Raffinages des programmes pour le projet 2021-2022 de Programmation Impérative

Clément Delmaire-Sizes et Priscilia Gonthier

Groupe MN02

I. Modules utilisés

1. Unbounded_String

Nous utilisons dans nos programmes et modules, les types, procédures et fonctions ci-dessous qui sont présents dans le module Unbounded_String d'ADA:

TYPE Unbounded_String
procedure Delete(Source : in out Unbounded_String ; From : in Entier ; Trough : in Entier)
fonction Length(Source : in Unbounded_String) retourne Entier
fonction To_Unbounded_String(Source : in Chaîne de caractère) retourne Unbounded_String
fonction To_String(Source : in Unbounded_String) retourne Chaîne de caractère
function Element (Source : **in** Unbounded_String; Index : **in** Entier) **retourne** Caractère
function Unbounded_Slice (Source : **in** Unbounded_String; Low : **in** Entier; High : **in** Entier)
retourne Unbounded_String;

2. AB

```
TYPE T_AB EST PRIVÉ LIMITÉ
TYPE T_Char EST NEW Entier
CAPACITE : Entier = 257

-- Initialiser un arbre. L'arbre est vide
procédure Initialiser_AB(AB : out T_AB)

-- Est-ce qu'un Arbre est vide ?
fonction Est_Vide_AB(AB : in T_AB) retourne Booléen

-- Enregistrer un caractère associé à sa fréquence dans un arbre.
procédure Enregistrer_AB(AB: in out T_AB; Char: T_Char; Frequence: Entier)

-- Donner la racine de l'arbre donné en paramètre
fonction Caractere_Racine(AB: in T_AB) retourne T_Char

-- Fusionner 2 sous-arbres
procédure Fusionner_AB(AB_cree: out T_AB; AB_Gauche: in T_AB; AB_Droit: in T_AB)

-- Extraire le sous-arbre gauche d'un Arbre
procedure Sous_Arbre_Gauche(AB_g : out T_AB; AB: in T_AB)

-- Extraire le sous-arbre droit d'un Arbre
procedure Sous_Arbre_Droit(AB_d : out T_AB; AB: in T_AB)

-- Afficher l'arbre de Huffman comme la figure 20 p 11 de l'énoncé
procédure Afficher_AB(AB : in T_AB)
        (Raffinage de cette procédure en Annexe)

-- Créer un Arbre de Huffman sous la forme d'une suite de 0 et de 1, correspondant au parcours infixe de l'arbre
fonction Code_AB(AB : in T_AB) retourne Unbounded_String
        (Raffinage de cette fonction en Annexe)

-- Copier un arbre AB dans un autre arbre AB_Copie
procedure Copier_AB(AB : in T_AB; AB_Copie : in out T_AB)
```

```
TYPE T_TAB_CAR EST ENREGISTREMENT
        Tableau: TABLEAU(1..CAPACITE) DE T_Char
        Taille: Entier
FIN ENREGISTREMENT

-- Créer l'arbre de Huffman correspondant au code de l'arbre de Huffman et de la table des caractères en paramètre
procédure Creer_AB_Huff(AB: in out T_AB; Tab_Car : in T_TAB_CAR; Code_AB : in out Unbounded_String)
        (Raffinage en Annexe)

-- Vider un arbre donné en paramètre
procédure Vider_AB(AB: T_AB)

PRIVÉ

TYPE T_Cellule

TYPE T_AB EST POINTEUR SUR T_Cellule

TYPE T_Cellule EST ENREGISTREMENT
        Fréquence : Entier
        Caractere : T_Char
        Arbre_Gauche : T_AB
        Arbre_Droit : T_AB
FIN ENREGISTREMENT
```

Annexe des raffinages de AB

Raffinage de la procédure Afficher_AB :
R0: Afficher l'arbre de Huffman comme la figure 20 p 11 de l'énoncé

R1: Comment "Afficher l'arbre de Huffman comme la figure 20 p 11 de l'énoncé"?
```
        procédure Afficher_AB(AB : in T_AB; mem_parcours : in Unbounded_String)
                (Raffinage ci-dessous)
        Afficher_AB(AB, To_Unbounded_String(""))
```

Raffinage de la sous-procédure Afficher_AB :
R0: Afficher l'arbre de Huffman comme la figure 20 p 11 de l'énoncé

