PIM : Projet 1

Auteurs : MAILLET Tom, MATHURIN Maël

Notes:
Les procédures ou fonctions issues d'un module sont considérés comme élémentaires et ne sont donc pas spécifiées.
Nous avons mis un # devant les lignes qui sont considérées comme des actions élémentaires.
Les types utilisés et le types des variables sont renseignés à la fin de ce document, ainsi que les principales fonctions

Raffinage de "compression" :

	R0: Compresser un fichier grâce au codage de Huffman	Nom_Du_Fichier : in
#	R1: Comment "Compresser un fichier grâce au codage de Huffman"? **Ouvrir**(Fichier, In_File, Nom_Du_Fichier) Créer le tableau de Huffman et l'arbre de Huffman Créer le fichier compressé	 Nom_Du_Fichier : in; Fichier : out Fichier : in; Tableau_Huffman, Arbre_Huffman : out Fichier,Nom_Du_Fichier,Tableau_Huffman,Arbre_Huffman :in; Fichier_Compresse : out
	R2: Comment "Créer le tableau de Huffman et l'arbre de Huffman"? Calculer les occurrences de chaques Octet dans fichier Créer l'arbre de Huffman Créer la table de Huffman	 Fichier : in; Tableau_Occurences : out Tableau_Occurences : in; Arbre_Huffman : out Arbre_Huffman : in; Tableau_Huffman : out
#	R2: Comment "Créer le fichier compressé"? Initialiser le fichier compressé Initialiser(Suite_Bits) Calculer la suite de bits par parcours préfixe de l'arbre Ecrire le fichier compressé	 Table_Huffman, Arbre_Huffman : in; Fichier_Compresse : out Suite_Bits : out Arbre_Huffman : in (renommé Arbre); Suite_Bits ; in out Nom_Du_Fichier, Suite_Bits, Tableau_Huffman, Fichier : in
#	R3: Comment "Calculer les occurrences de chaque Octet dans Fichier"? Initialiser(Tableau_Occurences)	 Tableau_Occurences : out
#	Ajouter(Tableau Occurrences, -1, 1)	Tableau_Occurences : in out
#	**PourChaque** Octet **Dans** Fichier **Faire** Ajouter(Tableau_Occurences,Donnee(Tableau_Occurences,Octet)+1)	Fichier : in; Octet : out: Octet : in, Tableau_Occurences : in out
	FinPC	
#	R3: Comment "Créer l'arbre de Huffman"? Initialiser Liste_Arbre_Huffman **TantQue** Taille(Liste_Arbre_Huffman) > 1 **Faire** Réduire Liste_Arbre_Huffman	 Tableau_Occurences : in; Liste_Arbre_Huffman : out Liste_Arbre_Huffman : in Liste_Arbre_Huffman : in out
	FinTQ	
#	Arbre_Huffman **<-** Element_Indice(Liste_Arbre_Huffman, 0)	Liste_Arbre_Huffman : in, Arbre_Huffman : out
#	R3: Comment "Calculer la table de Huffman"? Initialiser(Tableau_Huffman)	 Tableau_Huffman : out
#	Initialiser(Code)	Code : out
	Créer Tableau_Huffman par parcours infixe de l'arbre	Arbre_Huffman : in (renommé Arbre); Tableau_Huffman : out
	Inverser les codes de Tableau_Huffman	Tableau_Huffman : in out
#	R3: Comment "Initialiser le fichier compressé"? **Créer**(Fichier_Compresse, Out_File, Nom_Du_Fichier + ".hff")	 Nom_Du_Fichier : in; Fichier_Compresse : out
	Calculer la suite des octets	Tableau_Huffman : in; Suite_Octets : out
	PourChaque Octet **Dans** Suite_Octets **Faire**	Suite_Octets : in; Octet : out
#	**Ecrire(**Fichier_Compressé, Octet)	Octet : in; Fichier_Compresse : in out
	FinPC	
	R3: Comment "Calculer la suite de bit par parcours infixe de l'arbre"?	

