Projet Codage de Huffman

FRANOT Marin
TESTU François

Types de données :

```
generic
        T_donnee
        capacite : integer

Type T_tableau is array (1..capacite) of T_donnee


type T_Octet is mod 2 ** 8;

  Type T_Noeud;
  Type T_Arbre is access T_Noeud;
  Type T_Noeud is
        record
                cle : Integer;
                donnee : integer;
                fils_g : T_Arbre;
                fils_d : T_Arbre;
        end record;

  Type T_Cellule;
  Type T_LCA is access T_Cellule;
  Type T_Cellule is
     record
        donnee : T_octet;
        suivant : T_LCA;
     end record;

package tab_arbre is new tableau (T_donnee => T_arbre, capacite => 257)

package tab_lca is new tableau (T_donnee => T_LCA, capacite => 257)

package tab_entier is new tableau (T_donnee => integer, capacite => 258)
```

Raffinages :

Exercice 1 compresser un fichier texte:

R0: Compresser un fichier texte

R1 : Comment " Compresser un fichier texte" ?

Construire tableau des fréquence tab_noeuds: out tab_arbre.T_tableau, taille: out integer
Construire l'arbre de Huffman tab_noeuds : in out, arbre : out T_arbre
Créer le tableau de huffman arbre : in, tab_huffman : out tab_LCA.T_Tableau
Créer le tableau infixe arbre : in, tab_infixe : out tab_entier.T_tableau
Structurer l'arbre de huffman arbre : in, structure_arbre : out T_LCA
Encoder fichier tab_infixe, structure_arbre, tab_huffman : in
Afficher les étapes en cas de bavard arbre, tab_huffman, structure_arbre : in

R2: Comment "Construire tableau des fréquences" ?

Initialiser (tab_noeuds) tab_noeuds : out tab_arbre.T_tableau
taille <- 0
ouvrir le fichier texte
Répéter
    taille <- taille + 1
    Récupérer un caractère val : out integer
    Enregistrer le caractère dans tab_noeuds tab_noeuds : in out, val : in
Jusqu'à la fin du texte
Fin Répéter
fermer le fichier texte
Ajouter le caractère de fin de fichier tab_noeuds : in out, taille : in out

R2: Comment "Construire l'arbre de Huffman" ?

Calculer indice_min de tab_noeuds tab_noeuds : in, indice : out integer
Initialiser arbre arbre : out T_arbre
Tant Que la_cle (tab_noeuds (indice)) < taille
    traiter la fusion du 1er et 2e min teb_noeuds, indice : in out
Fin Tant Que

R2 : Comment "creer le tableau de huffman"?

Initialiser(code) code : out T_LCA
Initialiser (tab_huffman) tab_huffman : out Tab_LCA.T_Tableau
Creer_tab_huffman (arbre, tab_huffman, code) tab_huffman, code: in out, arbre : in,

R2 : Comment "Créer le tableau infixe" ?

taille <- 0
creer le tableau infixe brut arbre : in, taille, tab_infixe : in out
trouver la position du symbole de fin de fichier tab_infixe : in, indice : out integer
echanger le symbole de fin de fichier avec le debut du tableau tab_infixe : in out, indice : in

R2 : Comment "Structurer l'arbre de huffman"?

Si est_feuille(arbre) **Alors**
ajouter (structure_arbre, 1);
Sinon
ajouter (structure_arbre, 0);
Structurer l'arbre de huffman arbre^.fils_g : in, structure_arbre : in out
Structurer l'arbre de huffman arbre^.fils_d : in, structure_arbre : in out
FinSi

R2 : Comment "Encoder fichier" ?

creer un fichier
compteur <- 0
octet <- 0
Encoder le tableau infixe tab_infixe : in
encoder_sda (structure_arbre, compteur, octet) structure_arbre : in, compteur, octet : in out
Encoder le texte tab_huffman : in, compteur, octet : in out
encoder le dernier octet compteur, octet : in out
fermer le fichier

R2 : Comment "Afficher les étapes en cas de bavard"?

