TD8 : éléments de correction

Exercice 4

```
Fonction : CreerGrille
Précondition : alphabet > 0
Entrée :        - alphabet : entier
                - indiceLigne1 : entier
                - indiceColonne1 : entier
Local :         /
Sortie :        - grille : tableau 2D de alphabet*alphabet chaînes de caractères (de taille 3 chacune)
Début
|       Soit grille = tableau 2D de alphabet*alphabet chaînes de caractères initialisées à ""
|       Soit i = indiceLigne1
|       Soit j = indiceColonne1
|       Soit grille[i, j] = ChaîneSur3Caractères(1)        // convertit l'entier 1 en la chaîne "001"
|       Pour k de 2 à alphabet * alphabet Faire
|       |       Soit inext = i - 1
|       |       Soit jnext = j + 1
|       |       // dépassement vers la droite
|       |       Si jnext == alphabet Alors
|       |       |       Soit jnext = 0
|       |       Fin Si
|       |       // dépassement vers la gauche
|       |       Si inext < 0 Alors
|       |       |       Soit inext = alphabet - 1
|       |       Fin Si
|       |       // case déjà occupée
|       |       Si grille[inext, jnext] n'est pas égale à "") Alors
|       |       |       Soit inext = i + 1
|       |       |       Soit jnext = j
|       |       |       Si inext ≥ alphabet Alors
|       |       |       |       Soit inext = 0
|       |       |       Fin Si
|       |       Fin Si
|       |       // cas général
|       |       Soit grille[inext, jnext] = ChaineSur3Caractères(k)
|       |       Soit k = k + 1
|       |       Soit i = inext
|       |       Soit j = jnext
|       Fin Pour
|       Retourner grille
Fin
```

```
Fonction : DonnerIndice
Précondition : c appartient à l'ensemble {'A', ... , 'Z', ' ', '.'}
Entrée :        - c : caractère
Local :         /
Sortie :        - entier
Début
|       Si c == ' ' Alors
|       |       Retourner 26
|       Sinon Si c == '.' Alors
|       |       Retourner 27
|       Sinon
|       |       Retourner ASCII(c) - ASCII('A')        // conversion code ASCII → caractère
|       Fin Si
Fin
```

```
Fonction : Code2Gramme
Précondition : alphabet > 0
Entrée :        - grille : tableau 2D de alphabet*alphabet chaînes de caractères
                - message : chaîne de caractères
Local :         - indice1, indice2 : entier
Sortie :        - messageCode : chaîne de caractères
Début
|       Soit messageCode = ""
|       Soit indice1 = -1
|       Soit indice2 = -1
|       Pour i de 0 à Longueur(message) - 1 avec un pas de +2 Faire    // lecture des caractères 2 par 2
|       |       Soit indice1 = DonnerIndice(message[i])
|       |       Soit indice2 = DonnerIndice(message[i + 1])
|       |       Soit messageCode = messageCode + grille[indice1, indice2]
|       Fin Pour
|       Retourner messageCode
Fin
```

```
Fonction : DonnerCaractere
Précondition : indice appartient à l'intervalle [0, 27]
Entrée :        - indice : entier
Local :         /
Sortie :        - caractère
Début
|       Si indice == 26 Alors
|       |       Retourner ' '
|       Sinon Si indice == 27 Alors
|       |       Retourner '.'
|       Sinon
|       |       Retourner Caractère(indice + ASCII('A'))       // conversion code ASCII → caractère
|       Fin Si
Fin
```

```
Procédure : RechercheDansGrille
Précondition : message ne contient que des caractères appartenant à l'ensemble {'A', ... , 'Z', ' ', '.'}
Entrée :        - grille : tableau 2D de alphabet*alphabet chaînes de caractères
                - message : chaîne de caractères
                - indice1 : entier (modifiable)
                - indice2 : entier (modifiable)
Local :         /
Sortie :        void
Début
|       Soit indice1 = -1
|       Soit indice2 = -1
|       Pour i de 0 à alphabet − 1 avec un pas de +1 Faire
|       |       Pour j de 0 à alphabet − 1 avec un pas de +1 Faire
|       |       |       Si grille[i, j] est égale à message Alors
|       |       |       |       Soit indice1 = i
|       |       |       |       Soit indice2 = j
|       |       |       |       Retourner       // quitter la procédure
|       |       |       Fin Si
|       |       Fin Pour
|       Fin Pour
Fin
```

```
Fonction : Decode2Gramme
Précondition : message ne contient que des triplets caractères existant dans grille
Entrée :        - grille : tableau 2D de alphabet*alphabet chaînes de caractères
                - message : chaîne de caractères
Local :         - indice1, indice2 : entier
Sortie :        - messageDecode : chaînes de caractères
Début
|       Soit messageDecode = ""
|       Soit indice1 = -1
|       Soit indice2 = -1
|       Pour i de 0 à Longueur(message) − 1 avec un pas de +3 Faire    // lecture des caractères 3 par 3
|       |       Soit msg = SousChaîne(message, i, i+3)          // sous-chaîne de l'indice i à l'indice i+3
|       |       RechercherDansGrille(grille, msg, indice1, indice2)       // modification de indice1 et indice2
|       |       Si indice1 ≠ -1 ET indice2 ≠ -1 Alors
|       |       |       Soit messageDecode = messageDecode   + DonnerCaractere(indice1)
|       |       |                                            + DonnerCaractere(indice2)
|       |       Fin Si
|       Fin Pour
|       Retourner messageDecode
Fin
```

```
Fonction : CaractereADroite
Précondition : c appartient à l'ensemble {'A', ... 'Z', ' ', '.'}
Entrée :        - c : caractère
Local :         /
Sortie :        - caractère
Début
|       Si c == '.' Alors
|       |       Retourner 'A'
|       Sinon Si c == ' ' Alors
|       |       Retourner '.'
|       Sinon Si c == 'Z' Alors
|       |       Retourner ' '
|       Sinon
|       |       Retourner Caractère(ASCII(c) + 1)
|       Fin Si
Fin
```

```
Fonction : CaractereAGauche
Précondition : c appartient à l'ensemble {'A', ... 'Z', ' ', '.'}
Entrée :        - c : caractère
Local :         /
Sortie :        - caractère
Début
|       Si c == 'A' Alors
|       |       Retourner '.'
|       Sinon Si c == '.' Alors
|       |       Retourner ' '
|       Sinon Si c == ' ' Alors
|       |       Retourner 'Z'
|       Sinon
|       |       Retourner Caractère(ASCII(c) - 1)
|       Fin Si
Fin
```

```
Fonction : DecalerCaractere
Précondition : c appartient à l'ensemble {'A', ... 'Z', ' ', '.'}
Entrée :        - c : caractère
                - offset : entier
Local :         /
Sortie :        - caractereDecale : caractère
Début
|       Soit caractereDecale = c
|       Si offset > 0 Alors
|       |       Pour i de 1 à offset avec un pas de +1 Faire
|       |       |       Soit caractereDecale = CaractereADroite(caractereDecale)
|       |       Fin Pour
|       Sinon Si offset < 0 Alors
|       |       Pour i de 0 à -offset avec un pas de +1 Faire
|       |       |       Soit caractereDecale = CaractereAGauche(caractereDecale)
|       |       Fin Pour
|       Fin Si
|       Retourner caractereDecale
Fin
```

```
Fonction : DecalerCaracteres
Précondition : message ne contient que des caractères appartenant à l'ensemble {'A', ... 'Z', ' ', '.'}
Entrée :        - message : chaîne de caractères
                - offset : entier
Local :         /
Sortie :        - chaineDecale : chaîne de caractères
Début
|       Soit chaineDecale = ""
|       Pour i de 0 à Longueur(message) − 1 avec un pas de +1 Faire
|       |       Soit chaineDecale = chaineDecale + DecalerCaractere(message[i], offset)
|       Fin Pour
|       Retourner chaineDecale
Fin
```

```
Fonction : CodeDeCesar
Précondition : message ne contient que des caractères appartenant à l'ensemble {'A', ... 'Z', ' ', '.'}
Entrée :        - message : chaîne de caractères
                - offset : entier
                - coder : booléen
Local :         /
Sortie :        - messageCode : chaîne de caractères
Début
|       Soit messageCode = ""
|       Si coder est VRAI Alors
|               Soit messageCode = DecalerCaracteres(message, clef)
|       Sinon
|               Soit messageCode = DecalerCaracteres(message, -clef)
|       Fin Si
|       Retourner messageCode
Fin
```

