HISOFT

PASCAL

MANUEL DU PROGRAMMEUR DE LA VERSION 4T

D 170

Diffusion : DIRECO INTERNATIONAL
30 Avenue de Messine
75008 PARIS

TABLE DES MATIERES

NOTE DE MISE EN OEUVRE DU PASCAL 4T HISOFT
SUR ZX SPECTRUM 48K

Chargement de HP4 à partir d'une bande

Sortir la cassette de bande magnétique de sa boîte. Mettre la cassette dans le lecteur de cassettes, la face A (face portant l'étiquette) vers le haut. Vérifier que le SPECTRUM est bien en Mode Saisie de Mots-clés et entrer :

LOAD""(appuyer sur J et deux fois sur ")

Appuyer maintenant sur PLAY du magnétophone: le programme de chargement du HP4S est d'abord inséré dans la machine; cette opération se fait automatiquement et elle est suivie du chargement du code HP4S. Si une erreur de bande est détectée, il faut appuyer sur la barre d'espacement du SPECTRUM, arrêter la bande, rebobiner jusqu'au point de départ, appuyer sur NEW (Touche 'A') du SPECTRUM et entrer à nouveau LOAD"". Si l'erreur persiste, il faut tenter de régler le volume du lecteur de cassette; si l'erreur persiste néanmoins, renvoyer la cassette à DIRECO qui la remplacera.

Lorsque le code HP4S a été chargé, il s'exécute automatiquement et le message 'Haut de RAM?' est affiché - pour connaître les opérations à exécuter, il faut se reporter au chapitre 00 du Manuel du Programmeur du Langage Pascal 4T Hisoft.

Mise en oeuvre du SPECTRUM

Le ZX SPECTRUM est un ordinateur relativement inhabituel et, dans une certaine mesure, la mise en oeuvre du HP4T en est un peu le reflet. Les différents codes de commande examinés dans le Manuel du Programmeur s'obtiennent comme suit sur le SPECTRUM :

RETURN	En appuyant sur 'ENTER'
CC	En appuyant sur 'CAPS SHIFT et 1
CH	A l'aide de DELETE, c'est-à-dire CAPS SHIFT et 0
CI	En appuyant sur CAPS SHIFT et 8
CP	Par CHR(16) dans une instruction WRITE ou WRITELN
CX	En appuyant sur CAPS SHIFT et 5
CS	En appuyant sur CAPS SHIFT et SPACE

Le système de saisie des mots-clés du ZX SPECTRUM n'est pas supporté (nous considérons que c'est là un avantage important); en revanche, tout le texte doit être inséré à l'aide des touches alphanumériques normales. En employant SYMBOL SHIFT et une touche quelconque (sauf I) on obtient toujours le symbole ASCII associé à cette touche et non le mot-clé, par exemple SYMBOL SHIFT T donne '>' et SYMBOL SHIFT G donne ')'. Il ne faut pas utiliser les symboles uniques <=, <> et >=; Pour entrer ces symboles, il faut utiliser une combinaison des symboles <, > et =.

L'éditeur travaille en majuscules mais l'inversion peut se faire de la façon normale à l'aide de CAPS SHIFT et 2.

L'utilisateur peut contrôler les attributs temporaires des différentes positions de caractères sur l'écran, en employant les codes de commande standard (par exemple WRITE(CHR(17),CHR(4)) a pour effet que le 'papier' devient vert) mais l'utilisateur ne peut pas modifier les attributs permanents. Si, pendant la mise en oeuvre de ces codes de commande, une séquence incorrecte est détectée, le message 'erreur Appel Système' est affiché et l'exécution est clôturée prématurément. Lors d'un vidage de texte ou de code objet sur bande, le message 'Démarrer la bande et appuyer sur une touche quelconque' est affiché deux fois - l'utilisateur doit y répondre à chaque fois.