R1: Comment "Afficher l'arbre de Huffman comme la figure 20 p 11 de l'énoncé"?
```
        mem←mem_parcours
        Si AB /= Rien Alors
                Afficher les fréquences et symboles à la bonne place
                Afficher les caractères s'il y en a
        Sinon
```

Rien
FinSi

R2: Comment “Afficher les fréquences et symboles à la bonne place”?
Si mem=”” **Alors**
Afficher(‘(‘)
Afficher(AB^.Frequence)
Afficher(‘)’)
Sinon
Afficher les espaces et barres

Si mem(1)=’1’ **Alors**
Afficher(“\--1--(“)
Sinon
Afficher(“\--0--(“)
FinSi
Afficher(AB^.Frequence)
Afficher(‘)’)
FinSi

R2: Comment “Afficher les caractères s’il y en a”?
Si AB^.Caractere=-2 **Alors**
Afficher_AB(AB^.Arbre_Gauche,mem_parcours & ’0’)
Afficher_AB(AB^.Arbre_Droit,mem_parcours & ’1’)
Sinon
Afficher le caractère
FinSi
R3: Comment “Afficher les espaces et les barres”?
Nouvelle ligne
Afficher(“ “)
TantQue Length(mem)>1 **Faire**
Si mem(1)=’1’ **Alors**
Afficher(“ “)
Sinon
Afficher(“| “)
FinSi
Delete(mem, 1, 1)
FinTQ

R3: Comment “Afficher le caractère”?
Afficher(“ ‘ “)
Si AB^.Caractere = -1 **Alors**
Afficher(“\$”)
SinonSi AB^.Caractere = Ord(fin de ligne) **Alors**

Afficher("\n")
Sinon
Afficher(Chr(AB^.Caractere))
FinSi
Afficher(" ' ")

Raffinage de la procédure Code_AB :

R0: Créer un Arbre de Huffman sous la forme d'une suite de 0 et de 1, correspondant au parcours infixe de l'arbre

R1: Comment "Créer un Arbre de Huffman sous la forme d'une suite de 0 et de 1, correspondant au parcours infixe de l'arbre"?

Si AB = Rien **Alors**
retourner To_Unbounded_String("")
SinonSi AB^.Caractere = -2 **Alors**
retourner To_Unbounded_String('0') + Code_AB(AB.Arbre_Gauche) + Code_AB(AB.Arbre_Droit)
Sinon
retourner To_Unbounded_String('1') + Code_AB(AB.Arbre_Gauche) + Code_AB(AB.Arbre_Droit)
FinSi

Raffinage de la procédure Creer_AB_Huff :

R0 : Créer l'arbre de Huffman correspondant au code de l'arbre de Huffman et de la table des caractères en paramètre

R1 : Comment "Créer l'arbre de Huffman correspondant au code de l'arbre de Huffman et de la table des caractères en paramètre"?

procédure Creer_AB_Huff(AB: in out T_AB; Tab_Car : in T_TAB_CAR; Code_AB : in out Unbounded_String; Indice: in out Entier)
Creer_AB_Huff(AB, Tab_Car, Code_AB, 1)

3. TAB_AB

AVEC AB UTILISER AB

TYPE T_TAB_AB EST PRIVÉ

-- Initialiser un tableau d'arbres binaires vide
procédure Initialiser(Tab: out T_TAB_AB)

-- Indiquer si le tableau est vide
fonction Est_Vide(Tab: in T_TAB_AB) retourne Booléen

-- Indiquer si un des arbres du tableau contient le caractère donné en paramètre
fonction Character_Present(Tab: in T_TAB_AB, Char: in T_Char) retourne Booléen

-- Donner l'indice du tableau où se trouve un caractère
fonction Indice_Character(Tab: in T_TAB_AB, Char: in T_Char) retourne Entier

-- Retourner la fréquence du caractère en paramètre dans un des arbres du tableau
fonction Frequence_Character(Tab: in T_TAB_AB, Char: in T_Char) retourne Entier

-- Retourner la fréquence de la racine de l'arbre à la position i dans le tableau
fonction Frequence_Indice(Tab: in T_TAB_AB, Indice: in Entier) retourne Entier

-- Supprimer du tableau l'arbre à la position donnée en paramètre
procédure Supprimer(Tab: in out T_TAB_AB, Position: in Entier)

-- Ajouter dans le tableau un arbre contenant seulement le caractère associé à la fréquence tous deux en paramètre
procédure Enregistrer(Tab: in out T_TAB_AB, Char: in T_Char, Frequence: in Entier)