Procédure : Main
Entrée : void
Local :
- grille : tableau 2D de alphabet*alphabet chaînes de caractères
- alphabet : entier
- messageU : chaîne de caractères
- messageUCodeCesar, messageUDecodeCesar : chaîne de caractères
- messageUCode2Gramme, messageUDecode2Gramme : chaîne de caractères
- ligneDepart1, colonneDepart1, clefCesar : entier

Sortie : void
Début

```
|   Soit alphabet = 28        // {'A', ... 'Z', ' ', '.'} = 26 + 1 + 1 = 28 caractères
|   Soit ligneDepart1 = SaisirEntier()
|   Soit colonneDepart1 = SaisirEntier()
|   Soit clefCesar = (ligneDepart1 + colonneDepart1) % alphabet
|   Soit grille = CreerGrille(alphabet, ligneDepart1, colonneDepart1)
|   Soit messageU = SaisirChaîne()
|   // premier chiffrement : code de César
|   Soit messageUCodeCesar = CodeDeCesar(messageU, clefCesar, VRAI)
|   Afficher(messageUCodeCesar)
|   // second chiffrement : 2-gramme
|   Soit messageUCode2Gramme = Code2Gramme(grille, messageUCodeCesar)
|   Afficher(messageUCode2Gramme)
|   // premier déchiffrement : 2-gramme
|   Soit messageUDecode2Gramme = Decode2Gramme(grille, messageUCode2Gramme)
|   Afficher(messageUDecode2Gramme)
|   // second déchiffrement : code de César
|   Soit messageUDecodeCesar = CodeDeCesar(messageUDecode2Gramme, clefCesar, FALSE)
|   Afficher(messageUDecodeCesar)
|
```

Fin