Atelier 3

Objectifs pédagogiques :

- Assimilation du cycle de développement d'un logiciel notamment le **découpage modulaire**.
- Apprentissage du langage Pascal (programmation séparée des modules, modules internes et modules externes et/ou bibliothèque (s).

Problématique : Compression et Décompression d'un caractère.

Lors de la conception d'un logiciel de traitement d'images pour les caractères, chaque caractère est **représenté** par un tableau (matrice) **M [7,5]** à valeurs binaires (Gris = 1 et blanc = 0). Par exemple :

1. →	0	0	**1**	0	0
2. →	0	**1**	0	**1**	0
3. →	**1**	0	0	0	**1**
4. →	**1**	**1**	**1**	**1**	**1**
5. →	**1**	0	0	0	**1**
6. →	**1**	0	0	0	**1**
7. →	**1**	0	0	0	**1**

1. →	**1**	**1**	**1**	**1**	**1**
2. →	**1**	0	0	0	**1**
3. →	**1**	0	0	0	**1**
4. →	**1**	**1**	**1**	**1**	**1**
5. →	0	0	0	0	**1**
6. →	0	0	0	0	**1**
7. →	**1**	**1**	**1**	**1**	**1**

1. Le processus de **Compression** se présente comme suit :

A. Lecture des valeurs binaires de la Matrice **M [7,5].**

B. La matrice **M [7, 5]** du caractère est **convertie** en un vecteur **V** à valeurs binaires. Par exemple pour les deux caractères précédents on aura :

- Pour le **A** : V = **0 0 1 0 0 0** 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1
- Pour le **9** : V = **1 1 1 1 1 1 0 0 0** 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1

C. Le vecteur **V** est **compressé** en un vecteur **C** de manière suivante :

- C [1] = V [1] ;

Pour les autres éléments, ils présentent le nombre de **répétition d'une cellule binaire**. Par exemple pour les caractères précédents :

- Pour le **A** : C = **0 2 1 3** 1 1 1 1 1 3 7 3 2 3 2 3 1
- Pour le **9** : C = **1 6 3 2** 3 6 4 1 4 6

2. Le processus de **décompression** se fait selon **l'ordre inverse**.

Travail demandé :

Proposer une solution **modulaire** (avec justification) comportant tous les modules jugés nécessaires de lecture, de conversion, de compression, de décompression et d'affichage. **Prévoir un menu.**

NB :

- Afin de ne pas avoir à saisir à chaque fois le caractère de votre jeu d'essai une solution est envisageable, c'est celle de mettre dans un premier temps les valeurs du caractère au début de votre programme.
  Par exemple (pour le A) : M [1, 1] := 0 ; M [1, 2] := 0 ; M [1, 3] := 1 ; M [1, 4] := 0 ; M [1, 5] := 0 ; …… ; M [7,5] := 1 ;
- Si vous ne terminez pas la réalisation de TOUS les modules, mais seulement une partie : N'oubliez pas de mettre les petits programmes qui servent UNIQUEMENT à tester vos modules et faciliter la correction.
- Si vous utilisez des **modules déjà faits** : notez simplement leurs en-têtes et leurs rôles.

Conception Modulaire (/20 pts)

I. Partie : Justification modulaire et programme principal (4 pts)

1. Analyse Globale et justification modulaire /1 pt

Le but de ce TP c'est la compression et la décompression d'un caractère. Au début, le caractère est représenté par une matrice « M » binaire de taille 7*5. Afin de gagner en espace mémoire, la matrice « M » est compressée en un vecteur « C » de taille très réduite.

La première étape de la méthode consiste à rassembler toutes les lignes de la matrice par un vecteur V de 35 cases. La deuxième étape consiste à éliminer les répétitions en créant un autre vecteur C. La première case du vecteur C représente le contenu de la première case du vecteur V, qui sera suivi par le nombre de répétitions de cette valeur. A partir de cette case, ce n'est pas essentiel de sauvegarder les valeurs puisqu'on alterne uniquement entre deux valeurs 0 ou 1 (si nous avons terminé les occurrences d'une valeur, on rencontre les occurrences de l'autre valeur).

