|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Text.RegularExpressions; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace Cipher |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | Console.WriteLine("This application takes a key and the key is stripped down\n" + |
|  | "to only letters (You can use as many letters as you want).\n" + |
|  | "The application then encrypts the message you type in using the key.\n\n"); |
|  | string input; |
|  | do |
|  | { |
|  | Console.WriteLine("Enter 1 to use just a key to encrypt.\n" + |
|  | "Enter 2 to use the key and the message to encrypt.\n" + |
|  | "Enter 3 to use just the key to decrypt.\n" + |
|  | "Enter 4 to use the key and the message to decrypt.\n" + |
|  | "Enter X to exit."); |
|  | input = Console.ReadLine(); |
|  | if (input == "1") |
|  | EncryptKey(); |
|  | if (input == "2") |
|  | EncryptKeyMessage(); |
|  | if (input == "3") |
|  | DecryptKey(); |
|  | if (input == "4") |
|  | DecryptKeyMessage(); |
|  | } while (input.ToLower() != "x"); |
|  | } |
|  | static void EncryptKey() |
|  | { |
|  | Regex rgx = new Regex("[^a-zA-Z]"); |
|  |  |
|  | char[] key = null; |
|  | do |
|  | { |
|  | Console.Write("Enter the key: "); |
|  | key = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | if (key.Length > 0) |
|  | { |
|  | if (char.IsLetter(key[0])) |
|  | break; |
|  | } |
|  | else |
|  | Console.WriteLine("Please use at least one letter."); |
|  | } while (true); |
|  |  |
|  | Console.Write("Enter the message: "); |
|  |  |
|  | char[] message = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  |  |
|  | for (int i = 0; i < message.Length; i++) |
|  | { |
|  | int test = (((message[i] + key[(i % key.Length)]) - 128) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | message[i] = (char)(test + 64); |
|  | } |
|  | string mixed = new string(message); |
|  | Console.WriteLine($"{mixed}"); |
|  | } |
|  |  |
|  | static void DecryptKey() |
|  | { |
|  | Regex rgx = new Regex("[^a-zA-Z]"); |
|  |  |
|  | char[] key = null; |
|  | do |
|  | { |
|  | Console.Write("Enter the key: "); |
|  | key = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | if (key.Length > 0) |
|  | { |
|  | if (char.IsLetter(key[0])) |
|  | break; |
|  | } |
|  | else |
|  | Console.WriteLine("Please use at least one letter."); |
|  | } while (true); |
|  |  |
|  | Console.Write("Enter the message: "); |
|  |  |
|  | char[] message = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  |  |
|  | for (int i = 0; i < message.Length; i++) |
|  | { |
|  | int test = (((message[i] - 38) - (key[(i % key.Length)] - 64)) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | message[i] = (char)(test + 64); |
|  | } |
|  | string mixed = new string(message); |
|  | Console.WriteLine($"{mixed}"); |
|  | } |
|  |  |
|  | static void EncryptKeyMessage() |
|  | { |
|  | Regex rgx = new Regex("[^a-zA-Z]"); |
|  |  |
|  | char[] key = null; |
|  | do |
|  | { |
|  | Console.Write("Enter the key: "); |
|  | key = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | if (key.Length > 0) |
|  | { |
|  | if (char.IsLetter(key[0])) |
|  | break; |
|  | } |
|  | else |
|  | Console.WriteLine("Please use at least one letter."); |
|  | } while (true); |
|  |  |
|  | Console.Write("Enter the message: "); |
|  |  |
|  | char[] message = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | char[] output = (char[]) message.Clone(); |
|  |  |
|  | for (int i = 0; i < message.Length; i++) |
|  | { |
|  | if (i < key.Length) |
|  | { |
|  | int test = (((message[i] + key[(i % key.Length)]) - 128) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | output[i] = (char)(test + 64); |
|  | } |
|  | else |
|  | { |
|  | int test = (((message[i] + message[(i - key.Length)]) - 128) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | output[i] = (char)(test + 64); |
|  | } |
|  | } |
|  | string mixed = new string(output); |
|  | Console.WriteLine($"{mixed}"); |
|  | } |
|  |  |
|  | static void DecryptKeyMessage() |
|  | { |
|  | Regex rgx = new Regex("[^a-zA-Z]"); |
|  |  |
|  | char[] key = null; |
|  | do |
|  | { |
|  | Console.Write("Enter the key: "); |
|  | key = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | if (key.Length > 0) |
|  | { |
|  | if (char.IsLetter(key[0])) |
|  | break; |
|  | } |
|  | else |
|  | Console.WriteLine("Please use at least one letter."); |
|  | } while (true); |
|  |  |
|  | Console.Write("Enter the message: "); |
|  |  |
|  | char[] message = rgx.Replace(Console.ReadLine().ToUpper(), "").ToArray(); |
|  | char[] output = (char[])message.Clone(); |
|  |  |
|  | for (int i = 0; i < message.Length; i++) |
|  | { |
|  | if (i < key.Length) |
|  | { |
|  | int test = (((message[i] - 38) - (key[(i % key.Length)] - 64)) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | output[i] = (char)(test + 64); |
|  | } |
|  | else |
|  | { |
|  | int test = (((message[i] - 38) - (output[(i - key.Length)] - 64)) % 26); |
|  | if (test == 0) |
|  | test = 26; |
|  | output[i] = (char)(test + 64); |
|  | } |
|  | } |
|  | string mixed = new string(output); |
|  | Console.WriteLine($"{mixed}"); |
|  | } |
|  | } |
|  |  |
|  | } |