Si Argument_Count = 2 **Alors** - - correspond au cas ou on veut les affichages
Utiliser les fonctions affichages.
fin si

R3 : Comment "Enregistrer le caractère dans le tableau des noeuds" ?

initialiser(feuille) feuille : out T_arbre
Si le caractère n'a jamais été enregistré dans tab_noeuds **Faire**
initialiser le caractere tab_noeuds, feuille : in out, val : in
Sinon
Mettre à jour la fréquence du caractère tab_noeuds, feuille : in out, val : in
FinSi

R3 : comment "Ajouter le caractère de fin de fichier" ?

initialiser(feuille) feuille : out T_arbre
creer(feuille, 0, -1) feuille : in out

enregistrer (tab_noeuds, longueur (tab_noeuds), feuille) tab_noeuds, feuille : in

R3 : Comment “Calculer l’indice_min de tab_noeuds ” ?

indice <- 1

Tant Que est_vide(la_donnee(tab, indice))

indice <- indice +1 - - On cherche l’indice du premier élément du tableau

Fin TantQue

Chercher l’indice du min à partir du sous-tableau qui va de indice à longueur(tab)

indice : in out, tab_noeuds : in

R3 : Comment “traiter la fusion du 1er et 2e min” ?

fils_g <- tab_noeuds (indice)	fils_g : out T_arbre
tab_noeuds (indice) <- rien	
Calculer indice_min de tab_noeuds	tab_noeuds : in, indice : out integer
fils_d <- tab_noeuds (indice)	fils_d : out T_arbre
Fusionner les arbres	fils_g, fils_d : in, arbre : out T_arbre
tab_noeuds (indice) <- arbre	
Calculer indice_min de tab_noeuds	tab_noeuds : in, indice : out integer

R3 : Comment “creer_tab_huffman (arbre, tab_huffman, code) ”?

Si est_feuille(arbre) faire

indice <- la_donnee(arbre)

tab_huffman (indice) <- code

Sinon

creer code0 et code1 code : in, code0, code1 : out T_LCA

creer_tab_huffman (fls_gauche(arbre), tab_huffman, code0)

creer_tab_huffman (fils_droit(arbre), tab_huffman, code1)

vider(code)

FinSi

R3 : comment “creer le tableau infixe brut” ?

Si est_feuille (arbre) **faire**

taille<- taille+ 1;

donnee <- la_donnee (arbre);

tab_infixe (taille) = donnee;

Sinon faire

creer le tableau infixe brut	arbre^.fils_g : in, tab_infixe, taille : in out
creer le tableau infixe brut	arbre^.fils_d : in, tab_infixe, taille : in out

FinSi

tab_infixe(taille + 1) = tab_infixe(taille)

R3 : comment “trouver la position du symbole de fin de fichier” ?

```
indice <- 0
repeter
        indice <- indice + 1
        jusqu’a tab_infixe (indice) = -1
fin repeter
```

R3 : comment “echanger le symbole de fin de fichier avec le debut du tableau” ?

```
donnee <- tab_infixe (1)
tab_infixe(1) <- indice
si indice /= 1 faire
        tab_infixe (indice) <- donnee
fin si
```

R3 : comment “Encoder le tableau infixe” ?

```
indice <- 1
donnee <- tab_infixe(indice)
ecrire donnee dans le fichier
repeter
        indice <- indice + 1
        donnee <- tab_infixe(indice)
        ecrire donnee dans le fichier
        jusqu’a donnee = tab_infixe (indice -1)
fin repeter
```

R3 : comment “encoder_sda (sda, compteur, octet)” ?

```
si compteur = 8 faire
        encoder l’octet          compteur, octet : in out
fin si
si  sda /= rien faire
        ajouter la donnee de la lca dans l’octet          sda : in, compteur, octet : in out
fin si
```

R3 : Comment “Encoder le texte” ?

```
ouvrir le fichier texte
Répéter
        encoder un caractere du texte          tab_huffman : in, compteur, octet : in out
Jusqu'à la fin du texte
fin repeter
code <- tab_huffman (longueur(tab_huffman))          code : out T_LCA
```

encoder_sda (code, compteur , octet) code : in, compteur, octet : in out
fermer le fichier texte

R3 : comment “encoder le dernier octet” ?

tant que compteur < 8 faire
    compteur <- compteur + 1
    octet <- octet * 2
fin tant que
ecrire octet dans le fichier

R4 : Comment “Initialiser le caractere”

Creer(feuille, 1, val) feuille : in out, val : in
Enregistrer (tab_noeuds, val, feuille) tab_noeuds : in out, val, feuille : in

R4 : Comment “Mettre à jour la frequence du caractere”?

