**Deuxième partie du TP3– INF 2170 gr 10**

**Équipe G 16**

**Martin Valiquette** (VALM13037400) **, Olivier Viera** (VIEP12058605) **& Pierre-Alexandre Latour Leblanc** (LATP13098706)

**29 mai 2009**

**Code assemblé**

Addr Code Symbol Mnemon Operand Comment

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \* I N F 2 1 7 0 (TP3 partie 2)

; \*

; \* Ce programme prend un texte saisie par l'utilisateur et place ses mots en ordre

; \* alphabétique dans un arbre binaire. Ensuite, il affiche chaque mot (en ordre

; \* alphabétique) ainsi que son nombre d'occurrences.

; \*

; \* Les sous programmes suivants sont utilisés:

; \*

; \* New: Alloue une taille dans la zone de memoire demande. Sera utilise

; \* afin de creer des noeuds contenant les informations sur les mots

; \* ainsi que leur emplacement dans l'arbre binaire.

; \* Sous-programme fourni par Philippe Gabrini.

; \*

; \* LecMot: Lit les mots de la chaine entrée, converti les caractères alphabétiques

; \* en minuscules et transforme les caracteres invalides en carateres nuls Il s'occupe

; \* ensuite de les insérer dans un tampon. Les caracteres nuls delimitent les mots, un

; \* caracere nulls est conserve dans le tampon, entre chaque.

; \* Il retourne l'adresse de début et de fin du dernier mot saisi.

; \*

; \* Compare: Compare deux mots entrés en paramètre, il retournera une valeur entiere

; \* selon le resultat de la comparaison lexicographique entre les deux mots.

; \*

; \* Inserer; Insere le mot passé en paramètre dans l'arbre selon la position

; \* qu'il devrait y prendre. Il ajoutera automatiquement une occurrence

; \* au mot en question s'il est nouveau ou s'il existe déjà dans l'arbre.

; \* Ce sous-programme est recursif.

; \*

; \* Affiche: Affiche les mots de l'arbre passé en paramètre en ordre croissant.

; \* Chaque mot est accompagné de son nombre d'occurrences. Ce sous-programme est

; \* recursif.

; \*

; \*

; \*

; \* Groupe 16 >>

; \*

; \* Pierre-Alexandre Latour Leblanc

; \* P\_A\_L\_A@hotmail.com (LATP13098706)

; \*

; \*

; \* Martin Valiquette (VALM13037400)

; \* martin\_valiquette@videotron.ca

; \*

; \*

; \* Olivier Viera (VIEP12058605)

; \* viera.olivier@videotron.ca

; \*

; \*

; \*

; \* @version 02/06/2009

; \*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MOT: .EQUATE 0

NB\_OCC: .EQUATE 2

GAUCHE: .EQUATE 4

DROITE: .EQUATE 6

TAILLE: .EQUATE 8

T\_MAX: .EQUATE 20000

RACINE: .EQUATE 0

TMP: .EQUATE 2

nb: .EQUATE 4

ENDL: .EQUATE 0x000A ; Caractere fin de fichier

CAR\_FIN: .EQUATE 0x000B ; Caractere fin de fichier artificiel

0000 24 main: NOP0

0001 41036B STRO msgInv,d ; cout "Entrez chaine:"

0004 C00DF6 LDA ASCII,i ; Appel de la méthode lecMot pour fab

0007 E3FFF8 STA -8,s ;

000A C108E9 LDA ptrMot,d ;

000D E3FFFA STA -6,s ;

0010 680008 SUBSP 8,i ;

0013 160131 CALL lecMot ;

0016 C30000 LDA 0,s ; Load ladresse du debut du mot

0019 E108EB STA debMot,d ; Enregistrement de l'adresse du débu

001C B0FFFF CPA -1,i

001F 0A00E4 BREQ Fintamp

0022 C30002 LDA 2,s ; Load ladresse de la fin du mot

0025 E108E9 STA ptrMot,d ; Store dans ptrMot

0028 600004 ADDSP 4,i ;

002B 800001 SUBA 1,i ;

002E E108ED STA temp1,d ; Enregistrement de la position de pt

0031 C00000 LDA 0,i ;

0034 D208ED LDBYTEA temp1,n ; Compare "ptrMot,d - 1" et 0X0B

0037 B0000B CPA 0x000B,i ; 0X0B est insere artificiellement da

003A 0A00DD BREQ Fin ; si 0B alors cela veut dire qu'il n'

003D C008EF LDA racine,i ; Insersion du noeud racine avec GAUC

0040 E3FFF8 STA -8,s ;

0043 C108EB LDA debMot,d ;

0046 E3FFFA STA -6,s ;

0049 C008E9 LDA ptrMot,i ;

004C E3FFFC STA -4,s ;

004F C108F1 LDA heappnt,d ;