Il est inutile de sauvegarder le programme de chargement puisqu'un programme de chargement automatique est toujours vidé avec le programme objet - si l'utilisateur a employé la commande 'T' pour sauvegarder le code objet et l'exécution sur bande, il suffit pour charger et exécuter le programme d'entrer simplement 'LOAD ""' à partir du BASIC. Après exécution complète du code objet, il peut être ré-exécuté, s'il n'a rien modifié ou transformé, en entrant 'GOTO 7' à partir du BASIC.

Si l'opérateur utilise la commande 'B' à partir de l'éditeur HP4T pour revenir au BASIC du ZX SPECTRUM, et en supposant qu'il ne change pas le programme BASIC, il est possible de ré-entrer dans l'éditeur HP4T suivant l'une des deux façons suivantes : en entrant 'GOTO 9' pour exécuter un démarrage à chaud, c'est-à-dire en préservant le programme Pascal ou en entrant 'GOTO 12' pour faire un démarrage à froid, en ré-initialisant le Pascal et en effaçant les textes Pascal existants.

L'imprimante ZX est supportée par l'option 'P' du compilateur (voir Paragraphe 3.2 du Manuel du Programmeur) et par CHR(16) dans une instruction WRITE Ou WRITELN. Il faut remarquer que, du fait de ce qui précède, l'utilisateur ne peut pas employer CHR(16) dans une instruction WRITE(LN) pour spécifier la couleur de l'encre. Il doit employer CHR(15) pour établir la couleur de l'encre.

Pour réaliser un exemplaire de travail de HP4S, il faut procéder comme suit :

1. Charger l'interface du BASIC. Dans le SPECTRUM, le Pascal 4T Hisoft est composé de deux parties, une interface avec BASIC et le compilateur, les programmes d'exécution, etc. à proprement parler. L'interface BASIC est enregistrée la première sur la bande de référence, suivie par le code du compilateur, des programmes d'exécution et de l'éditeur. Habituellement, le programme de l'interface BASIC charge

automatiquement le reste du code et provoque ensuite le démarrage automatique du compilateur. L'exécution de cette séquence doit être bloquée. Il faut donc charger la cassette de référence HP4S dans le magnétophone à cassette, entrer LOAD"" (simplement J"" à l'aide de l'entrée des mots-clés) et diffuser (PLAY) la bande; lorsque le SPECTRUM trouve l'en-tête du programme d'interface BASIC, il affiche 'Programme: HP4S' et charge ensuite le programme d'interface à proprement parler. Après le chargement du programme d'interface, l'écran se vide - l'ordinateur cherche maintenant automatiquement le code du compilateur. Il faut appuyer sur BREAK (ou sur la barre d'espacement) pour clôturer la recherche dès que le programme d'interface a été chargé. Arrêter la bande mais sans la rebobiner.

2. Charger le code du compilateur, les programmes d'exécution et l'éditeur en entrant LOAD "HP4S" CODE et en redémarrant la bande. Il ne faut pas oublier d'utiliser l'entrée des mots-clés pour LOAD et CODE.

3. Mettre une bande vierge distincte dans le magnétophone puor y enregistrer la nouvelle bande de référence HP4S. Entrer SAVE "HP4S" LINE 1 pour sauvegarder le programme d'interface BASIC afin qu'il s'exécute automatiquement. Mettre le magnétophone en mode enregistrement et opérer normalement pour enregistrer sur bande.

4. Lorsque le vidage du programme d'interface BASIC est terminé, laisser l'enregistreur en mode enregistrement et entrer SAVE "HP4S" CODE 24598, 21105.Ceci provoque la sauvegarde du compilateur, des programmes d'exécution et de l'éditeur sur la bande.

5. L'opérateur vient de créer une nouvelle bande de référence qui peut être utilisée de la même façon que la bande fournie par Hisoft. Il est recommandé d'en faire une copie et de stocker la bande de référence Hisoft dans un lieu sûr, à l'écart des champs magnétiques et de l'unité.

Hisoft autorise l'utilisateur à faire un seul exemplaire de travail.

Il ne faut pas hésiter à nous contacter si vous avez des problèmes avec Pascal 4T Hisoft - nous pouvons résoudre les problèmes si nous les connaissons !