-- Ajouter dans le tableau un arbre donné en paramètre
procédure Ajouter(Tab: in out T_TAB_AB; AB: in T_AB)

-- Renvoyer la taille du tableau en paramètre
fonction Taille(Tab: in T_TAB_AB) retourne Entier

-- Créer un arbre dont le sous-arbre gauche est l'arbre en position i1 du tableau et le sous-arbre droit est l'arbre en position i2, supprime les deux sous-arbres du tableau et place l'arbre créé dans le tableau
procédure Regrouper(Tab: in out T_TAB_AB, i1: in Entier, i2: in Entier)
(Raffinage de cette procédure en Annexe)

-- Retourne l'arbre de Tab situé à l'indice demandé
fonction Extraire(Tab: in T_TAB_AB, Indice: in Entier) retourne T_AB

PRIVÉ

TYPE T_TAB EST TABLEAU (1..CAPACITE) DE T_AB

TYPE T_TAB_AB EST ENREGISTREMENT
Tableau : T_TAB
Taille : Entier
FIN ENREGISTREMENT

Annexe des raffinages de TAB_AB

Raffinage de la procédure Regrouper :

R0: Créer un arbre dont le sous-arbre gauche est l'arbre en position i1 du tableau et le sous-arbre droit est l'arbre en position i2, supprime les deux sous-arbres du tableau et place l'arbre créé dans le tableau

R1: Comment "Créer un arbre dont le sous-arbre gauche est l'arbre en position i1 du tableau et le sous-arbre droit est l'arbre en position i2, supprime les deux sous-arbres du tableau et place l'arbre créé dans le tableau"?

```
AB_Gauche←Extraire(Tab, i1)
AB_Droit←Extraire(Tab, i2)
Fusionner_AB(AB_Tmp, AB_Gauche, AB_Droit)
Supprimer(Tab, i1)
Supprimer(Tab,i2)
Ajouter(Tab, AB_Tmp)
Vider_AB(AB_Tmp)
```

4. TAB_HUFF

AVAC AB UTILISER AB

TYPE T_Tab_Huff EST PRIVÉ

-- Initialiser une table de Huffman vide
procédure Initialiser(Tab_Huff: out T_Tab_Huff)

-- Donner la taille d'une table de Huffman
fonction Taille_Table(Tab_Huff: in T_Tab_Huff) retourne Entier

-- Déterminer si la table en paramètre est vide
fonction Est_Vide(Tab_Huff: in T_Tab_Huff) retourne Booléen

-- Créer une table de Huffman à partir d'un arbre de Huffman
procedure Creer_Table(Tab_Huff :in out T_Tab_Huff, AB: in T_AB)
(Raffinage de cette procédure en annexe)

-- Renvoyer le code dans la cellule associée à la case du tableau d'indice donné en paramètre
fonction Code_Indice(Tab_Huff :in out T_Tab_Huff, Indice : in Entier) retourne Unbounded_String

-- Renvoyer le caractère dans la cellule associée à la case du tableau d'indice donné en paramètre
fonction Caractere_Indice(Tab_Huff :in out T_Tab_Huff, Indice : in Entier) retourne T_Char

-- Renvoyer le code de Huffman du caractère passé en argument
fonction Table_Code(Tab_Huff : in T_Tab_Huff ; Char: in T_Char) retourne Unbounded_String

-- Afficher la table de Huffman donnée en paramètre comme la figure 21 p.11 de l'énoncé
procédure Afficher_Table(Tab_Huff : in T_Tab_Huff)

-- Renvoyer la position d'un caractère dans la table
fonction Table_Indice(Tab_Huff: in T_Tab_Huff, Char : in T_Char) retourne Entier

PRIVÉ

TYPE T_Cellule_Code EST ENREGISTREMENT
    Caractere: T_Char
    Code : Unbounded_String
FIN ENREGISTREMENT

TYPE T_Tab_Huff EST ENREGISTREMENT
    Tableau : TABLEAU (1..CAPACITE) DE T_Cellule_Code
    Taille : Entier
FIN ENREGISTREMENT

Annexe des raffinages de TAB_HUFF

Raffinage de la sous-procédure Creer_Table :
R0: Créer une table de Huffman à partir d'un arbre de Huffman