0052 E3FFFE STA -2,s ;

0055 680008 SUBSP 8,i ;

0058 160257 CALL Inserer ;

005B D208ED LDBYTEA temp1,n ; Compare "ptrMot,d - 1" et 0X0B

005E B0000A CPA 0x000A,i ; 0X0B est insere artificiellement da

0061 0A00D1 BREQ Affiche ; break

0064 24 Lecture: NOP0 ; Boucle principale

0065 C00DF6 LDA ASCII,i ; Appel de la méthode lecMot pour fab

0068 E3FFF8 STA -8,s ;

006B C108E9 LDA ptrMot,d ;

006E E3FFFA STA -6,s ;

0071 680008 SUBSP 8,i ;

0074 160131 CALL lecMot ;

0077 C30000 LDA 0,s ;

007A B0FFFF CPA -1,i ;

007D 0A00E4 BREQ Fintamp ;

0080 E108EB STA debMot,d ; Enregistrement de l'adresse du débu

0083 C30002 LDA 2,s ;

0086 600004 ADDSP 4,i ;

0089 E108E9 STA ptrMot,d ;

008C 800001 SUBA 1,i ;

008F E108ED STA temp1,d ;

0092 C00000 LDA 0,i ;

0095 D208ED LDBYTEA temp1,n ;

0098 B0000B CPA 0x000B,i ; 0X0B est insere artificiellement da

009B 0A00D1 BREQ Affiche ; Fin de ligne cas 2:

009E C108F1 LDA heappnt,d ;

00A1 B00DEE CPA heapfin,i ;

00A4 1000ED BRGT FinHeap

00A7 C108EF Insere: LDA racine,d ;

00AA E3FFF8 STA -8,s ;

00AD C108EB LDA debMot,d ;

00B0 E3FFFA STA -6,s ;

00B3 C008E9 LDA ptrMot,i ;

00B6 E3FFFC STA -4,s ;

00B9 C108F1 LDA heappnt,d ;

00BC E3FFFE STA -2,s ;

00BF 680008 SUBSP 8,i ;

00C2 160257 CALL Inserer ;

00C5 D208ED LDBYTEA temp1,n ; Compare "ptrMot,d - 1" et 0X0B

00C8 B0000A CPA 0x000A,i ; 0X0B est insere artificiellement da

00CB 0A00D1 BREQ Affiche ; break

00CE 040064 BR Lecture ;

00D1 C108EF Affiche: LDA racine,d ;

00D4 E3FFFE STA -2,s ;

00D7 680002 SUBSP 2,i ;

00DA 160312 CALL Afficher ;

00DD 50000A Fin: CHARO 0x000A,i ; Saut de ligne

00E0 41037E STRO msgFin,d ;

00E3 00 STOP ;

00E4 50000A Fintamp: CHARO 0x000A,i ; Saut de ligne

00E7 41039B STRO msgtamp,d ;

00EA 0400F3 BR STOP

00ED 50000A FinHeap: CHARO 0x000A,i ;

00F0 4103C9 STRO msgHeap,d ;

00F3 00 STOP: STOP ;

;---------New ( Noeud )

; Le sous-programme new alloue la taille demande et place l'adresse

; de la zone dans le pointeur dont l'adresse est passe en paramtre.

; @param Taille du noeud à créer => Ntaille

; @param Adresse où sauvegarder l'adresse du nouveau pointeur => Npoint

NvieuxX: .EQUATE 0 ; sauvegarde X

NvieuxA: .EQUATE 2 ; sauvegarde A

NadRet: .EQUATE 4 ; adresse retour

Npoint: .EQUATE 6 ; adresse pointeur remplir

Ntaille: .EQUATE 8 ; taille requise

00F4 680004 new: SUBSP 4,i ; espace local

00F7 E30002 STA NvieuxA,s ; sauvegarder A

00FA EB0000 STX NvieuxX,s ; sauvegarder X

00FD C108F1 LDA heappnt,d ;

0100 E40006 STA Npoint,sf ; adresse retourne

0103 C30008 LDA Ntaille,s ; taille du noeud

0106 700001 ADDA 1,i ; arrondir pair

0109 90FFFE ANDA 0xFFFE,i ;

010C 7108F1 ADDA heappnt,d ; nouvelle valeur

010F B00DF5 CPA heaplmt,i ;

0112 10011B BRGT new0 ; si pas dpass la limite du heap

0115 E108F1 STA heappnt,d ; mettre jour heappnt

0118 040121 BR new1 ; et terminer

011B C00000 new0: LDA 0,i ; sinon

011E E40006 STA Npoint,sf ; mettre pointeur NULL

0121 C30004 new1: LDA NadRet,s ; dplacer adresse retour

0124 E30008 STA Ntaille,s ;

0127 C30002 LDA NvieuxA,s ; restaurer A

012A CB0000 LDX NvieuxX,s ; restaurer X

012D 600008 ADDSP 8,i ; nettoyer pile

0130 58 RET0 ; return; }

;------- lectureMot

; Voici les trois opprations de ce sous programme:

;

; 1) Lire chaque mot d'une chaine,

; 2) les convertir les carateres alphabetique en minuscules et transformer des caractères invalides

; selon la table ASCII modifiee mise en parametre.

