DOCUMENTATION Cryptage

Le cryptage se passe en 4 étapes:

Étape 1 (Remplissage de la grille) :

```
#region ETAPE 1
static string Alpha = "ABCDEFGHIKLMNOPQRSTUVWXYZ"; // Alphabet complet
static char[,] grille = new char[5, 5]; // Grille de cryptage vide
static bool paire = true;               // Si le message est paire
static string AncienMessageACrypter, MessageADecrypter;
// Message à crypter avec les esapces && message à décrypter avec les espaces


public static string Delta(string clé)
{
    string Beta = clé.ToUpper() + Alpha; // Cumul la clé (en majuscule) et l'alphabet complet
    for (int i = 0; i < clé.Length; i++)
    {
        int j = i + 1;
        for (; j < Beta.Length - 1; j++)
        {
            if (Beta[i] == Beta[j])
                Beta = Beta.Remove(j, 1); // Retire les doublons
        }
    }
    return Beta;
}
```

```
public static void RemplirTableau()
{
    Console.Write("Entrez la clé de chiffrement : ");   // Clé pour chiffrer
    string cle = Console.ReadLine();
    string Epsilone = Delta(cle);  // Cumul de la clé rentrée avec l'alphabet sans doublon
    foreach (char c in Epsilone)
    {
        if (c == 'J')
            Epsilone = Epsilone.Replace('J', 'I'); // Remplacer tous les J par des I
    }
    int Ligne = grille.GetLength(0), Colonne = grille.GetLength(1); // pour ne pas se perdre avec 0 et 1

    for (int i = 0, k = 0; i < Ligne; i++)
    {
        for (int j = 0; j < Colonne; k++, j++)
        {
            grille[i, j] = Epsilone[k]; // Remplissage du tableau
        }
    }
}
```

Étapes 2 & 3 (Traitement de la chaîne) :

```
// Fonction pour garder les espaces, les majuscules/minuscules au meme endroit
public static string RetoucherChaine(string input, string output)
{
    StringBuilder aRetourner = new StringBuilder(output);
    for (int i = 0; i < input.Length; i++)
    {
        if (!char.IsLetter(input[i]))
            aRetourner = aRetourner.Insert(i, input[i].ToString());
        if (char.IsLower(input[i]))
            aRetourner[i] = Char.ToLower(aRetourner[i]);
    }
    return aRetourner.ToString();
}
```

Étape 4 (Chiffrement) :

```
public static string Crypter()
{
    RemplirTableau();                   // Remplissage du tableau avec la clé et l'alphabet sans doublon
    string[] PasCrypter = MessageACrypter();
    string Crypter = null;
    for (int i = 0; i < PasCrypter.Length; i++)
    {
        Crypter += ChiffrementLettres(PasCrypter[i]);
    }
    return RetoucherChaine(AncienMessageACrypter, Crypter);     // Remet les espace au meme endroit
}
```

```
public static string ChiffrementLettres(string l1)
{
    char[] TPaires = l1.ToCharArray();
    int coorLigne = -1, coorColonne = -1, cl1 = -1, cc1 = -1; // Déclaration des variables pour les
                                                              // coordonnées des lignes & colonnes
    bool Sl = false;
    foreach (char c in TPaires)     // Cherche les coordonner des deux lettres d'une paires
    {
        for (int j = 0; j < grille.GetLength(0); j++)
        {

            for (int k = 0; k < grille.GetLength(1); k++)
            {

                if (grille[j, k] == c)
                {
                    coorLigne = j;
                    coorColonne = k;

                    if (Sl == false)
                        Sl = true;
                    else
                        Sl = false;
```

```
                    if (Sl == true)
                    {
                        cl1 = coorLigne;
                        cc1 = coorColonne;
                    }
                }
            }
        }
    }
    return LettresACrypter(cl1, cc1, coorLigne, coorColonne);          // Crypte les lettre avec les coordonnées

}
```

```
public static string[] MessageACrypter()
{
    string[] PaireMess; // Tableau de string (pour les paires de lettres)
    string mac = null; // mac = Message à crypter (entré par l'utilisateur)
    do
    {
        Console.Write("Veuillez entrez votre message : ");
        mac = Console.ReadLine();
    }
    while (mac == null);
    AncienMessageACrypter = mac;    // Message rentrée par l'utilisateur avec les espaces
    for (int i = 0; i < mac.Length; i++) // Retire tout ce qui n'est pas une lettre dont les espaces
    {
        if (!Char.IsLetter(mac, i))
        {
            mac = mac.Remove(i, 1);
            i--;
        }
        if (mac[i] == 'J')
        {
            mac = mac.Replace('J', 'I');
        }
    }
```

```
    mac = mac.ToUpper(); // Mettre en majuscule
    if (mac.Length % 2 != 0)// Rajoute 'X' au message si le nombre de lettres est impaire
    {
        mac += 'X';
        paire = false;
    }

    PaireMess = new string[mac.Length / 2]; // Longueur du tableau de paire = la moitié de la longeur du message
                                            // à crypter
    for (int i = 0, j = 0; j < PaireMess.Length; j++, i += 2) // Confection des paires
    {
        char c = mac[i], c2 = mac[i + 1];
        PaireMess[j] += c;
        PaireMess[j] += c2;
    }
    return PaireMess;
}
```