CHAPITRE 0 - PREMISSES

0.0 Pour démarrer

Pascal HT Hisoft (HP4T) est une version rapide, puissante et d'emploi facile du langage Pascal tel qu'il est décrit dans le Rapport et Manuel de l'Utilisateur de Pascal (Jensen/Wirth Deuxième Edition). Les omissions par rapport à cette spécification sont les suivantes :

les FICHIERS ne sont pas mis en oeuvre mais les variables peuvent être stockées sur bande.
Un type "RECORD" ne peut pas contenir une partie "VARIANT".

Les PROCEDURES et FONCTIONS ne sont pas autorisées comme paramètres.

De nombreuses procédures et fonctions supplémentaires sont incluses pour tenir compte de l'évolution de l'environnement dans lequel sont utilisés les compilateurs; entre autres, on trouve maintenant POKE, PEEK, TIN, TOUT et ADDR.

Le compilateur occupe 12K de mémoire environ, tandis que les programmes d'exécution occupent à peu près 4K Tous deux sont fournis sur cassette, dans le format utilisé par les programmes d'exécution. Toutes les interfaces entre le langage HP4T et la machine hôte se font par des vecteurs placés de façon pratique au début des programmes d'exécution (voir le Guide des Modifications) - cette disposition facilite la rédaction par l'utilisateur de ses propres routines d'E/S personnalisées, le cas échéant.

Pascal 4T Hisoft utilise différents codes de commande, essentiellement dans l'éditeur. Bien entendu, des systèmes différents peuvent comporter des claviers conçus différemment et, dans chaque cas, la façon d'obtenir des codes de commande varie. Dans ce Manuel, les codes de commande utilisés sont appelés RETURN, CC, CH, CI, CP, CS et CX. La note de mise en oeuvre jointe précise les touches correspondantes de votre système.

Lorsque HP4T attend une ligne d'entrée, les caractères de commande peuvent être utilisés comme suit :

RETURN	pour terminer la ligne,
CC	pour revenir à l'éditeur,
CH	pour supprimer le dernier caractère frappé,
CI	pour passer à la prochaine position de tabulation,
CP	pour acheminer la sortie vers l'imprimante (si elle est disponible, ou bien si la sortie était déjà acheminée vers l'imprimante, pour la ramener sur l'écran,
CX	pour supprimer toute la ligne frappée jusqu'à ce point.

Un programme de chargement simple est également fourni avec ce logiciel afin que l'utilisateur puisse charger, à partir d'une bande, des données qui ont été enregistrées en format HP4T.

Ainsi, pour charger le compilateur et les programmes d'exécution à partir de la bande de référence fournie par Hisoft, l'utilisateur doit d'abord charger le programme de chargement; le cas échéant, ce programme est fourni dans une forme convenant à son chargement par le système d'exploitation de l'utilisateur. Si l'utilisateur ne peut pas accéder au système d'exploitation de l'ordinateur, la routine d'amorçage doit être entrée directement dans la mémoire de l'ordinateur, à l'aide d'un assembleur ou d'un langage de haut niveau comme BASIC - les modalités détaillées de cette opération et une routine d'amorçage schématique figurent dans le Guide des Modifications HP4T.

Lorsque le programme de chargement a été exécuté, il recherche un fichier enregistré en format sur bande HP4T. Lorsqu'un fichier de ce format a été trouvé, le programme de chargement le met en mémoire. Si une erreur est détectée à un moment quelconque de la lecture de la bande, un message est affiché; l'utilisateur doit alors rebobiner la bande jusqu'au début du fichier et relancer le chargement. Si plusieurs erreurs successives interviennent, il faut régler la commande de volume de l'enregistreur; si cette opération échoue, il faut renvoyer la bande à Hisoft qui la remplacera.

Ainsi, le programme de chargement assure automatiquement le chargement en mémoire du compilateur et des programmes d'exécution.

Lorsque le chargement du compilateur a abouti, il y a exécution automatique et lancement du message :

Haut de la RAM?