; 3) Placer le mot lu dans le tampon ptrMot et retourner son adresse.

;

; @param Table de conversion => ASCII

; @param Adresse tu tampon pour les mot => ptrMot

; @return Adresse du mon insere => debMot

lecCode: .EQUATE 0 ; Variable temporaire

lecRegX: .EQUATE 2 ; Registre X

lecRegA: .EQUATE 4 ; Registre A

lecARet: .EQUATE 6 ; Adresse de retour

lecConv: .EQUATE 8 ; tableau de conversion

lecPtr: .EQUATE 10 ; Adresse du tampon de mots

lecVRet: .EQUATE 12 ; Valeur de retour ( Adresse du mot )

lecVRet2:.EQUATE 14 ; Adresse du tampon de mots

lecNARet:.EQUATE 10 ; Nouvelle valeur de retour

0131 680006 lecMot: SUBSP lecARet,i ; espace local

0134 E30004 STA lecRegA,s ; sauvegarde A

0137 EB0002 STX lecRegX,s ; sauvegarde X

013A C80000 LDX 0,i ;

013D C00000 LDA 0,i ;

0140 C3000A LDA lecPtr,s ;

0143 E3000C STA lecVRet,s ; Met l'adresse actuelle du tampon da

0146 C00000 LDA 0,i ;

0149 4B0000 lecBou0: CHARI lecCode,s ; cin >> caractere;

014C C00000 LDA 0,i ;

014F D30000 LDBYTEA lecCode,s ;

0152 E30000 STA lecCode,s ;

0155 CB0000 LDX lecCode,s ; prend le code du caratere

0158 D70008 LDBYTEA lecConv,sxf; transpose le code dans le table de

015B B00001 CPA 1,i ; compare avec 1 apres conversion : c

015E 0A01A0 BREQ lecFinB ;

0161 B00000 CPA 0,i ;

0164 0C0182 BRNE lecEcri ; le caractere n'est pas nul: break

0167 040149 BR lecBou0 ; tant que lecaratere lu est null apr

016A 4B0000 lecBou: CHARI lecCode,s ; On lit un caractère

016D C00000 LDA 0,i ;

0170 D30000 LDBYTEA lecCode,s ;

0173 B0000A CPA 0x000A,i ; On vérifie si c'est un retour de ch

0176 0A01A9 BREQ lecFinA ; Si c'est le cas on quitte

0179 E30000 STA lecCode,s ;

017C CB0000 LDX lecCode,s ;

017F D70008 LDBYTEA lecConv,sxf;

0182 F4000A lecEcri: STBYTEA lecPtr,sf ; On écrit le caractere dans le tampo

0185 0A01C8 BREQ lecFin ; S'il est égal à zéro, on quitte

0188 B00001 CPA 1,i ;

018B 0A01A9 BREQ lecFinA ;

018E C3000A LDA lecPtr,s ;

0191 700001 ADDA 1,i ;

0194 B008DF CPA finTamp,i ; Fin tampon est un .BLOCK placé apre

0197 0A01C2 BREQ lecFinC ;

019A E3000A STA lecPtr,s ;

019D 04016A BR lecBou ; Tant que le caratere lu n'est pas n

01A0 C0000B lecFinB: LDA 0x000B,i ;

01A3 F4000A STBYTEA lecPtr,sf ;

01A6 0401C8 BR lecFin ;

01A9 24 lecFinA: NOP0 ;

01AA C00000 LDA 0,i ;

01AD F4000A STBYTEA lecPtr,sf ;

01B0 C3000A LDA lecPtr,s ;

01B3 700001 ADDA 1,i ;

01B6 E3000A STA lecPtr,s ;

01B9 C0000A LDA 0x000A,i ;

01BC F4000A STBYTEA lecPtr,sf ;

01BF 0401C8 BR lecFin ;

01C2 C0FFFF lecFinC: LDA -1,i ; Quand le tampon est plein

01C5 E3000C STA lecVRet,s ; retourne -1

01C8 C3000A lecFin: LDA lecPtr,s ;

01CB 700001 ADDA 1,i ;

01CE E3000E STA lecVRet2,s ;

01D1 C30006 LDA lecARet,s ;

01D4 E3000A STA lecNARet,s ; Stocker l'adresse de retour

01D7 C30004 LDA lecRegA,s ; Restaure le Registre A

01DA CB0002 LDX lecRegX,s ; Restaure le Registre X

01DD 60000A ADDSP lecNARet,i ;

01E0 58 RET0 ;

;------- comparer

; Ce sous-programme compare deux mots de facon lexicographique

; et renvoit 1 si l premier mot est plus grand, -1 'il est plus

; petit et 0 si les deux mot sont egaux.