Découpage modulaire : Une simple lecture de l'énoncé suffit de faire ressortir les modules à construire ou à utiliser comme modules internes/externes ou à partir d'une bibliothèque (modules vus en cours et TD, Lect2D, Ecrire2D, Ecrire1D). Ainsi, nous avons besoin de concevoir et réaliser un programme englobant :

- La déclaration de tout l'environnement du TP (déclarations des variables et modules nécessaires du TP).
- Un programme principal présentant un menu des tâches qui seront effectuées par des appels à des modules internes ou externes (dans une bibliothèque éventuellement) dont certains sont traités en cours ou en TD dont les analyses et les algorithmes peuvent ne pas être repris dans la conception.
- Les modules nécessaires à la conception et réalisation du TP sont notamment le :
  - Module Lect2D qui réalise la lecture d'une matrice ou tableau 2D ;
  - Module Ecrire2D qui réalise l'écriture d'une matrice ou tableau 2D ;
  - Module Ecrire1D qui réalise l'écriture d'un tableau 1D ;
  - Module LectAffM qui permet de donner un choix à l'utilisateur entre la saisie manuelle de la matrice ou l'utilisation d'un jeu d'essai prédéfini ;
  - Module ConvMenV qui convertit la matrice en vecteur V ;
  - Module ConvVenC qui convertit le vecteur V en vecteur C ;
  - Module DConvCenV qui réalise la décompression de C en un vecteur V ;
  - Module ConvVenM qui réalise la conversion de V en une matrice M ;

2. Construction de l'algorithme principal : /3 pts (Analyse : 1pt, Algorithme : 2 pts)

A. Analyse /1 pt

Dans cette partie on présente un menu à l'utilisateur pour lui donner la possibilité de choisir entre :

---------- MENU ----------

1. Lecture et Affichage de la matrice M
2. Conversion de M [7, 5] en un vecteur V [35]
3. Compression de V [35] en un vecteur C
4. Décompression de C en un vecteur V
5. Conversion de V en une matrice M
6. Quitter

Afin de s'assurer que l'utilisateur respecte les étapes précédentes, il faut prévoir une variable booléenne pour chaque étape.

A titre d'exemple, pour la phase de compression :

– Si l'utilisateur a choisi de convertir la matrice M en un vecteur V, il faut vérifier que la matrice est déjà lue.

– Si l'utilisateur a choisi de compresser le vecteur V en un vecteur C, il faut vérifier que la matrice est déjà lue et elle est convertie en vecteur V.

Il en est de même pour la phase de décompression :

– Si l'utilisateur a choisi la décompression, il faut s'assurer, notamment, que la phase de conversion a été effectuée avec succès.

B. Algorithme / 2 pts

```
Algoritme CompDecomChar ; //TP2 1CP 2022-2023
//Uses Bib ; utilisation éventuellement d'une bibliothèque comportant certains modules utilisés dans le programme
Constantes      MaxLM = 7 ;
                MaxCM = 5;
                MaxV = 35;
                Max = 100;
Type            Tind = 1..Max;
                TindL = 1..MaxLM;
                TindC = 1..MaxCM;
                Tab1D = Tableau [Tind] d'entiers ;
                Tab2D = Tableau [Tindl, Tindc] d'entiers ;
Variables       M : Tab2D ;
                C,V : Tab1D ;
                TaiV, TaiC : Tind ;
                Nbl : TindL ;
                Nbc : TindC ;
                Ch, Ch1 : entier ;
                M1, M2, M3, M4, M5, T : Boolean ;
// autres variables utilisées dans le programme
// {$i modules} ; utilisation éventuellement de modules externes utilisées dans le programme
Et/Ou // Corps des modules internes utilisées dans le programme
...
// Lect2d .... (Vu en Cours et TD)
// Ecrire2D ... (Vu en Cours et TD)
// Ecrire1D ... (Vu en Cours et TD)
//-------------------------------------------------------------------------------------------------------------
PROCEDURE LectAffM (Var M : Tab2D; Var Nbl :Tindl; Var Nbc : Tindc);
//Lecture de la matrice M [7, 5]
// .... Corps de LectAffM
PROCEDURE ConvMenV (M : Tab2D ; Nbl : Tindl ; Nbc : TindC ; Var V : Tab1D ; Var TaiV : Tind) ;
// Convertir la matrice M [7, 5] en un vecteur V [35]
// ... Corps de ConvMenV
PROCEDURE ConvVenC (V:Tab1D; TaiV : Tind; Var C : Tab1D; Var TaiC : Tind) ;
//Compression de V [35] en un vecteur C
// ... Corps de ConvVenC
//-------------------------------------------------------------------------------------------------------------
Procedure DConvCenV (C : Tab1D ; TaiC : Tind ; Var V : Tab1D ; Var TaiV : Tind) ;
//DéCompression de C en un vecteur V
// ... Corps de DConvCenV
Procedure ConvVenM (V : Tab1D ; TaiV : Tind ; Var M : Tab2D ; Nbl : Tindl ; Nbc : Tindc) ;
// Convertir le vecteur V [35) en une matrice M [7, 5]
//... Corps de ConVen M
//-------------------------------------------------------------------------------------------------------------
Début  //Principal
   Ecrire ('---------- Réalisé par ..... G1 & ..... G5 ----------');
   Ecrire ('-------------------------- 2022-2023 ------------------------------');
   //Initialisation des variables booléennes relatives aux traitement des modules
    M1← Faux; M2← Faux; M3← Faux; M4← Faux ;
```

