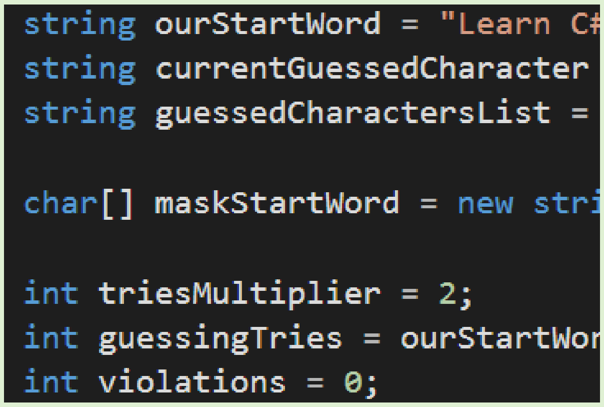
Wrong:



```
bool gameOver = false;
string[] messages =...;
string[] counting =...;
string ourStartWord = "Learn C# Fu
string currentGuessedCharacter = "
string guessedCharactersList = "";
char[] maskStartWord = new string(
int triesMultiplier = 2;
int guessingTries = ourStartWord.L
int violations = 0;
Console.CursorVisible = false;
for (int i = counting.Length; i >
{
    Console.WriteLine(messages[0])
    Console.WriteLine(counting[i -
    Thread.Sleep(100);
    Console.Clear();
}
```

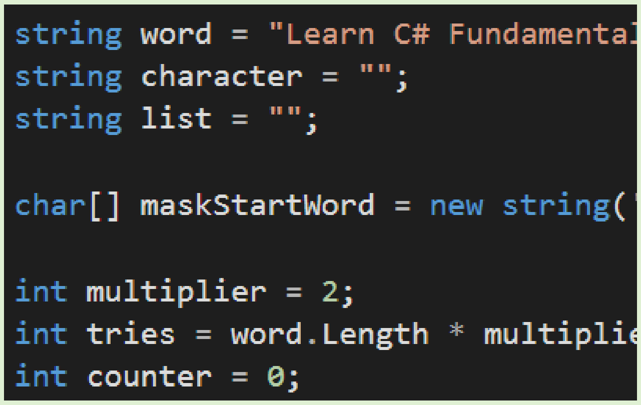
VARIABLES NAMING – The names of your variables should always be concrete and descriptive, and when you name your variables they should always be in camelCase.

Correct:



```
string ourStartWord = "Learn C#
string currentGuessedCharacter
string guessedCharactersList =

char[] maskStartWord = new stri

int triesMultiplier = 2;
int guessingTries = ourStartWor
int violations = 0;
```

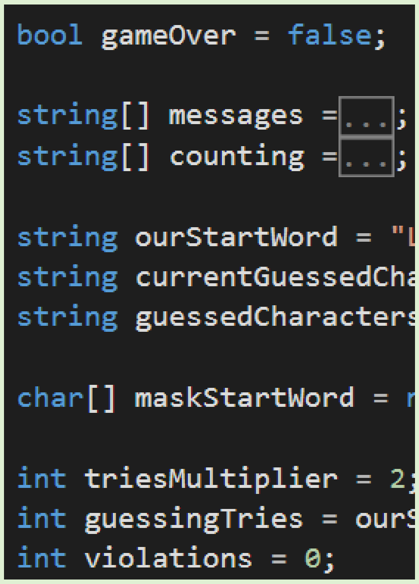
Wrong:



```
string word = "Learn C# Fundamenta
string character = "";
string list = "";

char[] maskStartWord = new string(

int multiplier = 2;
int tries = word.Length * multipli
int counter = 0;
```

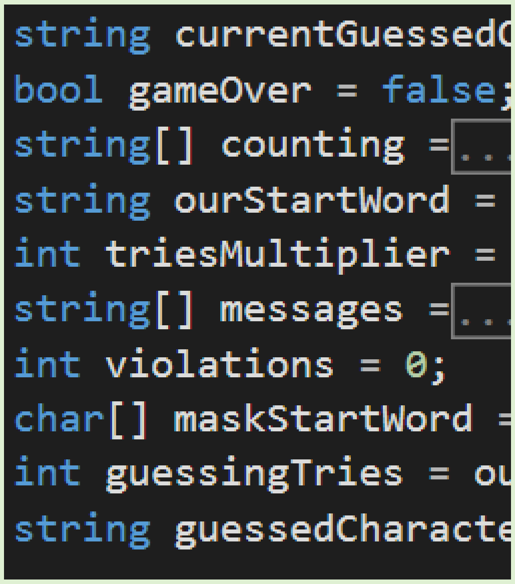
DATA TYPES – When creating your variables it's always good practice to keep the same data types together and not mix them with other data types, IF that is possible (sometimes you have no choice). This will not affect how your code works, but it will make it look so much better and easier to be understood.

Correct:



```
bool gameOver = false;

string[] messages =...;
string[] counting =...;

string ourStartWord = "
string currentGuessedCha
string guessedCharacters

char[] maskStartWord =

int triesMultiplier = 2;
int guessingTries = ourS
int violations = 0;
```

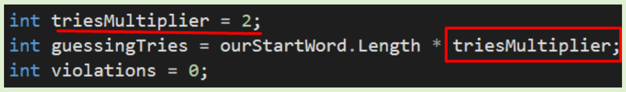
Wrong:



```
string currentGuessedC
bool gameOver = false;
string[] counting =...
string ourStartWord =
int triesMultiplier =
string[] messages =...
int violations = 0;
char[] maskStartWord =
int guessingTries = ou
string guessedCharacte
```

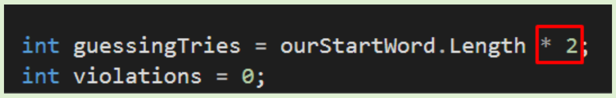
MAGICAL NUMBERS – Never use magical numbers in your code. Magical numbers are numbers that are placed randomly in your code and have no apparent reason to be there. Always extract magical numbers into variables. Just by looking at your code you should be able to understand what is what and not wonder "Where did that number come from?

Correct:



```
int triesMultiplier = 2;
int guessingTries = ourStartWord.Length * triesMultiplier;
int violations = 0;
```

Wrong:



```
int guessingTries = ourStartWord.Length * 2;
int violations = 0;
```