R1: Comment "Créer une table de Huffman à partir d'un arbre de Huffman"?
    procedure Creer_Table(Tab_Huff :in out T_Tab_Huff, AB: in T_AB, Code : in Unbounded_String)
        (Raffinage ci-dessous)
    Creer_Table(Tab_Huff, AB, To_Unbounded_String(""))

Raffinage de la sous-procédure Creer_Table :
R0: Créer une table de Huffman à partir d'un arbre de Huffman

R1: Comment "Créer une table de Huffman à partir d'un arbre de Huffman"?
Si Non Est_Vide(AB) **Alors**
    Si Caractere_Racine(AB) /= '' **Alors**
        Tab_Huff.Taille ← Tab_Huff.Taille + 1
        Tab_Huff.Tableau(Taille).Caractere ← Caractere_Racine(AB)
        Tab_Huff.Tableau(Taille).Code ← Code
    Sinon
        Creer_table(Tab_Huff, Sous_Arbre_Gauche(AB), Code + '0')
        Creer_table(Tab_Huff, Sous_Arbre_Droit(AB), Code + '1')
    FinSi
FinSi

5. OCTET

TYPE T_Octet EST modulo 2 ** 8

-- Convertir un octet en Unbounded_String.

fonction To_Unbounded_String(Octet : in T_Octet) retourne Unbounded_String

-- Convertir un Unbounded_String en octet.

fonction To_Octet(Source : in Unbounded_String) retourne T_Octet

II. Raffinages des Programmes

1. Programme compresser

R0: Compresser un fichier Texte à l'aide de l'algorithme de Huffman.

R1: Comment "Compresser un fichier Texte à l'aide de l'algorithme de Huffman"?

Faire l'interface (transcription du cahier des charges)
Déterminer si l'option -b ou – bavard est mise
Effectuer la compression

Affichage: out Booléen, Num_Arg_Nom : out Entier

R2: Comment "Effectuer la compression"?

TantQue Num_Arg_Nom <= Argument_Count **Faire**
Créer l'arbre de Huffman. — Tableau: out T_TAB_AB, Arbre_B: out T_AB, Num_Arg_Nom : in
Créer la table de Huffman. — Tab_Huff: out T_Tab_Huff
Si Affichage **Alors**
Afficher_AB(Arbre_B)
Afficher_Table(Tab_Huff)
Sinon
Rien
FinSi
Créer le fichier compressé en format .hff. — Tab_Huff: in
Vider les arbres et tableaux d'arbres.
Num_Arg_Nom <--Num_Arg_Nom + 1
FinTQ

R2: Comment "Déterminer si l'option -b ou --bavard est mise"?

Si Argument_Count = 0 **Alors**
Lever l'exception Command_error
SinonSi Argument_Count >= 2 **EtAlors (**Argument(1) = "-b" or else Argument(1) = "--bavard") **Alors**
Affichage ←Vrai
Num_Arg_Nom ← 2

Sinon
Affichage ← Faux
Num_Arg_Nom ← 1
FinSi

R3: Comment “Créer l’arbre de Huffman”?
Ouvrir le fichier texte en lecture seule.
Enregistrer les fréquences dans un tableau d'arbre en prenant en compte le caractère \$.
Tableau: out
Fermer le fichier texte.
Regrouper les arbres du tableau pour obtenir un arbre de Huffman.
Tableau: in out
Arbre_B: out

R3: Comment “Créer la table de Huffman”?
Initialiser(Tab_Huff)
Creer_Table(Tab_Huff, Arbre_B)

R3: Comment “Créer le fichier compressé en format .hff.”?
Ouvrir un fichier du même nom .hff en écriture.
Nom_Fichier_Texte : out Unbounded_String
Insérer la liste des symboles.
Tab_Huff: in
Insérer le code de l’arbre.
Arbre_B: in, Code_Inserer: out Unbounded_Sring
Insérer le texte codé.
Tab_Huff: in, Code_Inserer: in out
Fermer le fichier .hff

R3 : Comment “Vider les arbres et tableaux d’arbres”?
Vider(Tableau)
Vider_AB(Arbre_B)