En réponse, l'utilisateur doit entrer un nombre décimal positif jusqu'à 65536 (et manoeuvrer ensuite RETURN) ou appuyer seulement sur RETURN (Voir la note de mise en oeuvre).

Si l'utilisateur entre un nombre, il est interprété comme représentant la plus haute position + 1 de la mémoire RAM; autrement, la première position non-RAM est calculée automatiquement. La pile de l'ordinateur est mise à cette valeur et on peut ensuite réserver des positions hautes de la mémoire (éventuellement, pour des extensions du compilateur) en fournissant délibérément une valeur inférieure à cette du haut vrai de la RAM. Dans la version du ZX Spectrum, le haut 'vrai' de la RAM est considéré comme étant le début de la zone des graphiques définie par l'utilisateur (UDG dans le Manuel Sinclair).

Le système lance alors le message d'interrogation suivant :

Haut de RAM pour 'T'

Dans ce cas il est possible d'entrer un nombre décimal ou de faire jouer la valeur par défaut qui est la valeur du 'haut de la RAM' spécifiée précédemment. Ce que l'utilisateur entre est interprété comme la pile, le code objet résultant étant exécuté après le lancement de la commande 'T' de l'éditeur (Pour plus de détails, voir le Chapitre 4). Il faut définir une pile d'exécution différente du haut de la RAM si, par exemple, l'utilisateur a écrit des extensions aux programmes d'exécution et souhaite les sauvegarder dans des positions hautes de la mémoire.

Enfin, le système demande :

Taille de la table?

La valeur entrée par l'utilisateur précise la quantité de mémoire à affecter à la table des symboles du compilateur.

A nouveau, il faut entrer un nombre décimal positif suivi de RETURN ou simplement appuyer sur RETURN et, dans ce cas, une valeur par défaut égale à la RFAM disponible divisée par 16 est utilisée pour définir la taille de la table des symboles. Cette table ne peut pas être au delà de l'adresse machine £8000* (32768 en décimal). Si la valeur indiquée est trop grande et provoque un débordement au delà de cette valeur, le système repose la question 'Haut de la RAM', etc.

Optionnellement, il est possible d'inscrire 'E' avant le nombre entré après ce message; dans ce cas, l'éditeur interne de lignes ne sera pas désigné pour emploi par le compilateur. Ainsi, si l'utilisateur souhaite employer son propre éditeur avec le compilateur (pour plus de détails sur cette opération, voir le guide des modifications), il doit entrer 'E'.

Maintenant, le compilateur et l'éditeur intégré (s'il doit être utilisé) sont translatés, c'est-à-dire mis à la fin de la table des symboles et l'exécution passe à l'éditeur supporté.

* Note : Dans ce Manuel, le symbole monétaire '£' est remplacé par le signe numéro (35 en décimal, 23 en hexadécimal, shift et '3') sur tous les systèmes qui n'utilisent pas le code ASCII britannique. Les valeurs numériques précédées de ce symbole sont en hexadécimal.(Soit sur Spectrum le symbole "#").

0.1 Objet de ce manuel

Ce manuel n'est pas destiné à vous apprendre le langage Pascal; pour cela, vous devez vous reporter aux excellents ouvrages figurant dans la bibliographie, si vous n'avez aucune expérience de la programmation en Pascal.

Ce manuel est un document de référence dont le rôle est de décrire les caractéristiques particulières de Pascal 4 Hisoft.

Le Chapitre 1 précise la syntaxe et la sémantique attendues par le compilateur.

Le Chapitre 2 explique de façon détaillée les différents identificateurs pré-définis qui existent dans Pascal 4 Hisoft, des CONSTANTES aux FONCTIONS.

Le Chapitre 3 contient des informations sur les différentes options compilateur disponibles ainsi que sur le format des commentaires.

Le Chapitre 4 précise la façon d'utiliser l'éditeur de lignes qui est une partie intégrante de HP4T; si vous ne souhaitez pas utiliser cet éditeur et que vous préférez réaliser une interface avec votre propre éditeur, vous devez consulter le Guide des Modifications HP4T.