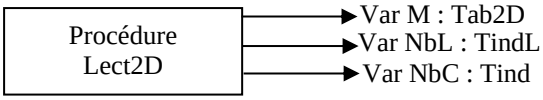
```
    Ecrire ('---------- Menu des modules ----------') ;
    Ecrire ('--------------------------------------------------------------------') ;
    Ecrire ('1. Lecture et Affichage de la matrice M [7,5] du caractère à compresser') ;
    Ecrire ('2. Conversion de M [7,5] en un vecteur V [35] avec affichage de V ') ;
    Ecrire ('3. Compression de V [35] en un vecteur C avec affichage de C') ;
    Ecrire ('4. DeCompression de C en un vecteur V avec affichage de V') ;
    Ecrire ('5. Conversion de V en une matrice M avec affichage de M') ;
    Ecrire ('6. Quitter') ;
    Ecrire ('Votre Choix ? ') ;
    Lire (Ch) ;
   SelonQue Ch Vaut
            1: Début //1
                  M1← Vrai ;
                  LectAffM (M, Nbl, Nbc) ;
               Fin ;  //1
             //-------------------------------------------------------------------------
            2: Début //2
                 SI (M1 = Faux) Alors Ecrire ('Matrice M non lue')
                   SINON
                      Début
                         M2←  Vrai ;
                         ConvMenV (M, Nbl, Nbc, V, TaiV) ;
                       Fin ;
              Fin ; //2
             //-------------------------------------------------------------------------
             3: Début //3
                   M3← Vrai ;
                   T ←  (M1 ET M2) ;
                   SI (T = Faux) Alors Ecrire ('Matrice M non lue OU non convertie en V')
                      SINON ConvVenC (V, TaiV, C, TaiC) ;
               Fin ; //3
             //-------------------------------------------------------------------------
             4: Début //4
                   M4 ← Vrai ;
                   T←  (M1 ET M2 ET M3);
                   SI (T = Faux) Alors Ecrire ('M, V et C non disponibles')
                     SINON DConvCenV (C, TaiC, V, TaiV) ;
                Fin ; //4
             //-------------------------------------------------------------------------
              5: Début //5
                    M5 ←  Vrai ;
                    T←  (M1 ET M2 ET M3 ET M4);
                    SI (T = Faux) Alors Ecrire ('M, V et C non disponibles')
                      SINON ConvVenM (V, TaiV, M, Nbl, Nbc) ;
                 Fin ; //5
             //-------------------------------------------------------------------------
             6:  Ecrire ('Fin du programme')
        SINON Ecrire ('Erreur de choix') ;
    Fin ; //SelonQue
Fin. //1
```

I. Partie : Compression (9 pts)

3. Construction des modules

4.1/ Module Lect2D

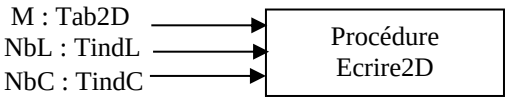
On aura besoin d'utiliser un module Lect2D pour lire la matrice M en entrée. (**Vu en cours et en TD)**



Rôle : permet de lire et remplir un tableau à 2 dimension M de 'NbL' lignes et 'NbC' colonnes

4.2/ Module Ecrire2D

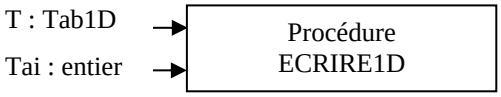
On aura besoin d'utiliser un module Ecrire2D pour Afficher la matrice M en entrée. (**Vu en cours et en TD**)