R4: Comment “Enregistrer les fréquences dans un tableau d'arbre en prenant en compte le caractère \$.”?
Octet: T_Octet
Frequence: Entier
Char_Lu: T_Char
Initialiser(Tableau)
Si Fin du fichier texte **Alors**
Lever une erreur de fichier vide
FinSi
TantQue Non fin du fichier texte **Faire**
Octet ← Octet lu
Char_Lu ← Ord(Chr(Octet))
Si Caractere_Present(Tableau, Char_Lu) **Alors**
Frequence ← Frequence_Character(Tableau, Char_Lu)+1
Enregistrer(Tableau, Char_lu, Frequence)
Sinon
Enregistrer(Tableau, Char_lu, 1)

```
        FinSi
    FinTQ
    Enregistrer(Tableau, -1, 0)
```

R4: Comment “Regrouper les arbres du tableau pour obtenir un arbre de Huffman.” ?

```
    Taille_Tab <-- Taille(Tableau)
    TantQue Taille_Tab > 1 Faire
        Trouver les deux arbres dont la valeur de fréquence de la racine est la plus faible.
        Regrouper ces deux arbres.
    FinTQ
    Copier(Tableau, 1, Arbre_B)
```

Taille_Tab: Entier

Tableau: in, i_min_1: out Entier, i_min_2: out Entier, Taille_Tab : in

Tableau: in out, i_min_1: in, i_min_2: in

R4: Comment “Insérer la liste des symboles”?

```
    Indice ← Table_Indice(Tab_Huff, -1)
    Taille ← Taille_Table(Tab_Huff)
    Insérer Indice en octet dans le fichier

    Pour i De 1 À Taille Faire
        Si i /= Indice Alors
            Insérer Valeur(Caractere_Indice(Tab_Huff, i)) en octet
        Sinon
            Rien
        FinSi
    FinPour
    Insérer Valeur(Caractere_Indice(Tab_Huff, Taille)) en octet
```

Indice: Entier
Taille: Entier

R4: Comment “Insérer le code de l’arbre”?

```
    Code_Arbre← Code_AB(Arbre_B)
    Code_Inserer← To_Unbounded_String(“”)
    Pour i De 1 À Length(Code_Arbre) Faire
        Code_Inserer← Code_Inserer & Element(Code_Arbre, i)
        Si Length(Code_Inserer)=8 Alors
            Inserer Code_Inserer en octet
            Code_Inserer← ‘To_Unbounded_String(“”)
        Sinon
            Rien
        FinSi
    FinPour
```

Code_Arbre : Unbounded_String,
Code_Inserer : Unbounded_String

R4: Comment “Insérer le texte codé.”?

```
    Ouvrir le fichier texte en lecture seule
    TantQue Non fin du fichier texte Faire
```

Octet: T_Octet
Char_Lu: Caractère

```
        Octet ← Octet lu
        Char_Lu←Ord(Chr(Octet))
        Code_Inserer ← Code_Inserer & Table_Code(Tab_Huff, Char_Lu)
        Si Length(Code_Inserer)=8 Alors
            Inserer To_Octet(Code_Inserer)
            Code_Inserer← To_Unbounded_String("")
        SinonSi Length (Code_Inserer) > 8 Alors
            Inserer To_Octet(Unbounded_Slice(Code_Inserer, 1, 8))
            Code_Inserer← Unbounded_Slice(Code_Inserer, 9, Length(Code_Inserer))
        Sinon
            Rien
        FinSi
    FinTQ
    Insérer le caractère fin de texte
    Insérer le dernier Octet
    Fermer le fichier texte
```

Code_Inserer: in out

R5: Comment “Trouver les deux arbres dont la valeur de fréquence de la racine est la plus faible.” ?

```
i_min_1 ← 1
i_min_2 ← 2
Pour i De 3 À Taille_Tab Faire
    Si Frequence_Indice(Tableau, i) < max (Frequence_Indice(Tableau, i_min_1) , Frequence_Indice(Tableau,
    i_min_2)) Alors
        Si Frequence_Indice(Tableau, i_min_1) > Frequence_Indice(Tableau, i_min_2)   Alors
            i_min_1 ← i
        Sinon
            i_min_2 ← i
        FinSi
    Sinon
        Rien
    FinSi
FinPour
```

R5: Comment “Regrouper ces deux arbres.”?

```
Si Frequence_Indice(Tableau, i_min_1) < Frequence_Indice(Tableau, i_min_2) Alors
    Regrouper(Tableau, i_min_1, i_min_2)
Sinon
    Regrouper(Tableau, i_min_2, i_min_1)
FinSi
```