#	**Si** estFeuille(Arbre) **Alors**	Arbre : in
#	Ajouter_Fin(Suite_Bits, 1)	Suite_Bits : in out
	Sinon	
#	Ajouter_Fin(Suite_Bits, 0)	Suite_Bits : in out
	Calculer la suite de bit par parcours préfixe de l'arbre gauche	Arbre : in; Suite : in out
	Calculer la suite de bit par parcours préfixe de l'arbre droite	Arbre : in; Suite : in out
	FinSI	
	R3: Comment " Ecrire le fichier compressé"?	
#	Octet **<-** 0	
#	i **<-** 0	Octet : out
	Ecrire la suite de bits	i : out
	Ecrire les octets du fichiers encodé	Suite_Bits : in; Fichier_Compresse, Octet, i : in out
		Fichier : in; Fichier_Compresse, Octet, i:; in out
	R4: Comment "Initialiser Liste_Arbre_Huffman"?	
#	Initialiser(Liste_Arbre_Huffman)	Liste_Arbre_Huffman
	PourChaque Clé **Dans** Tableau_Occurences **Faire**	Cle : out; Tableau_Occurences : in
#	Élément **<-** Clé	Cle : in; Element : out
#	Valeur **<-** Donnee(Tableau_Occurences, Cle)	Tableau_Occurences, Cle : in; Valeur :out
#	Initialiser(Arbre_Elementaire, Valeur, Element)	Valeur, Element : in; Arbre_Elementaire : out
#	Ajouter(Liste_Arbre_Huffman, Sous_Arbre)	Sous_Arbre : in; Liste_Arbre_Huffman : out
	FinPC	
	R4: Comment "Réduire Liste_Arbre_Huffman"?	
#	Arbre_Min1 <- Retirer_Min(Liste_Arbre_De_Huffman)	Arbre_Min1 : out; Liste_Arbre_Huffman : in out
#	Arbre_Min2 <- Retirer_Min(Liste_Arbre_De_Huffman)	Arbre_Min2 : out; Liste_Arbre_Huffman : in out
#	Fusionner(Arbre_Min1, Arbre_Min2)	Arbre_Min2 : out; Arbre_Min1 : in out
#	Ajouter(Liste_Arbre_Huffman, Arbre_Min1)	Arbre_Min1 : in; Liste_Arbre_Huffman : in out
	R4: Comment "Créer Tableau_Huffman par parcours infixe de l'arbre"?	
#	**Si** estFeuille(Arbre) **Alors**	Arbre : in
#	Ajouter(Tableau_Huffman, Element(Arbre), Code)	Arbre, Code : in; Tableau_Huffman : in out
	Sinon	
#	Ajouter_Fin(Code, 0)	Code : in out
	Parcours infixe de l'arbre gauche avec Code	Arbre : in; Tableau_Huffman, Code : in out
#	Retirer_Fin(Code)	Code : in out
#	Ajouter_Fin(Code,1)	Code : in out
	Parcours infixe de l'arbre droite avec Code	Arbre : in; Tableau_Huffman, Code : in out
	FinSi	
	R4: Comment "Calculer le tableau des octets" ?	
#	Initialiser(Suite_Octets)	Suite_Octets : out
	PourChaque Cle **Dans** Tableau_Huffman **Faire**	Tableau_Huffman : in; Cle : out
#	Ajouter_Fin(Suite_Octets, Cle)	Cle : in; Suite_Octets : in out
	FinPC	
#	Ajouter_Fin(Suite_Octets, Cle)	Cle : in; Suite_Octets : in out
#	Indice <- Retirer_Elements(Suite_Octets, -1)	Suite_Octets : in; Indice : : out
#	Ajouter_Debut(Suite_Octets, Indice)	Indice : in; Suite_Octets : in out
	R4: Comment "Ecrire la suite de bits"?	
	PourChaque Bit **Dans** Suite_Bits **Faire**	Suite_Bits : in; Bit : out
	Assembler le bit à l'octet et l'écrire si besoin	Bit, Octet, Fichier_Compresse : in out
	FinPC	
	R4: Comment "Ecrire les octets du fichier encodé?"	
	PourChaque Element **Dans** Fichier **Faire**	Fichier : in; Element : out
#	Code **<-** Donnee(Tableau_Huffman, Element)	Code : out; Tableau_Huffman, Element : in
#	**TantQue** non estVide(Code) **Faire**	Code : in
#	Bit <- Retirer_Debut(Code)	Code : in; Bit : out
	Assembler le bit à l'octet et l'écrire si besoin	Bit, Octet, i, Fichier_Compresse : in out
	FinTQ	
	FinPC	
	Ecrire le code du fin de fichier	Tableau_Huffman : in; Bit, Octet, i, FIchier_Compresse : in out