Rôle : : affiche le tableau M à 2 dimension de taille 'NbL' lignes et 'NbC' colonnes

4.3/ Module Ecrire1D

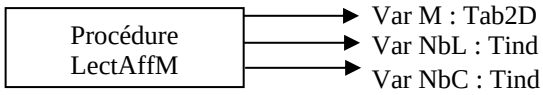
On aura besoin d'utiliser un module Ecrire1D pour Afficher la vecteur V et C. (**Vu en cours et en TD**)



Rôle : Ecrit les Tai éléments du tableau à une dimension T

4.4/ Module LectAffM (3 pts)

A. Analyse : On aura besoin d'utiliser un module LectAffM pour lire la matrice M en entrée. Elle permet de donner un choix à l'utilisateur entre la saisie manuelle de la matrice ou l'utilisation d'un jeu d'essai prédéfini.



Rôle : permet de lire M [7, 5] de valeurs '0' ou '1'

B. Algorithme

```
Procedure LectAffM (Var M : Tab2D; Var Nbl :Tindl; Var Nbc : Tindc);
//Lecture de la matrice M [7, 5]
Debut //LectAffM
    Ecrire ('Lecture et Affichage de la matrice M [7,5] du caractère à compresser') ;
    Ecrire ('1. ------> Lecture de la matrice') ;
    Ecrire ('2. ------> Jeu essai prédéfini') ;
    Ecrire ('Choix de lecture ? ');
    Lire (Chl);
    SelonQue Chl Vaut
        1: Debut // Avec lecture de la matrice M
              Lect2D (M, Nbl, Nbc) ;
              Ecrire ('Valeurs de la matrice') ;
              Ecrire2D (M, Nbl, Nbc) ;
           Fin ;
        2: Debut // Avec jeu essai prédéfini : Caractere A
              M [1,1] ← 0 ; M [1,2] ← 0 ; M [1,3] ← 1 ; M [1,4] ← 0 ; M [1,5] ←  0 ;
              M [2,1] ← 0 ; M [2,2] ← 1 ; M [2,3] ← 0 ; M [2,4] ← 1;  M [2,5] ←  0 ;
              …..
              M [7,1] ← 1;  M [7,2] ← 0 ; M [7,3] ← 0 ; M [7,4] ← 0 ; M [7,5] ← 1 ;
              Ecrire2D (M, Nbl, Nbc) ;
           Fin
           SiNon Ecrire ('Erreur de choix') ;
     FinSelon ;
Fin ; //LectAffM
```

4.5/ Module ConvMenV (3 pts)

Rôle : Convertir la matrice M [7, 5] en un vecteur V [35]

A. Analyse

- Le but du module c'est de convertir la matrice M en un vecteur V.
- Puisqu'on rassemble ligne par ligne, donc la ligne i est insérée à partir de l'indice (i-1)*nombre_colonnes.
- Pour retrouver la position d'une SelonQue M [i, j], il faut faire la somme entre l'indice du début de la ligne et j.
- Puisqu'on connait l'indice d'insertion de chaque élément de la matrice, il suffit de la parcourir afin d'insérer chaque élément à sa bonne position dans V.

B. Algorithme

```
Procedure ConvMenV (M : Tab2D ; Nbl : Tindl ; Nbc : TindC ; Var V : Tab1D ; Var TaiV : Tind) ;
// Convertir la matrice M [7, 5] en un vecteur V [35]
Variables i, j, k : Entier ;
DEBUT
    Pour i allant de 1 à Nbl faire
      Pour j allant de 1 à Nbc faire
            DPOUR
               K ← (i * Nbc) + j ;
               V [ k ] ← M [i, j] ;
            FPOUR ;
  TaiV ← Nbl * Nbc ;
FIN ;
```

2.5/ Module ConvVenC (3 pts)

Rôle : Compression de V [35] en un vecteur C

A. Analyse

- Le but du module c'est de réduire la taille du vecteur V.
- On commence par initialiser la première SelonQue du vecteur C à la valeur de la première SelonQue du vecteur V.
- Aussi, il faut initialiser l'indice d'insertion ' P ' à 2 puisqu'on a déjà inséré dans la première SelonQue et initialiser le nombre de répétitions ' k ' à 1.
- Ensuite, on fait parcourir le vecteur V de la première SelonQue à l'avant dernière, à chaque itération i :
  - On vérifie l'égalité entre la valeur de la SelonQue i et la suivante afin d'incrémenter le compteur de répétitions et de le sauvegarder à la position p.
  - Dans le cas contraire, on rencontre la première occurrence de la valeur opposée, donc on incrémente l'indice P et on initialise cette SelonQue à 1.
  - Avant de passer à la prochaine itération, il faut initialiser le nombre de répétitions à 1.