R5: Comment “Insérer le caractère fin de texte”?

```
Code_Inserer <-- Code_Inserer & Code_Indice(Tab_Huff, Indice)
Si Length(Code_Inserer)=8 Alors
    Inserer To_Octet(Code_Inserer)
```

```
        Code_Inserer← To_Unbounded_String("")
    SinonSi Length (Code_Inserer) > 8 Alors
        Inserer To_Octet(Unbounded_Slice(Code_Inserer, 1, 8))
        Code_Inserer← Unbounded_Slice(Code_Inserer, 9, Length(Code_Inserer))
    Sinon
        Rien
    FinSi
```

R5: Comment “Insérer le dernier octet”?

```
    Si Code_Inserer/= To_Unbounded_String(“”) Alors
        Pour i De Length(Code_Inserer) À 7 Faire
            Code_Inserer← Code_Inserer & “0”
        FinPour
        Inserer To_Octet(Code_Inserer)
    Sinon
        Rien
    FinSi
```

1.1) Tableau d’évaluation Compresser

		Evaluation (I/P/A/+)
Forme (D-21)	Respect de la syntaxe Ri : Comment "... une action complexe ..." ? des actions combinées avec des structures de controle Rj : ...	+
	Verbes à l'infinitif pour les actions complexes	+
	Noms ou équivalent pour expressions complexes	A
	Tous les Ri sont écrits contre la marge et espacés	+
	Les flots de données sont définis	+
	Une seule décision ou répétition par raffinage	A
	Pas trop d'actions dans un raffinage (moins de 5 ou 6)	A
	Bonne présentation des structures de contrôle	+
Fond (D21-D22)	Le vocabulaire est précis	+
	Le raffinage d'une action décrit complètement cette action	+
	Le raffinage d'une action ne décrit que cette action	+
	Les flots de données sont cohérents	+
	Pas de structure de contrôle déguisée	+

2. Programme décompresser

R0: Décompresser un fichier Texte, codé à l'aide de l'algorithme de Huffman.

R1: Comment "Décompresser un fichier Texte, codé à l'aide de l'algorithme de Huffman"?

Faire l'interface (transcription du cahier des charges)
Ouvrir le fichier compressé
Recréer la liste des caractères du texte dans leur ordre d'apparition
lors d'un parcours infixe de l'arbre de Huffman
Recréer l'arbre de Huffman à partir de la liste précédente et de l'arbre codé dans le fichier

Tab_Caract: in
Arb_Huff : out T_AB
Code: out Unbounded_String

Créer un fichier Texte vide
Afficher l'arbre si l'option bavard a été utilisée
Ajouter dans ce fichier les caractères décodés à l'aide de l'arbre de Huffman

Code: in out, Tab_Huff: out T_Tab_Huff, Arb_huff: in

Fermer le fichier texte
Fermer le fichier compressé

R2: Comment "Recréer la liste des caractères du texte dans leur ordre d'apparition lors d'un parcours infixe de l'arbre de Huffman" ?

Fin_Tableau: Booléen

Tab_Caract.Taille ← 0
octet_fin ← premier octet
octet_lu ← Chr(octet lu)

Répéter
Rajouter un élément dans le tableau puis lire un octet

Tab_Caract : in out
octet_lu : out T_Char
Fin_Tableau: out

JusquÀ Fin_Tableau

R3: Comment "Rajouter un élément dans le tableau puis lire un octet" ?

Si Tab_Caract.Taille = octet_fin - 1 **Alors**
Tab_Caract.Taille ← Tab_Caract.Taille + 1
Tab_Caract.Tableau(Tab_Caract.Taille) ← -1
Tab_Caract.Taille ← Tab_Caract.Taille + 1
Tab_Caract.Tableau(Tab_Caract.Taille) ← octet_lu
Sinon
Tab_Caract.Taille ← Tab_Caract.Taille + 1
Tab_Caract.Tableau(Tab_Caract.Taille) ← octet_lu
FinSi

octet_lu ← octet lu converti en T_Char
Fin_Tableau ← (octet_lu = Tab_Caract.Tableau(Tab_Caract.Taille))

R2: Comment “Recréer l’arbre de Huffman à partir de la liste précédente et de l’arbre codé dans le fichier” ?

Enregistrer le code de l’arbre

Code_AB : out Unbounded_String,
Code: out Unbounded_String

Initialiser_AB(Arb_Huff)
Creer_AB_Huff(Arb_Huff,Tab_Caract,Code_AB)

R2: Comment “Ajouter dans ce fichier les caractères décodés à l’aide de l’arbre de Huffman ” ?

