TD4 – éléments de corrigé (algorithmes)

Exercice 4

Algorithme 1 : DefinirTableauConversionANSIRGB
Principe : Créer un tableau 2D de 16 par 4 entiers. Chaque ligne est une entrée de conversion, la première colonne contient le code ANSI, les trois suivantes les composantes R, G, et B correspondantes.
Entrée : /
Local : /
Sortie : tab : tableau 2D de 16 * 4 entiers
Début

```
|   Soit tab = tableau 2D de 16 * 4 entiers initialisé avec {
|       {30, 0, 0, 0},
|       {31, 255, 0, 0},
|       {32, 0, 255, 0},
|       {33, 255, 255, 0},
|       {34, 0, 0, 255},
|       {35, 255, 0, 255},
|       {36, 0, 255, 255},
|       {37, 255, 255, 255},
|       {90, 85, 85, 85},
|       {91, 255, 85, 85},
|       {92, 85, 255, 85},
|       {93, 255, 255, 85},
|       {94, 85, 85, 255},
|       {95, 255, 85, 255},
|       {96, 85, 255, 255},
|       {97, 255, 255, 255}
|   }
|   Retourner tab
```

Fin

Algorithme 2 : AfficherValeurPixelRGB
Principe : Parcourir l'image d'entrée (lignes i et colonnes j) et pour chaque case, parcourir le tableau de conversion en regardant la première colonne. Si le code dans la première colonne du tableau de conversion correspond à la valeur dans la case i,j de l'image, afficher les colonnes 2, 3, et 4 de cette ligne (du tableau de conversion).

Entrée :
- image : tableau 2D de hauteurImage * largeurImage entiers
- tabConversion : tableau 2D de hauteurTab * largeurTab entiers
- hauteurImage : entier
- largeuImager : entier
- hauteurTab : entier
- largeurTab : entier

Local :
- i : entier
- j : entier
- k : entier
- r : entier
- g : entier
- b : entier
- trouve : booléen

Sortie : void

```
Début
|       Pour i de 0 à hauteurImage − 1 avec un pas de +1 Faire
|       |       Pour j de 0 à largeurImage − 1 avec un pas de +1 Faire
|       |       |       // parcours de la table de conversion jusqu'à trouver ou non le code ANSI
|       |       |       Soit trouve = faux
|       |       |       Soit k = 0
|       |       |       Soit r = -1
|       |       |       Soit g = -1
|       |       |       Soit b = -1
|       |       |       Tant Que k < hauteurTab OU trouve == faux Faire
|       |       |       |       Si tabConversion[k,0] == image[i,j] Alors
|       |       |       |       |       r = tabConversion[k,1]
|       |       |       |       |       g = tabConversion[k,2]
|       |       |       |       |       b = tabConversion[k,3]
|       |       |       |       |       trouve = vrai
|       |       |       |       Fin Si
|       |       |       |       k = k + 1
|       |       |       Fin Tant Que
|       |       |       AfficherChaîne("({r},{g},{b})") // soit les bonnes valeurs, soit -1,-1,-1 si non trouvé
|       |       Fin Pour
|       |       AfficherCaractère('\n')
|       Fin Pour
Fin
```

Algorithme 3 : ConvertirImageANSIEnImageRGB
Principe : Créer une image 3D de même hauteur et largeur que l'image d'entrée mais avec un troisième dimensions à 3 (R,G,B). Parcourir l'image d'entrée (lignes i et colonnes j) et pour chaque case, parcourir le tableau de conversion en regardant la première colonne. Si le code dans la première colonne du tableau de conversion correspond à la valeur dans la case i,j de l'image, mettre les valeurs des colonnes 2, 3, et 4 dans l'image vide à la position (i,j,0..2).
Précondition : toutes les valeurs de image sont des codes ANSI valides !

Entrée :
- image : tableau 2D de hauteurImage * largeurImage entiers
- tabConversion : tableau 2D de hauteurTab * largeurTab entiers
- hauteurImage : entier
- largeuImager : entier
- hauteurTab : entier
- largeurTab : entier

Local :
- i : entier
- j : entier
- k : entier
- r : entier
- g : entier
- b : entier
- trouve : booléen

Sortie : imageRGB : tableau 3D de hauteur * largeur * 3 entiers

```
Début
|       Soit imageRGB = tableau 3D de hauteur * largeur * 3 entiers
|       Pour i de 0 à hauteur − 1 avec un pas de +1 Faire
|       |       Pour j de 0 à largeur − 1 avec un pas de +1 Faire
|       |       |       // parcours de la table de conversion jusqu'à trouver ou non le code ANSI
|       |       |       Soit trouve = faux
|       |       |       Soit k = 0
|       |       |       Soit r = -1
|       |       |       Soit g = -1
|       |       |       Soit b = -1
|       |       |       Tant Que k < hauteurTab OU trouve == faux Faire
|       |       |       |       Si tabConversion[k,0] == image[i,j] Alors
|       |       |       |       |       r = tabConversion[k,1]
|       |       |       |       |       g = tabConversion[k,2]
|       |       |       |       |       b = tabConversion[k,3]
|       |       |       |       |       trouve = vrai
|       |       |       |       Fin Si
|       |       |       |       k = k + 1
|       |       |       Fin Tant Que
|       |       |       image3D[i,j,0] = r
|       |       |       image3D[i,j,1] = g
|       |       |       image3D[i,j,2] = b
|       |       Fin Pour
|       Fin Pour
|       Retourner imageRGB
Fin
```

Algorithme 4 : ConvertirImageRGBEnImageANSI
Principe : Créer une image 2D de même hauteur et largeur que l'image d'entrée. Parcourir l'image d'entrée (lignes i et colonnes j) et pour chaque case, parcourir le tableau de conversion en regardant les colonnes 2, 3 et 4. Si les trois valeurs correspondent aux trois valeurs de l'image d'entrée en (i,j,0..2) alors mettre dans l'image vide le code correspondant dans la table en colonne 1.
Précondition : toutes les valeurs de image sont des codes RGB donnant des codes ANSI valides !

Entrée :
- image : tableau 3D de hauteurImage * largeurImage * 3 entiers
- tabConversion : tableau 2D de hauteurTab * largeurTab entiers
- hauteurImage : entier
- largeuImager : entier
- hauteurTab : entier
- largeurTab : entier

Local :
- i : entier
- j : entier
- k : entier
- code : entier
- trouve : booléen

Sortie : imageANSI : tableau 2D de hauteur * largeur entiers

```
Début
|       Soit imageANSI = tableau 2D de hauteur * largeur entiers
|       Pour i de 0 à hauteurImage − 1 avec un pas de +1 Faire
|       |       Pour j de 0 à largeurImage − 1 avec un pas de +1 Faire
|       |       |       // parcours de la table de conversion jusqu'à trouver ou non le code RGB
|       |       |       Soit trouve = faux
|       |       |       Soit k = 0
|       |       |       Soit code = -1
|       |       |       Tant Que k < hauteurTab OU trouve == faux Faire
|       |       |       |       Si tabConversion[k,1] == image[i,j,0] ET
|       |       |       |       |   tabConversion[k,2] == image[i,j,1] ET
|       |       |       |       |   tabConversion[k,3] == image[i,j,2] Alors
|       |       |       |       |           code = tabConversion[k,0]
|       |       |       |       |           trouve = vrai
|       |       |       |       Fin Si
|       |       |       |       k = k + 1
|       |       |       Fin Tant Que
|       |       |       imageANSI[i,j] = code
|       |       Fin Pour
|       Fin Pour
|       Retourner imageANSI
Fin
```