Les chapitres précédents doivent être lus attentivement par tous les utilisateurs.

L'Annexe 1 précise en détail les messages d'erreur générés par le compilateur et les programmes d'exécution.

L'Annexe 2 donne la liste des identificateurs prédéfinis et des mots réservés.

L'Annexe 3 précise en détail la représentation interne des données dans Pascal 4 Hisoft - cette partie est utile pour les programmeurs qui n'ont pas peur de se salir les mains.

L'Annexe 4 donne quelques exemples de programmes Pascal - Ne manquez pas de les étudier si vous avez des difficultés à écrire des programmes en Pascal 4T Hisoft.

0.2 Compilation et exécution

L'explication détaillée de la façon de créer, modifier, compiler et exécuter un programme HP4T à l'aide de l'éditeur intégré figure au Chapitre 4 du présent Manuel. Pour savoir ce qu'il faut faire si vous utilisez votre propre éditeur, il faut se reporter au Guide des Modifications HP4T.

Lorsqu'il a été appelé, le compilateur génère une liste de la forme :

```
xxxx nnnn texte de la ligne source
```

où : xxxx est l'adresse à laquelle commence le code généré par cette ligne
nnnn est le numéro de la ligne après suppression des zéros de début.

Si une ligne contient plus de 80 caractères, le compilateur insère un caractère de saut-de-ligne pour que la longueur d'une ligne ne dépasse jamais 80 caractères.

Le cas échéant, la liste peut être transmise à l'imprimante en utilisant l'option P si elle est supportée (Voir le Chapitre 3).

Il est possible d'arrêter provisoirement la pause à tout moment en appuyant sur CS; il faut ensuite utiliser CC pour revenir à l'éditeur ou employer une autre touche pour redémarrer la fonction de listage.

Si une erreur est détectée pendant la compilation, le message '*ERREUR*' est affiché suivi d'une tête de flèche montante ('^') qui pointe après le symbole ayant généré l'erreur et un numéro d'erreur (Voir l'Annexe 1). La liste marque un temps d'arrêt; il faut appuyer sur 'E' pour revenir à l'éditeur et éditer la ligne affichée, sur 'P' pour revenir à l'éditeur et revenir à la ligne précédente (s'il y en a une) ou sur n'importe quelle touche pour continuer la compilation.

Si le programme se termine incorrectement (c'est-à-dire sans 'END'), le message 'Plus de texte' est affiché et le contrôle est rendu à l'éditeur.

Si le compilateur n'a plus d'espace de table, le message 'Plus d'espace de table' est affiché et le contrôle revient à l'éditeur. Habituellement, le programmeur procède alors à la sauvegarde du programme sur bande, recharge le compilateur et spécifie une 'taille de table' plus grande (Voir le paragraphe 0.0).

Si la compilation se termine correctement, mais qu'elle contenait des erreurs, le nombre d'erreurs détectées est affiché et le code objet est supprimé. Si la compilation réussit, le message 'Exécution?' est affiché; si vous désirez que le programme soit exécuté immédiatement, il faut répondre par 'Y'; autrement, le contrôle est rendu à l'éditeur.

Pendant une exécution du code objet, différents messages d'erreur d'exécution peuvent être générés (voir l'annexe 1). L'exécution peut être interrompue à l'aide de CS; ultérieurement, il faut employer CC pour clôturer l'exécution ou une autre touche pour la reprendre.

0.3 Différenciation des TYPES

Des langages différents ont des façons diverses de s'assurer que l'utilisateur n'emploie pas un élément de donnée d'une façon incompatible avec sa définition.

A la fin de l'échelle, on trouve le code machine dans lequel aucune vérification n'est faite sur le TYPE de la variable

citée. Nous trouvons ensuite un langage comme le 'Petit Pascal' dans lequel les données caractères, entières et booléennes peuvent être mélangées librement sans générer d'erreurs. Plus loin encore se trouve BASIC qui distingue les nombres et les chaînes et, parfois, les entiers et les réels (en utilisant dans certains cas le signe '%' pour désigner les entiers). C'est ensuite que se trouve Pascal qui va plus loin encore puisqu'il permet de distinguer les types énumérés par l'utilisateur. En haut de l'échelle (actuellement) se trouve un langage comme ADA dans lequel il est possible de définir des types numériques différents et incompatibles.