Initialiser(Tab_Huff)
Creer_Table(Tab_Huff, Arb_Huff)
Vider_AB(Arb_Huff)
Afficher la table de Huffman si l’option bavard a été utilisée
fin_texte ← false
code_tmp ← Code
Code ← To_Unbounded_String(“”)

fin_texte: Booléen
code_tmp : Unbounded_String

Pour i **De** 1 **à** Length(Code_tmp) **Faire**

Rechercher le code dans la table de Huffman et rajouter le caractère correspondant si un code correspond

code_lu: Unbounded_String
Tab_Huff: in out
Code: in out
fin_texte : out

FinPour

Continuer à ajouter dans ce fichier les caractères décodés à l’aide de l’arbre de Huffman

R3: Comment “Enregistrer le code de l’arbre”

nb_char : Entier

```
nb_char← 0
Code ← To_Unbounded_String(octet lu)
Code_AB ← To_Unbounded_String(“”)
TantQue nb_char<Tab_Char.Taille Faire
    Si Code= To_Unbounded_String(“”) Alors
        Code← To_Unbounded_String(octet lu)
    Sinon
        Code_AB← Code_AB+code(1)
        Si Code(1) = To_Unbounded_String(“1”) Alors
            nb_char ← nb_char + 1
        FinSi
        Delete(Code,1,1)
    FinSi
```

FinTQ

R3: Comment:"Rechercher le code dans la table de Huffman et rajouter le caractère correspondant si un code correspond" ?

```
code_trouve ← faux                                  code_trouve : Booléen
j ← 1                                               j : Entier
code ← code & Code_tmp(i)
TantQue (Non code_trouve EtAlors j<=Taille_table(tab_huff)) Faire
    Si Code = Code_indice(Tab_Huff, j) Alors
        Si  octet_fin = j Alors
            fin_texte ← Vrai
        Sinon
            Ajouter  Character'Val(Caractere_indice(Tab_huff, j)) dans le texte
            code_trouve ← Vrai
            Code ← To_Unbounded_String("")
        FinSi
    Sinon
        Rien
    FinSi
    j ← j + 1
FinTQ
```

R3 : Comment:"Continuer à ajouter dans ce fichier les caractères décodés à l'aide de l'arbre de Huffman" ?

```
TantQue non (fin du fichier) Faire
    Code_lu ← Unbounded_String(octet_lu)
    Pour i De 1 à 8 Faire
        Code ← Code & Code_lu(1)
        Delete(Code_lu,1,1)
        Rechercher le code dans la table de Huffman et rajouter le caractère correspondant
si un code correspond (pour le reste du texte)
    FinPour
FinTQ
```

R4: Comment:”Rechercher le code dans la table de Huffman et rajouter le caractère correspondant si un code correspond (pour le reste du texte)” ?

```
code_trouve ← faux
j ← 1
code ← code & Code_tmp(i)
TantQue (Non code_trouve EtAlors j<=Taille_table(tab_huff)) Faire
    Si Code = Code_indice(Tab_Huff, j) Alors
        Si octet_fin = j Alors
            fin_texte ← Vrai
        Sinon
            Ajouter  Character’Val(Caractere_indice(Tab_huff, j)) dans le texte
            code_trouve ← Vrai
            Code ← To_Unbounded_String(“”)
        FinSi
    Sinon
        Rien
    FinSi
    j ← j + 1
FinTQ
```

2.1) Tableau d’évaluation Décompresser

		Evaluation (I/P/A/+)
Forme (D-21)	Respect de la syntaxe Ri : Comment "... une action complexe ..." ? des actions combinées avec des structures de controle Rj : ...	+
	Verbes à l'infinitif pour les actions complexes	+
	Noms ou équivalent pour expressions complexes	A
	Tous les Ri sont écrits contre la marge et espacés	+
	Les flots de données sont définis	+
	Une seule décision ou répétition par raffinage	A
	Pas trop d'actions dans un raffinage (moins de 5 ou 6)	A
	Bonne présentation des structures de contrôle	+
Fond (D21-D22)	Le vocabulaire est précis	+
	Le raffinage d'une action décrit complètement cette action	+
	Le raffinage d'une action ne décrit que cette action	+
	Les flots de données sont cohérents	+
	Pas de structure de contrôle déguisée	+