Décrypter :

```
public static string DeCrypter()
{
    Console.Write("Message à Décrypter : ");
    string CrypterOld = Console.ReadLine();
    MessageADecrypter = CrypterOld;
    CrypterOld = CrypterOld.ToUpper();
    for (int i = 0; i < CrypterOld.Length; i++)
    {
        if (!Char.IsLetter(CrypterOld[i]))
        {
            CrypterOld = CrypterOld.Remove(i, 1);
            i--;
        }
    }
    string[] Crypter = new string[CrypterOld.Length / 2];
    for (int i = 0, j = 0; j < Crypter.Length; j++, i += 2)
    {
        Crypter[j] += CrypterOld[i];
        Crypter[j] += CrypterOld[i + 1];

    }
```

```
    Console.Write("Clé de cryptage : ");
    string Cle = Console.ReadLine();
    RemplirTableauDecrypto(Delta(Cle));
    string Decrypter = null;

    for (int i = 0; i < Crypter.Length; i++)
    {
        Decrypter += DechiffrementLettres(Crypter[i]);
    }
    if (!paire)
        Decrypter = Decrypter.Remove((Decrypter.Length - 1), 1);
    return RetoucherChaine(MessageADecrypter, Decrypter);
}
```

```
public static void RemplirTableauDecrypto(string cle)
{
    grille = new char[5, 5];
    string Epsilone = cle;  // Définition de la clé en dur
    foreach (char c in Epsilone)
    {
        if (c == 'J')
            Epsilone = Epsilone.Replace('J', 'I');
    }
    int Ligne = grille.GetLength(0), Colonne = grille.GetLength(1); // pour ne pas se perdre avec 0 et 1

    for (int i = 0, k = 0; i < Ligne; i++)
    {
        for (int j = 0; j < Colonne; k++, j++)
        {
            grille[i, j] = Epsilone[k]; // Remplissage du tableau
        }
    }
}
```

```
public static string DechiffrementLettres(string l1)
{
    char[] TPaires = l1.ToCharArray();
    int coorLigne = -1, coorColonne = -1, cl1 = -1, cc1 = -1; // Déclaration des variables
    bool Sl = false;
    foreach (char c in TPaires)
    {
        for (int j = 0; j < grille.GetLength(0); j++)
        {

            for (int k = 0; k < grille.GetLength(1); k++)
            {

                if (grille[j, k] == c)
                {
                    coorLigne = j;
                    coorColonne = k;

                    if (Sl == false)
                        Sl = true;
                    else
                        Sl = false;
```

Le Main :

```
static void Main(string[] args)
{
    /*
     message en clair:"Hello World";
     message chiffré :"Ecttq Vvgmb
     string texte = "Hello World";
     string texte = "General Mac Mahon";
     string texte = "Pilonage le huit de huit a midi";
     string texte = "Attaque Malakoff a midi";
     string texte = "Par Premier Reg Zouaves";
     string texte = "Et Septieme Reg Infanterie";
     string texte = "Signe General Pelissier";
     string texte_chiffre = Chiffrer(text, "Poker"); // CLE = “cipher”
     string texte_dechiffre = Dechiffrer(cipherText, "Poker"); // CLE = “cipher”
     Console.WriteLine(texte_chiffre);
     Console.WriteLine(texte_dechiffre);
    */
    Console.WriteLine(Crypter());
    Console.WriteLine();
    Console.WriteLine(DeCrypter());

    Console.ReadKey();
}
```