;

;@param adresse du premier mot => comT1

;@param adresse du deuxieme mot => comT2

;@return int 0 si =, -1 si <, 1 sir >

comInd: .EQUATE 0 ; Indice

comCar: .EQUATE 2 ; Caractere ( temporaire )

comRegX: .EQUATE 4 ; Registe X

comRegA: .EQUATE 6 ; Registre A

comARet: .EQUATE 8 ; variable temporaire

comT1: .EQUATE 10 ; mot 1

comT2: .EQUATE 12 ; mot 2

comVRet: .EQUATE 14 ; Adresse de retour

comNARet:.EQUATE 12 ; taille maximum

01E1 680008 compare: SUBSP 8,i ; espace local

01E4 E30004 STA comRegX,s ; sauvegarde A

01E7 EB0006 STX comRegA,s ; sauvegarde X

01EA C80000 LDX 0,i ;

01ED C00000 LDA 0,i ;

01F0 E3000E STA comVRet,s ;

01F3 EB0000 STX comInd,s ;

01F6 D7000A comBLec: LDBYTEA comT1,sxf ; On compare lettre par lettre

01F9 B00000 CPA 0,i ;

01FC 0A0238 BREQ longDifA ;

01FF D7000C LDBYTEA comT2,sxf ;

0202 B00000 CPA 0,i ;

0205 0A022F BREQ finAp ; Si l'octet est nul, un des mots est

0208 CB0000 LDX comInd,s ;

020B D7000C LDBYTEA comT2,sxf ;

020E E30002 STA comCar,s ;

0211 D7000A LDBYTEA comT1,sxf ;

0214 B30002 CPA comCar,s ;

0217 080226 BRLT finAv ; Lorsque les caractères sont différe

021A 10022F BRGT finAp ; la valeur selon la différence

021D 780001 ADDX 1,i ;

0220 EB0000 STX comInd,s ;

0223 0401F6 BR comBLec ;

0226 C0FFFF finAv: LDA -1,i ; -1, le premier mot est plus petit

0229 E3000E STA comVRet,s ;

022C 040247 BR finCom ;

022F C00001 finAp: LDA 1,i ; 1, le premier mot est plus grand

0232 E3000E STA comVRet,s ;

0235 040247 BR finCom ;

0238 D7000C longDifA:LDBYTEA comT2,sxf ; Le premier mot est plus long

023B 0A0241 BREQ memeLong ;

023E 0C0226 BRNE finAv ;

0241 C00000 memeLong:LDA 0,i ; Les deux mots sont égaux

0244 E3000E STA comVRet,s ;

0247 C30008 finCom: LDA comARet,s ;

024A E3000C STA comNARet,s ; Stocker l'adresse de retour

024D C30006 LDA comRegA,s ; Restaurer Registre A

0250 CB0004 LDX comRegX,s ; Restaurer Registre X

0253 60000C ADDSP comNARet,i ;

0256 58 RET0 ;

;------- Inserer

; Sous programme recursif qui parcour l'arbre et le cherche dans le but

; d'inserer le mot a la bonne position dans l'arbre

;

; @param Adresse du noeud => Insptnd

; @param Adresse du mot => Insptmot

; @param Adresse de la fin du mot => Insptfi

; @param Adresse de la zone mémoire réservé => Inshpptr

InsInd: .EQUATE 0

InsRegX: .EQUATE 2 ; Registre X

InsRegA: .EQUATE 4 ; Registre A

InsRet: .EQUATE 6 ;

Insptnd: .EQUATE 8 ; pointeur noeud

Insptmot:.EQUATE 10 ; pointeur mot

Insptfi: .EQUATE 12 ; addresse du pointeur fin mot

Inshpptr:.EQUATE 14 ; pointeur du heap

InsNRet: .EQUATE 14 ;

0257 680006 Inserer: SUBSP 6,i

025A E30004 STA InsRegA,s ; sauvegarde A

025D EB0002 STX InsRegX,s ; sauvegarde X

0260 C80000 LDX 0,i ; indice = 0 ;

0263 C30008 LDA Insptnd,s ;

0266 B00000 CPA 0x0000,i ;

0269 0A02C3 BREQ nNoeud ;

026C C80000 compa: LDX MOT,i ;

026F C70008 LDA Insptnd,sxf;

0272 E3FFFC STA -4,s ;

0275 C3000A LDA Insptmot,s ;

0278 E3FFFE STA -2,s ;

027B 680004 SUBSP 4,i ;