B. Algorithme

```
Procedure ConvVenC (V : Tab1D; TaiV : Tind ; Var C : Tab1D ; Var TaiC : Tind) ;
//Compression de V [35] en un vecteur C
Variables k, p, i: Entier ;
DEBUT // ConvVenC
    C [1] ← V [1];
    p ← 2 ;
    k ← 1 ;
    Pour i allant de 1 à Taiv-1 faire
        DPOUR
          SI (V [i] = V [i+1]) alors
             Debut
               k ← k + 1 ;
               C [p] ← k ;
             Fin
           SINON
             Debut
               k ← 1 ;
               p ← p+1 ;
               C [p] ← 1 ;
             Fin ;
        FPOUR ;
    TaiC ← p ;
FIN ; // ConvVenC
```

II. Partie : Décompression (7 pts)

2.6/ Module DConvCenV (4 pts)

Rôle : Décompression de C en un vecteur V

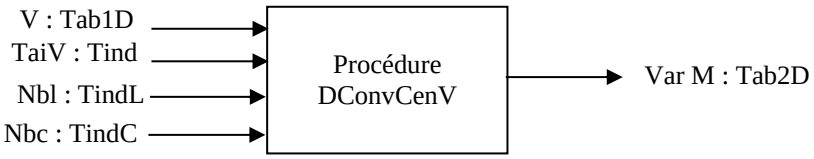
A. Analyse

- On commence par initialiser ‘k’ un indice d’insertion dans le vecteur V à 1.
- On a besoin aussi d’une autre variable ‘b’ pour sauvegarder la valeur à insérer, elle est initialisée à la première valeur du vecteur C.
- On fait varier un compteur de 2 à la taille du vecteur C (TaiC) afin de le parcourir, à chaque itération i :
  - On récupère le nombre de répétions ‘RP’ qui représente le contenue de la SelonQue i du vecteur C ;
  - On fait varier un autre compteur J de 1 à RP et à chaque itération ;
    - On insère la valeur de la variable b dans le tableau V à l’indice k ;
    - Ensuite on incrémente k.
- Afin d’alterner vers la prochaine valeur à insérer on soustrait b du 1.

B. Algorithme

```
Procedure DConvCenV (C : Tab1D ; TaiC : Tind ; Var V : Tab1D ; Var TaiV : Tind) ;
//DéCompression de C en un vecteur V
Variables i, j, b, k, rp : Entier ;;
DEBUT // DConvCenV
        k ← 1 ;
        b ←c[1] ;
        POUR i allant de 2 à  TaiC faire
        DPOUR
           RP ←  c [i] ;
           POUR   j allant de 1 à RP faire
              DPOUR
                 V [k] ←  b ;
                 k ← k + 1 ;
              FPOUR ;
           b ← 1- b ;
        FPOUR ;
  TaiV ←  k ;
Fin ; // DConvCenV
```

2.7/ Module ConvVenM (3 pts)



A. Analyse

Rôle : Convertir le vecteur V [35] en une matrice M [7, 5]

- On commence par initialiser ‘k’ un indice de parcours du vecteur V à 1.
- On parcourt la matrice M :
  - Chaque élément de la matrice reçoit la valeur de la SelonQue k du vecteur V.
  - Ensuite on incrémente l’indice ‘k’ afin de passer au prochain élément.

B. Algorithme

```
Procedure ConvVenM (V : Tab1D ; TaiV : Tind ; Var M : Tab2D ; Nbl : Tindl ; Nbc : Tindc) ;
// Convertir le vecteur V [35) en une matrice M [7, 5]
Variables i, j, k : Entier
DEBUT
   k ← 1 ;
   POUR i allant de 1 à Nbl faire
      POUR j allant de 1 à Nbc faire
        DPOUR
            M [i , j] ← V [ k ] ;
            k ← k + 1 ;
        FPOUR ;
Fin ;
```