Les mises en oeuvre de Pascal comportent essentiellement deux approches de ce problème; l'équivalence structurelle et l'équivalence des noms. Pascal 4 Hisoft utilise l'équivalence des noms pour les "RECORD" et les "TABLEAUX". Les conséquences de ce choix sont précisées au Chapitre 1; contentons-nous ici d'en donner un exemple; imaginons que deux variables sont définies comme suit :

```
VAR A : ARRAY('A'..'C') OF INTEGER;
    B : ARRAY('A'..'C') OF INTEGER;
```

On peut alors être tenté de penser qu'il est possible d'écrire A:=B; malheureusement, dans le cas de Pascal 4 Hisoft, ceci produirait une erreur (*ERREUR*10) puisque deux enregistrements de TYPES distincts ont été créés par les définitions ci-dessus. En d'autres termes, l'utilisateur ne peut pas décider que A et B doivent représenter le même type de données. Il pourrait opérer de cette façon :

```
VAR A,B : ARRAY('A'..'C') OF INTEGER;
```

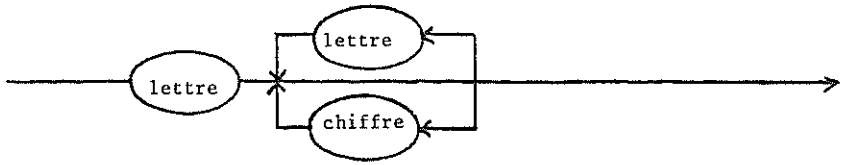
et maintenant, il est possible d'assigner librement A à B et vice versa puisqu'un seul enregistrement TYPE a été créé.

Bien qu'au premier abord l'approche de l'équivalence des noms puisse sembler un peu compliquée, de façon générale, elle provoque moins d'erreurs de programmation puisqu'elle exige du programmeur qu'il réfléchisse un peu plus avant de commencer.

CHAPITRE 1 : SYNTAXE ET SEMANTIQUES

Ce chapitre décrit de façon détaillée la syntaxe et la sémantique de Pascal 4 Hisoft; sauf indication contraire, la mise en oeuvre est conforme à la deuxième édition de l'ouvrage Rapport et Manuel de l'Utilisateur de Pascal (Jensen/Wirth).

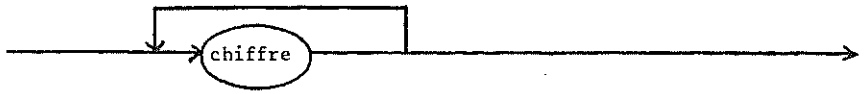
1.1 IDENTIFICATEUR



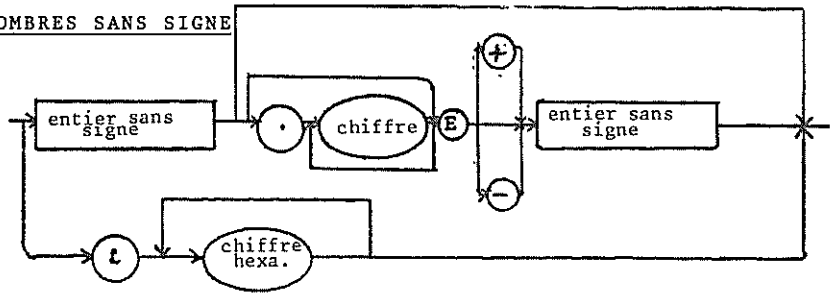
Seuls les 10 premiers caractères d'un identificateur sont considérés comme significatifs.

Les identificateurs peuvent contenir des minuscules et des majuscules. Les minuscules n'étant pas transformées en majuscules, les identificateurs SALUT, SALut et salut sont différents. L'entrée des mots réservés et des identificateurs prédéfinis doit seulement se faire en majuscules.