Freq <- la_cle(la_donnee(tab_noeuds, val) tab_noeuds, val : in, freq : out integer
Creer(feuille, freq +1, val) feuille : in out, val, freq : in
Enregistrer (tab_noeuds, val, feuille) tab_noeuds : in out, val, feuille : in

R4 : comment “Chercher l’indice du min à partir du sous-tableau qui va de indice à longueur(tab)”

min <- la_cle (tab_noeuds(indice))
pour i de indice a longueur(tab_noeuds) faire
    mettre a jour l’indice tab_noeuds : in, indice, min : in out
fin pour

R4 : comment “fusionner les arbres”

arbre <- new T_noeud
arbre^.cle <- fils_g^.cle + fils_d^.cle
arbre^.fils_g <- fils_g
arbre^.fils_d <- fils_d

R4 : comment “creer code0 et code1” ?

code0 <- code
code1 <- code
ajouter (code0, 0) code0 : in out
ajouter(code1, 1) code1 : in out

R4 : comment “encoder l’octet”

compteur <- 0
ecrire octet dans le fichier
octet <- 0

R4 : comment “ajouter la donnee de la lca dans l’octet”
bit <- sda^.donnee
octet <- (octet * 2) ou bit
compteur <- compteur + 1
encoder_sda (sda^.suivant, compteur, octet)

R4 : comment “encoder un caractere du texte” ?
Récupérer un caractère val : out integer
code <- tab_huffman (val) code : out T_LCA
encoder_sda (code, compteur, octet) code : in, compteur, octet : in out

R4 : Comment “ajouter (sda, valeur)”
Si sda= null faire
sda<- new T_celleule
sda^.valeur = valeur
sda^.suivant = null
Sinon
ajouter (sda^.suivant, valeur)
FinSi

R4 : Comment “encoder_caracteres (tab_huffman, tab_infixe, texte_code) ” ?
octet <- tab_infixe^.donnee
code <- tab_huffman(octet)
ajouter (texte_code, code)
Si tab_infixe^.suivant = null faire
ajouter (texte_code, code)
Sinon
encoder_caracteres (tab_huffman, tab_infixe^.suivant, texte_code)
FinSi

R5 : comment “mettre a jour l’indice”
si tab_noeuds(i) /= rien et alors min > la_cle (tab_noeuds (i)) faire
indice <- i
min <- la_cle (tab_noeuds (i))
fin si

Evaluation par les étudiants

		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	A
	Une seule décision ou répétition par raffinage	+
	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	A
	Le raffinage d'une action décrit complètement cette action	A
	Le raffinage d'une action ne décrit que cette action	A
	Les flots de données sont cohérents	A
	Pas de structure de contrôle déguisée	+
	Qualité des actions complexes	+

Exercice 2 : décompresser un fichier texte

R0: Décompresser un fichier texte

R1 : Comment “ Décompresser un fichier texte” ?

Récupérer tab infixe tab_infixe :out tab_entier.T_tableau, Taille : out integer

Récupérer la structure de l’arbre

structure_arbre : out T_LCA, taille : in, octet : out T_octet, compteur : out integer

Indice <- 0

Initialiser(arbre) arbre : out T_arbre
Reconstituer l'arbre arbre, structure_arbre, indice : in out, tab_infixe : in
Initialiser (sda_binaire) sda_binaire : out T_LCA
Créer tab huffman arbre : in, tab_huffman : out tab_LCA.T_tableau, sda_binaire : in out
Décoder fichier octet, compteur : in out, tab_huffman : in out

R2 : Comment "Récupérer tab infixe " ?
Taille <- 0
Récupérer les caractères taille :in out integer, tab_infixe : out T_tableau
traiter la position du symbole de fin de fichier

R2 : Comment "récupérer la structure de l'arbre" ?
Initialiser(structure_arbre) structure_arbre : out T_LCA
Nb_de_1 <- 0
Compteur <- 8
Octet <- 0
Tant Que Nb_de_1 < taille faire
ajouter un bit a structure_arbre structure_arbre, compteur, octet, Nb_de_1 : in out
Fin Tant Que