	R5: Comment “Assembler le bit à l’octet et l’écrire si besoin”?	
#	Octet <- (Octet * 2) or Bit	Bit : out, Octet : in out
#	i <- i+1	i : in out
#	**Si** i = 8 **Alors**	i : in
#	**Ecrire(**Fichier_Compresse, Octet)	Octet : in; Fichier_Compresse : in out
#	Octet <- 0	Octet : out
#	i <- 0	i : out
	FinSI	
	R5: Comment “Ecrire le code du fin de fichier”	
#	Code **<-** Donnee(Tableau_Huffman, -1)	Code : out; Tableau_Huffman, Element : in
#	**TantQue** non estVide(Code) **Faire**	Code : in
#	Bit <- Retirer_Debut(Code)	Code : in; Bit : out
	Assembler le bit à l’octet et l’écrire si besoin --R5	Bit, Octet, i, Fichier_Compresse : in out
	FinTQ	
#	**TantQue** i < 8 **Faire**	i : in
#	Octet **<-** (Octet*2)	Octet : in out
	FinTQ	
#	**Ecrire(**Fichier_Compresse, Octet)	Octet : in; Fichier_Compresse : in out

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	+
	Tous les Ri sont écrits contre la marge et espacés	+
	Les flots de données sont définis	+
	Une seule décision ou répétition par raffinage	P
	Pas trop d'actions dans un raffinage (moins de 5 ou 6)	A
	Bonne présentation des structures de contrôle	A
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	+
	Pas de structure de contrôle déguisée	A
	Qualité des actions complexes	A

Raffinage de décompression :

	R0: Décompresser un fichier grâce au codage de Huffman	Nom_Du_Fichier : in
	R1: Comment “Décompresser un fichier grâce au codage de Huffman” ?	
#	Ouvrir(Fichier, In_File, Nom_Du_Fichier)	Nom_Du_Fichier : in; Fichier_Compresse : out
	Recréer l’arbre et le tableau de Huffman	Fichier_Compresse : in; Tableau_Huffman : out
	Écrire le fichier décompressé	Nom_Du_Fichier, Tableau_Huffman, Fichier_Compresse: in; Fichier : out
	R2: Comment “Recréer l’arbre et le tableau de codage de Huffman” ?	
	Récupérer la suite d’octets	Fichier_Compresse : in; Suite_Octets : out
	Recréer la table de Huffman	Arbre_Huffman : in; Tableau_Huffman, Octet, i : out
	R2: Comment “Écrire le fichier décompressé” ?	
	Initialiser le fichier décompressé	Fichier_Decompresse : out
	Ecrire le fichier décompressé avec la table de codage	Nom_Du_Fichier : in; Fichier_Decompresse : in out
	R3: Comment “Récupérer la suite d’octets”?	
#	**Lire**(Fichier_Compresse, Indice_Fin_Fichier)	Fichier_Compresse : in; Octet_Indice_Fin_Fichier : out
#	Initialiser(Suite_Octets)	Suite_Octets : out
#	**Lire**(Fichier_Compresse, Octet)	Fichier_Compresse : in; Octet : out
	Répéter	
#	Ajouter_Fin(Suite_Octets, Octet)	Octet : in; Suite_Octets : in out
#	Octet_Pre **<-** Octet	Octet : in; Octet_Pre : out
#	Lire(Fichier_Compresse, Octet)	Fichier_Compresse : in; Octet : out
#	**Jusqu’à** Octet = Octet_Pre	Octet, Octet_Pre: in
#	Ajouter_Indice(Suite_Octets, -1, Indice_Fin_Fichier)	Indice_Fin_Fichier : in; Suite_Octet : in out
	R3: Comment “Recréer la table de Huffman”?	
	Initialiser Code et Tableau_Huffman	Code, Tableau_Huffman : out
	Répéter	
#	**Lire(**Fichier_Compresse, Octet)	Fichier_Compresse : in, Octet : out
#	i <- 8	i : out
	Répéter	
#	Bit <- Octet / 128	Octet : in;Bit : out
#	Octet <- Octet * 2	Octet : in out
#	i <- i -1	i : in out
	Agir en fonction de la valeur du bit	Bit, Tableau_Huffman : in, Code : in out
#	**Jusqu’à** i = 0 **ou** estVide(Code)	i, Code : in
#	**Jusqu'à** estVide(Code)	Code : in
	R3: Comment “Initialiser le fichier décompressé?”	
#	n <- Nom_Fichier_Compresse’range	Nom_Fichier_Compresse : in; n : out
#	Creer(Fichier, Out_File, Nom_Fichier_Compresse(1..n-4))	Nom_Fichier_Compresse : in; Fichier : out
	R3: Comment “Écrire le fichier décompressé avec la table de codage” ?	
#	Initialiser(Code)	Code : out
#	Code_Fin <- Cle_Associe(Tableau_Huffman, -1)	Tableau_Huffman : in; Code_Fin : in
#	**TantQue** Code /= Code_Fin **Faire**	Code, Code_Fin : in
#	**TantQue** i /= 0 **Faire**	i : in
	Actualiser le bit, l’octet, i et le code	Bit : out; Octet, i, Code : in out
	Ecrire l’octet correspondant au code si le code existe	Tableau_Huffman : in; Fichier, Code : in out
	FinTQ	
#	**Lire**(Fichier_Compresse, Octet)	Fichier_Compresse : in; Octet : out
#	i <- 8	i : out

