[My Home](https://www.codecademy.com/learn)

Course Menu

Connected to Codecademy

Fork

Get Unstuck

Tools



**Caesar Cipher**

**Objective**

**LEARN C#**

**Caesar Cipher**

By 6 a.m. on Sunday, your team’s project is nearly finished. The Object of Your Affection (the name’s a work in progress) is getting attention from other teams in the hackathon. With one day left, they’re getting desperate. To steal your project idea, your competitors have been reading your team’s emails!

As the team’s C# expert, you have been asked to write a cipher: a tool to encrypt text, making it unreadable to other teams. You’ve decided to implement the [Caesar Cipher](https://en.wikipedia.org/wiki/Caesar_cipher), which was used by the Roman Empire to encode military secrets.

To use the cipher, draw the alphabet in a circle like so:

Take every letter of your message and shift it three places to the right. For example, A becomes D, B becomes E, C becomes F, and “attack” becomes “dwwdfn”.

Your teammates can decrypt your message by reversing the process: shift each letter three places to the left.

For this project you’ll need to convert strings to arrays of characters with [ToCharArray()](https://docs.microsoft.com/en-us/dotnet/api/system.string.tochararray" \t "_blank):

string msgString = "brutus";  
char[] msgArray = msgString.ToCharArray();

You’ll also need to convert arrays of characters to strings with [Join()](https://docs.microsoft.com/en-us/dotnet/api/system.string.join):

char[] msgArray = new char[] {'b', 'r', 'u', 't', 'u', 's'};  
string msgString = String.Join("", msgArray);

The team is counting on you. Let’s get started!

**Tasks**

5/14 Complete

Mark the tasks as complete by checking them off

**Prepare for Encryption**

**1.**

You’ll build the encryption tool as an interactive console app.

First, ask the user for a secret message and store the result in a variable.

Hint

Use Console.Write() to write output to the console and Console.ReadLine() to capture input from the console.

**2.**

Convert the captured string to an array of characters. Store the result in a new variable called secretMessage.

Hint

Convert strings to arrays of characters with [ToCharArray()](https://docs.microsoft.com/en-us/dotnet/api/system.string.tochararray" \t "_blank):

string msgString = "brutus";  
char[] msgArray = msgString.ToCharArray();

**3.**

Create a new, empty array of characters to hold the encrypted message. It should be named encryptedMessage and have a length equal to the length of secretMessage.

Hint

Get the length of an array using Length. Set the length of an array when defining it:

char[] emptyEightArray = new char[8];

**Encrypt**

**4.**

We’ll need to perform encryption for every letter in the message.

Create an empty for loop that loops through each character of secretMessage.

* The iterator variable should be called i and start at 0
* It should continue as long as it is less than secretMessage.Length
* Each iteration it should increment by 1

Hint

Here’s an example for loop that has an iterator variable j starting at 2. It continues as long as j is less than 10, and it is incremented by 1 on each iteration.

for (int j = 2; j < 10; j++) {}

**5.**

Within the loop, access the character at position i in the secretMessage array and store it in a variable.

Hint

Use bracket notation to access an element in an array:

char letter = letters[6];

**6.**

Find the position of the character in the alphabet array using the method Array.IndexOf(). Store the value in a variable.

Hint

The IndexOf() method takes two arguments:

* a character array to search in
* a character to search for

It will return the position of the character in the array as an integer.

For example, this search for 'a' in the array letters:

int index = Array.IndexOf(letters, 'a');

**7.**

Add 3 to the letter position and store the value in a variable.

Hint

So far we have taken the character at index i and found its position in the alphabet. At this point we need to “move three letters to the right”.

**8.**

Find the new encrypted character by getting the character in the alphabet array with that new position.

Hint

Use bracket notation to access an element in an array:

char letter = letters[6];

**9.**

Add the encrypted character to the array encryptedMessage.

Store the character at the index i (the iterator variable).

Hint

Use bracket notation to store an element in an array:

letters[1] = 'f';

**Test and Improve**

**10.**

Your loop is finished! Now we need to convert our array of encrypted characters back into a readable string that we can print to the console.

Convert the character array to a string using String.Join() and print it to the console.

Hint

[Convert arrays of characters to strings with Join()](https://docs.microsoft.com/en-us/dotnet/api/system.string.join):

char[] msgArray = new char[] {'b', 'r', 'u', 't', 'u', 's'};  
string msgString = String.Join("", msgArray);

**11.**

Run your app and try these messages (not all of them may work yet!):

* hello converts to khoor
* citizen converts to flwlchq

Hint

Use dotnet run to run the app.

**12.**

What went wrong? When the program tried to encrypt the z in citizen, it found its index in the alphabet, 25. It looked up the letter 3 spaces to the right, which would be alphabet[28].

This threw an error because the alphabet is only 26 letters!

We can “wrap around” by using the modulo operator: %. On the line where you add 3 to the letter position, surround the expression letterPosition + 3 with parentheses and take the modulo of 26.

Now the new letter position will never go past 26.

Hint

The expression should look like this. Your variable names might be different

int newLetterPosition = (letterPosition + 3) % alphabet.Length;

**13.**

Test the code again with citizen, which converts to flwlchq.

Hint

If the program doesn’t convert the message correctly, go back to make sure you followed the steps exactly.

**Extensions**

**14.**

Well done! You built an automated encryption engine that your team can use to communicate in secrecy. The hackathon is all but won!

If you’d like, you can keep building on your app:

* The app doesn’t work with uppercase letters. Fix that by converting any message to lowercase.
* The app doesn’t work with symbols, like ! or ?. Skip any symbols in your loop so that they are not encrypted.
* Rewrite the loop as a method Encrypt() which takes a character array and key and returns an encrypted character array .
* Write a Decrypt() method which takes a character array and key and returns a decrypted character array.

char[] secretMessage = {'h', 'e', 'l', 'l', 'o'};  
   
// encrypted should equal  {'k', 'h', 'o', 'o', 'r'}  
string encrypted = Encrypt(secretMessage, 3);  
   
// decrypted should equal {'h', 'e', 'l', 'l', 'o'}  
string decrypted = Decrypt(encrypted, 3);

Hint

Convert strings to lowercase with ToLower():

// quiet will be "hey"  
string loud = "HEY";  
string quiet = loud.ToLower();

Use if and else statements to perform different actions based on different character types.

Check if a character is a letter using Char.IsLetter():

// b will be false  
char ch = "?";  
bool b = Char.IsLetter(ch);