R2 : Comment "Reconstituer l'arbre" ?
Si est_vide(structure_arbre) faire
rien
Sinon Si la_donnee (structure_arbre) = 1 faire
creer une feuille arbre : in out, tab_infixe, indice : in
Sinon
creer un arbre arbre, indice, structure_arbre : in out, tab_infixe : in
Fin Si

R2 : Comment "décoder fichier" ?
Initialiser(courant) courant : out T_LCA
Créer le fichier texte
Répéter
décoder chaque caractère
octet, compteur : in out, tab_huffman : in, pos : out integer
Jusqu'à pos = taille(tab_huffman)
Fin Répéter
vider(courant)
Fermer le fichier texte

R3 : Comment "récupérer caractères " ?

Répéter

recuperer un caractere val: out integer

Taille <- taille + 1

Enregistrer(tab_infixe, taille, val)

Jusqu'à taille > 1 et alors val = tab_infixe(taille-1)

Fin Répéter

Taille <- taille -1

R3 : Comment "traiter la position du symbole de fin de fichier"

Indice <- tab_infixe (1)

Tab_infixe (1) <- tab_infixe (indice)

Tab_infixe (indice) <- -1

R3 : comment “ajouter un bit a structure_arbre” ?

transferer le premier bit de texte_code

bit : out T_octet, octet : in out , compteur : in out

ajouter (structure_arbre, bit) structure_arbre : in out, bit : in

Si bit =1 faire

Nb_de_1 <- Nb_de_1 + 1

FinSi

R3 : Comment “creer une feuille”?

indice <- indice + 1

donnee <- tab_infixe(indice)

creer(arbre, 0, donnee)

Supprimer la premiere donnee de structure_arbre

R3 : Comment “creer un arbre”?

donnee <- 0

creer(arbre, 0, donnee) arbre : in out, donnee : in

Supprimer la premiere donnee de structure_arbre structure_arbre : in out

Reconstituer l’arbre

arbre^.fils_g : in out, structure_arbre : in out, tab_infixe : in, indice : in out

Reconstituer l’arbre

arbre^.fils_d : in out, structure_arbre : in out, tab_infixe : in, indice : in out

R3 : Comment “decoder chaque caractere”?

Transferer le premier bit de texte_code octet, compteur : in out, bit : out T_octet

ajouter(courant, bit) bit : in integer, courant : in out T_LCA

trouver position de courant dans tab_huffman courant, tab_huffman : in, pos : out integer
Si pos /= -1 et pos /= taille(tab_huffman) faire
ajouter pos au texte
vider(courant)
Fin Si

R4 : Comment “creer(arbre, cle, donnee)”?
Si est_vide(arbre) faire
arbre <- new T_nooeud
Fin Si
arbre^.cle <- cle
arbre^.donnee <- donnee
arbre^.fils_g <- rien
arbre^.fils_d <- rien

R4 : Comment “supprimer la premiere donnee de structure_arbre”?
Si structure_arbre /= rien faire
a_detruire <- structure_arbre
structure_arbre <- structure_arbre^.suivant
free(a_detruire)
Fin Si

R4 : Comment “transferer le premier bit de texte_code” ?
Si compteur = 8 faire
Compteur <- 1
Recuperer (texte_code, octet)
Sinon
Compteur<- compteur + 1
octet <- octet * 2
Fin Si
bit <- octet/128

R4 : Comment “trouver position de courant dans tab_huffman” ?
compteur <- 1
pos <- (-1)
Tant Que compteur <= capacite et pos = -1 faire
comparer courant et tab_huffman (compteur)
pos, compteur : in out, tab_huffman, couran t: in
Fin Tant Que

R5 : Comment “comparer courant et tab_huffman(compteur)”

Si sont_egales (tab_huffman(compteur), courant) faire

pos <- compteur

Fin Si

compteur <- compteur + 1

Evaluation par les étudiants

		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	A
	Une seule décision ou répétition par raffinage	+
	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	A
	Le raffinage d'une action décrit complètement cette action	A
	Le raffinage d'une action ne décrit que cette action	A
	Les flots de données sont cohérents	A
	Pas de structure de contrôle déguisée	+
	Qualité des actions complexes	+