027E 1601E1 CALL compare ; On compare le mot avec la racine

0281 C30000 LDA 0,s ;

0284 B00000 CPA 0,i ;

0287 0A02E7 BREQ incre ; Si 0, le mot existe déjà

028A 080290 BRLT droit ;

028D 100296 BRGT gauche ;

0290 C80006 droit: LDX DROITE,i ; Le mot est plus grand (alphabétique

0293 040299 BR recure ;

0296 C80004 gauche: LDX GAUCHE,i ; Le mot est plus petit (alphabétique

0299 C70008 recure: LDA Insptnd,sxf; La racine est le mot choisi selon l

029C E3FFF8 STA -8,s ;

029F C3000A LDA Insptmot,s ;

02A2 E3FFFA STA -6,s ;

02A5 C3000C LDA Insptfi,s ;

02A8 E3FFFC STA -4,s ;

02AB C3000E LDA Inshpptr,s ;

02AE E3FFFE STA -2,s ;

02B1 680008 SUBSP 8,i ;

02B4 160257 CALL Inserer ; On appelle insérer avec la nouvelle

02B7 C70008 LDA Insptnd,sxf;

02BA B00000 CPA 0x0000,i ;

02BD 0A02FC BREQ update ;

02C0 040302 BR final ;

02C3 C3000E nNoeud: LDA Inshpptr,s ; Le mot n'existe pas déjà dans l'arb

02C6 E3FFFC STA -4,s ; On crée donc un nouveau noeud lié à

02C9 C00008 LDA TAILLE,i ;

02CC E3FFFE STA -2,s ;

02CF 680004 SUBSP 4,i ;

02D2 1600F4 CALL new ;

02D5 C3000A LDA Insptmot,s ;

02D8 E4000E STA Inshpptr,sf;

02DB C80002 LDX 2,i ;

02DE C00001 LDA 1,i ;

02E1 E7000E STA Inshpptr,sxf;

02E4 040302 BR final ;

02E7 C80002 incre: LDX NB\_OCC,i ; Le mot existe déjà dans l'arbre

02EA C70008 LDA Insptnd,sxf;

02ED 700001 ADDA 1,i ; On ne fait qu'ajouter une occurence

02F0 E70008 STA Insptnd,sxf;

02F3 C3000A LDA Insptmot,s ;

02F6 E4000C STA Insptfi,sf ;

02F9 040302 BR final ;

02FC C3000E update: LDA Inshpptr,s ;

02FF E70008 STA Insptnd,sxf;

0302 C30006 final: LDA InsRet,s ;

0305 E3000E STA InsNRet,s ;

0308 C30004 LDA InsRegA,s ; restaure A

030B CB0002 LDX InsRegX,s ; restaure X

030E 60000E ADDSP InsNRet,i ; nettoyer pile

0311 58 RET0 ;

;------- Affiche

; Sous programme recursif affichant l'arbre en ordre alphabetique

; ainsi que le nombre d'occurence de chaque mot

; @param L'adresse de la racine de l'arbre

temp: .EQUATE 0

AffRegX: .EQUATE 2

AffRegA: .EQUATE 4

AffRet: .EQUATE 6

AffArb: .EQUATE 8

AffNRet: .EQUATE 8

0312 24 Afficher:NOP0 ;

0313 680006 SUBSP AffRet,i ;

0316 EB0002 STX AffRegX,s ;

0319 E30004 STA AffRegA,s ;

031C C30008 LDA AffArb,s ;

031F B00000 CPA 0x0000,i ; if (Arbre != NULL)

0322 0A035B BREQ affFinal ;

0325 C80004 LDX GAUCHE,i ;

0328 C70008 LDA AffArb,sxf ;

032B E3FFFE STA -2,s ;

032E 680002 SUBSP 2,i ;

0331 160312 CALL Afficher ; Afficher(Arbre->gauche)

0334 C80002 LDX NB\_OCC,i ;

0337 C40008 LDA AffArb,sf ;

033A E30000 STA temp,s ;

033D 440000 STRO temp,sf ; cout << Arbre->mot

0340 4103B8 STRO msgUti,d ;

0343 3F0008 DECO AffArb,sxf ; cout << Arbre->compte

0346 4103C2 STRO msgFois,d ;

0349 50000A CHARO 0x000A,i ; Saut de ligne

034C C80006 LDX 6,i ;

034F C70008 LDA AffArb,sxf ;

0352 E3FFFE STA -2,s ;

0355 680002 SUBSP 2,i ;

0358 160312 CALL Afficher ; Afficher(Arbre->droite)

035B C30006 affFinal:LDA AffRet,s ;

035E E30008 STA AffNRet,s ;

0361 C30004 LDA AffRegA,s ;

0364 CB0002 LDX AffRegX,s ;

0367 600008 ADDSP AffNRet,i ;

036A 58 RET0

; ------------- variables:

036B 456E74 msgInv: .ASCII "Entrez un texte. \x0A\x00"