	FinTQ	
	R4: Comment “Initialiser Indice_Element, Code et Tableau_Huffman“?	
#	Initialiser(Code)	Code : out
#	Initialiser(Tableau_Huffman)	Tableau : out
	R4: Comment “Agir en fonction de la valeur du bit”?	
#	**Si** Bit = 0 **Alors**	Bit : in
#	Ajouter _Fin(Code, 0)	Code : in out
	Sinon	
#	Ajouter(Tableau_Huffman, Code, Retirer_Debut(Suite_Octets))	Tableau_Huffman, Suite_Octets : in out; Code : in
#	**TantQue** Dernier_Element(Code) = 1 **Faire**	Code : in
#	Retirer_Fin(Code)	Code : in out
	FinTQ	
#	**Si** non estVide(Code) **Alors**	Code : in
#	Retirer_Fin(Code)	Code : in out
#	Ajouter_Fin(Code, 1)	Code : in out
	FinSI	
	FinSI	
	R4: Comment “Actualiser le bit, l’octet, i et le code”?	
#	Bit <- Octet / 128	Octet : in; Bit : out
#	Octet <- Octet * 2	Octet : in out
#	i <- i - 1	i : in out
#	Ajouter_Fin(Code, Bit)	Bit : in; Code : in out
	R4: Comment “Ecrire l’octet correspondant au code si le code existe”?	
#	**Si** estCle(Tableau_Huffman(Code)) **Alors**	Tableau_Huffman, Code : in
#	**Ecrire**(Fichier, Code)	Code : in; Fichier : in out
#	Vider(Code)	Code : in out
	FinSi	

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	+
	Tous les Ri sont écrits contre la marge et espacés	+
	Les flots de données sont définis	+
	Une seule décision ou répétition par raffinage	P
	Pas trop d'actions dans un raffinage (moins de 5 ou 6)	A
	Bonne présentation des structures de contrôle	A
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	A
	Qualité des actions complexes	A

Tableaux des principaux types utilisés, avec leurs fonctions importantes et les variables qui les utilisent

T_SDA (stockage de donnée que l'on peut accéder grâce à une clé)	procédure Initialiser(Sda) procédure Ajouter(Sda, Clé, Donnée) fonction Donnee(Sda, Clé) retourne Donnée fonction Cle_Associe(Sda,Donnee) retourne Cle fonction estCle(Sda) retourne Booleen procedure Pour_Chaque_Cle(Sda, procedure(Cle))	Tableau_Huffman Tableau_Occurences
T_LISTE_C (liste chainé d'éléments, indexé à 0)	procédure Initialiser(Liste) procédure Ajouter_Debut(Liste_Element) procédure Ajouter_Fin(Liste,Donnee) procedure Ajouter_Indice(Liste, Donnee, Indice) procédure Vider(Liste) fonction Retirer_Debut(Liste) retourne Element fonction Retirer_Fin(Liste) retourne Element fonction Retirer_Element(Liste, Element) retourne Indice fonction Retirer_Min(Liste,function (Element) retourne Valeur) retourne Element fonction estVide(Liste) retourne Booleen fonction Dernier_Element(Liste) retourne Indice fonction Element_Indice(Liste, Indice) retourne Element fonction estEgal(Liste1, Liste2) retourne Booleen	Suite_Octets Suite_Bits Code Cle (pour decompression) Liste_Arbre_Huffman
T_ARBRE_B (arbre binaire où chaque cellule contient un éléments, et une valeur de type Integer)	procedure Initialiser(Arbre, Valeur, Element) procedure Fusionner(Arbre1 (in ou), Arbre2 (in)) function estFeuille(Arbre) retourne Booleen function Valeur(Arbre) retourne Valeur	Arbre_Huffman Arbre_Min1 Arbre_Min2 Arbre_Elementaire Arbre

D'autres types sont aussi utilisé :
T_Octet pour Octet et Cle (Cle pour compression)
T_Bit pour Bit
Int pour Valeur, i et n
File_Type pour Fichier et Decompresse
String pour Nom_Du_Fichier