72657A

20756E

207465

787465

2E200A

00

037E 417272 msgFin: .ASCII "Arrêt normal du programme. \x0A\x00"

EA7420

6E6F72

6D616C

206475

207072

6F6772

616D6D

652E20

0A00

039B 4C696D msgtamp: .ASCII "Limite du tampon atteinte. \x0A\x00"

697465

206475

207461

6D706F

6E2061

747465

696E74

652E20

0A00

03B8 207574 msgUti: .ASCII " utilisé \x00"

696C69

73E920

00

03C2 20666F msgFois: .ASCII " fois.\x00"

69732E

00

03C9 4C696D msgHeap: .ASCII "Limite du heap atteinte. \x0A\x00"

697465

206475

206865

617020

617474

65696E

74652E

200A00

03E4 000000 tabMot: .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

04E3 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

05E2 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

06E1 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

07E0 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

08DF 000000 finTamp: .BLOCK 10

000000

000000

00

08E9 03E4 ptrMot: .ADDRSS tabMot

08EB 03E4 debMot: .ADDRSS tabMot

08ED 0000 temp1: .WORD 0

08EF 08F3 racine: .ADDRSS heap

08F1 08F3 heappnt: .ADDRSS heap ; initialement pointe heap

08F3 000000 heap: .BLOCK 255 ; espace heap; dpend du systme

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

09F2 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

0AF1 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

0BF0 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

0CEF 000000 .BLOCK 255

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

000000

0DEE 000000 heapfin: .BLOCK 7

000000

00

0DF5 00 heaplmt: .BYTE 0 ;

0DF6 01 ASCII: .BYTE 1

0DF7 00 .BYTE 0

0DF8 00 .BYTE 0

0DF9 00 .BYTE 0

0DFA 00 .BYTE 0

0DFB 00 .BYTE 0

0DFC 00 .BYTE 0

0DFD 00 .BYTE 0

0DFE 00 .BYTE 0

0DFF 0A .BYTE 10

0E00 00 .BYTE 0

0E01 00 .BYTE 0

0E02 00 .BYTE 0

0E03 00 .BYTE 0

0E04 00 .BYTE 0

0E05 00 .BYTE 0

0E06 00 .BYTE 0

0E07 00 .BYTE 0

0E08 00 .BYTE 0

0E09 00 .BYTE 0

0E0A 00 .BYTE 0

0E0B 00 .BYTE 0

0E0C 00 .BYTE 0

0E0D 00 .BYTE 0

0E0E 00 .BYTE 0

0E0F 00 .BYTE 0

0E10 00 .BYTE 0

0E11 00 .BYTE 0

0E12 00 .BYTE 0

0E13 00 .BYTE 0

0E14 00 .BYTE 0

0E15 00 .BYTE 0

0E16 00 .BYTE 0

0E17 00 .BYTE 0

0E18 00 .BYTE 0

0E19 00 .BYTE 0

0E1A 00 .BYTE 0

0E1B 00 .BYTE 0

0E1C 00 .BYTE 0

0E1D 00 .BYTE 0

0E1E 00 .BYTE 0

0E1F 00 .BYTE 0

0E20 00 .BYTE 0

0E21 00 .BYTE 0

0E22 00 .BYTE 0

0E23 2D .BYTE 45 ; -

0E24 00 .BYTE 0

0E25 00 .BYTE 0

0E26 30 .BYTE 48 ; 0

0E27 31 .BYTE 49 ; 1

0E28 32 .BYTE 50 ; 2

0E29 33 .BYTE 51 ; 3

0E2A 34 .BYTE 52 ; 4

0E2B 35 .BYTE 53 ; 5

0E2C 36 .BYTE 54 ; 6

0E2D 37 .BYTE 55 ; 7

0E2E 38 .BYTE 56 ; 8

0E2F 39 .BYTE 57 ; 9

0E30 00 .BYTE 0

0E31 00 .BYTE 0

0E32 00 .BYTE 0

0E33 00 .BYTE 0

0E34 00 .BYTE 0

0E35 00 .BYTE 0

0E36 00 .BYTE 0

0E37 61 .BYTE 97 ; A

0E38 62 .BYTE 98 ; B

0E39 63 .BYTE 99 ; C

0E3A 64 .BYTE 100 ; D

0E3B 65 .BYTE 101 ; E

0E3C 66 .BYTE 102 ; F

0E3D 67 .BYTE 103 ; G

0E3E 68 .BYTE 104 ; H

0E3F 69 .BYTE 105 ; I

0E40 6A .BYTE 106 ; J

0E41 6B .BYTE 107 ; K

0E42 6C .BYTE 108 ; L

0E43 6D .BYTE 109 ; M

0E44 6E .BYTE 110 ; N

0E45 6F .BYTE 111 ; O

0E46 70 .BYTE 112 ; P

0E47 71 .BYTE 113 ; Q

0E48 72 .BYTE 114 ; R

0E49 73 .BYTE 115 ; S

0E4A 74 .BYTE 116 ; T

0E4B 75 .BYTE 117 ; U

0E4C 76 .BYTE 118 ; V

0E4D 77 .BYTE 119 ; W

0E4E 78 .BYTE 120 ; X

0E4F 79 .BYTE 121 ; Y

0E50 7A .BYTE 122 ; Z

0E51 00 .BYTE 0

0E52 00 .BYTE 0

0E53 00 .BYTE 0

0E54 00 .BYTE 0

0E55 00 .BYTE 0

0E56 00 .BYTE 0

0E57 61 .BYTE 97 ; a

0E58 62 .BYTE 98 ; b

0E59 63 .BYTE 99 ; c

0E5A 64 .BYTE 100 ; d

0E5B 65 .BYTE 101 ; e

0E5C 66 .BYTE 102 ; f

0E5D 67 .BYTE 103 ; g

0E5E 68 .BYTE 104 ; h

0E5F 69 .BYTE 105 ; i

0E60 6A .BYTE 106 ; j

0E61 6B .BYTE 107 ; k

0E62 6C .BYTE 108 ; l

0E63 6D .BYTE 109 ; m

0E64 6E .BYTE 110 ; n

0E65 6F .BYTE 111 ; o

0E66 70 .BYTE 112 ; p

0E67 71 .BYTE 113 ; q

0E68 72 .BYTE 114 ; r

0E69 73 .BYTE 115 ; s

0E6A 74 .BYTE 116 ; t

0E6B 75 .BYTE 117 ; u

0E6C 76 .BYTE 118 ; v

0E6D 77 .BYTE 119 ; w

0E6E 78 .BYTE 120 ; x

0E6F 79 .BYTE 121 ; y

0E70 7A .BYTE 122 ; z

0E71 00 .BYTE 0

0E72 00 .BYTE 0

0E73 00 .BYTE 0

0E74 00 .BYTE 0

0E75 00 .BYTE 0

0E76 00 .BYTE 0

0E77 00 .BYTE 0

0E78 00 .BYTE 0

0E79 00 .BYTE 0

0E7A 00 .BYTE 0

0E7B 00 .BYTE 0

0E7C 00 .BYTE 0

0E7D 00 .BYTE 0

0E7E 00 .BYTE 0

0E7F 00 .BYTE 0

0E80 00 .BYTE 0

0E81 00 .BYTE 0

0E82 00 .BYTE 0

0E83 00 .BYTE 0

0E84 00 .BYTE 0

0E85 00 .BYTE 0

0E86 00 .BYTE 0

0E87 00 .BYTE 0

0E88 00 .BYTE 0

0E89 00 .BYTE 0

0E8A 00 .BYTE 0

0E8B 00 .BYTE 0

0E8C 00 .BYTE 0

0E8D 00 .BYTE 0

0E8E 00 .BYTE 0

0E8F 00 .BYTE 0

0E90 00 .BYTE 0

0E91 00 .BYTE 0

0E92 00 .BYTE 0

0E93 00 .BYTE 0

0E94 00 .BYTE 0

0E95 00 .BYTE 0

0E96 00 .BYTE 0

0E97 00 .BYTE 0

0E98 00 .BYTE 0

0E99 00 .BYTE 0

0E9A 00 .BYTE 0

0E9B 00 .BYTE 0

0E9C 00 .BYTE 0

0E9D 00 .BYTE 0

0E9E 00 .BYTE 0

0E9F 00 .BYTE 0

0EA0 00 .BYTE 0

0EA1 00 .BYTE 0

0EA2 00 .BYTE 0

0EA3 00 .BYTE 0

0EA4 00 .BYTE 0

0EA5 00 .BYTE 0

0EA6 00 .BYTE 0

0EA7 00 .BYTE 0

0EA8 00 .BYTE 0

0EA9 00 .BYTE 0

0EAA 00 .BYTE 0

0EAB 00 .BYTE 0

0EAC 00 .BYTE 0

0EAD 00 .BYTE 0

0EAE 00 .BYTE 0

0EAF 00 .BYTE 0

0EB0 00 .BYTE 0

0EB1 00 .BYTE 0

0EB2 00 .BYTE 0

0EB3 00 .BYTE 0

0EB4 00 .BYTE 0

0EB5 00 .BYTE 0

0EB6 E0 .BYTE 224 ;

0EB7 E1 .BYTE 225 ; a+,

0EB8 E2 .BYTE 226 ;

0EB9 E3 .BYTE 227 ; a+~

0EBA E4 .BYTE 228 ;

0EBB E5 .BYTE 229 ; a+

0EBC E6 .BYTE 230 ; ae

0EBD E7 .BYTE 231 ;

0EBE E8 .BYTE 232 ;

0EBF E9 .BYTE 233 ;

0EC0 EA .BYTE 234 ;

0EC1 EB .BYTE 235 ;

0EC2 EC .BYTE 236 ;

0EC3 ED .BYTE 237 ; i+,

0EC4 EE .BYTE 238 ;

0EC5 EF .BYTE 239 ;

0EC6 00 .BYTE 0 ;

0EC7 F1 .BYTE 241 ; n+~

0EC8 F2 .BYTE 242 ;

0EC9 F3 .BYTE 243 ; o+,

0ECA F4 .BYTE 244 ;

0ECB F5 .BYTE 245 ; o+~

0ECC F6 .BYTE 246 ;

0ECD 00 .BYTE 0 ;

0ECE F8 .BYTE 248 ; o+/

0ECF F9 .BYTE 249 ;

0ED0 FA .BYTE 250 ; u+,

0ED1 FB .BYTE 251 ;

0ED2 FC .BYTE 252 ;

0ED3 00 .BYTE 0 ;

0ED4 00 .BYTE 0 ;

0ED5 FF .BYTE 255 ;

0ED6 E0 .BYTE 224 ;

0ED7 E1 .BYTE 225 ; a+,

0ED8 E2 .BYTE 226 ;

0ED9 E3 .BYTE 227 ; a+~

0EDA E4 .BYTE 228 ;

0EDB E5 .BYTE 229 ; a+

0EDC E6 .BYTE 230 ; ae

0EDD E7 .BYTE 231 ;

0EDE E8 .BYTE 232 ;

0EDF E9 .BYTE 233 ;

0EE0 EA .BYTE 234 ;

0EE1 EB .BYTE 235 ;

0EE2 EC .BYTE 236 ;

0EE3 ED .BYTE 237 ; i+,

0EE4 EE .BYTE 238 ;

0EE5 EF .BYTE 239 ;

0EE6 00 .BYTE 0 ;

0EE7 F1 .BYTE 241 ; n+~

0EE8 F2 .BYTE 242 ;

0EE9 F3 .BYTE 243 ; o+,

0EEA F4 .BYTE 244 ;

0EEB F5 .BYTE 245 ; o+~

0EEC F6 .BYTE 246 ;

0EED 00 .BYTE 0 ;

0EEE F8 .BYTE 248 ; o+/

0EEF F9 .BYTE 249 ;

0EF0 FA .BYTE 250 ; u+,

0EF1 FB .BYTE 251 ;

0EF2 FC .BYTE 252 ;

0EF3 00 .BYTE 0 ;

0EF4 00 .BYTE 0 ;

0EF5 FF .BYTE 255 ;

.END

Symbol Value

MOT 0000

NB\_OCC 0002

GAUCHE 0004

DROITE 0006

TAILLE 0008

T\_MAX 4E20

RACINE 0000

TMP 0002

nb 0004

ENDL 000A

CAR\_FIN 000B

main 0000

Lecture 0064

Insere 00A7

Affiche 00D1

Fin 00DD

Fintamp 00E4

FinHeap 00ED

STOP 00F3

NvieuxX 0000

NvieuxA 0002

NadRet 0004

Npoint 0006

Ntaille 0008

new 00F4

new0 011B

new1 0121

lecCode 0000

lecRegX 0002

lecRegA 0004

lecARet 0006

lecConv 0008

lecPtr 000A

lecVRet 000C

lecVRet2 000E

lecNARet 000A

lecMot 0131

lecBou0 0149

lecBou 016A

lecEcri 0182

lecFinB 01A0

lecFinA 01A9

lecFinC 01C2

lecFin 01C8

comInd 0000

comCar 0002

comRegX 0004

comRegA 0006

comARet 0008

comT1 000A

comT2 000C

comVRet 000E

comNARet 000C

compare 01E1

comBLec 01F6

finAv 0226

finAp 022F

longDifA 0238

memeLong 0241

finCom 0247

InsInd 0000

InsRegX 0002

InsRegA 0004

InsRet 0006

Insptnd 0008

Insptmot 000A

Insptfi 000C

Inshpptr 000E

InsNRet 000E

Inserer 0257

compa 026C

droit 0290

gauche 0296

recure 0299

nNoeud 02C3

incre 02E7

update 02FC

final 0302

temp 0000

AffRegX 0002

AffRegA 0004

AffRet 0006

AffArb 0008

AffNRet 0008

Afficher 0312

affFinal 035B

msgInv 036B

msgFin 037E

msgtamp 039B

msgUti 03B8

msgFois 03C2

msgHeap 03C9

tabMot 03E4

finTamp 08DF

ptrMot 08E9

debMot 08EB

temp1 08ED

racine 08EF

heappnt 08F1

heap 08F3

heapfin 0DEE

heaplmt 0DF5

ASCII 0DF6

No errors. Successful assembly.