1.2 ENTIERS SANS SIGNE



1.3 NOMBRES SANS SIGNE



En Pascal 4 Hisoft, les entiers ont une valeur absolue inférieure ou égale à 32767. Les nombres entiers plus grands sont traités comme des réels.

La longueur de la mantisse des nombres réels est de 23 bits. La précision obtenue avec des réels est donc de 7 chiffres significatifs environ. Nota : Une partie de la précision est perdue si le résultat d'un calcul est très inférieur aux valeurs absolues de ses arguments, par exemple 2.00002 - 2 ne donne pas 0.00002. Ceci résulte de l'imprécision découlant de la représentation des fractions décimales par des fractions binaires. Ce phénomène n'a pas lieu lorsque des entiers de taille modérée sont représentés par des réels, par exemple 200002 - 20000 = 2 exactement.

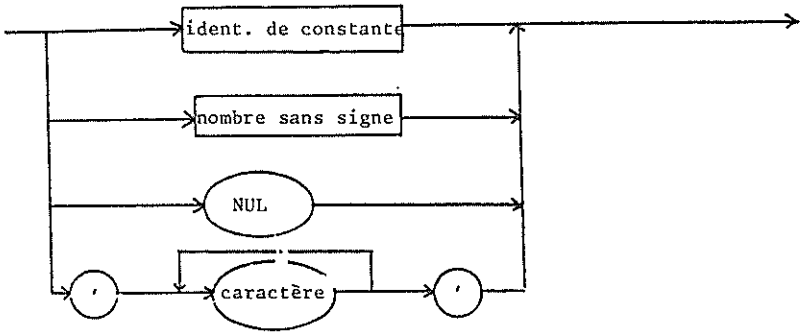
Le plus grand réel disponible est 3.4E38 et le plus petit est 5.9E-39.

Pour spécifier des nombres réels, il est inutile d'employer plus de 7 chiffres dans la mantisse puisque les chiffres supplémentaires sont ignorés, à l'exception de leur valeur de position.

Lorsque la précision est importante, il faut éviter les zéros de début puisqu'ils sont comptés comme un chiffre chacun. Ainsi, 0,000123456 est une représentation moins précise que 1.23456E-4.

Le programmeur peut utiliser des nombres hexadécimaux pour préciser des adresses de mémoire pour l'édition des liens du langage d'assemblage. Nota : il doit y avoir un chiffre hexadécimal au moins après le symbole '£'; autrement, une erreur (*ERREUR*51) est générée.

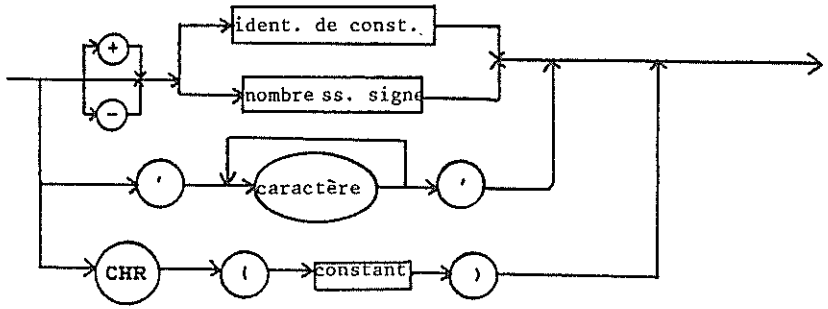
1.4 CONSTANTE SANS SIGNE



Nota : Une chaîne ne peut pas contenir plus de 255 caractères. Les types de chaînes sont TABLEAU (1..N) DE CAR où N est un entier entre 1 et 255 inclus. Les chaînes de littéraux ne doivent jamais contenir un caractère de fin de ligne (CHR(13)); autrement, une '*ERREUR*68' est lancée.

Les caractères disponibles sont le jeu étendu complet des valeurs ASCII à 256 éléments. Pour assurer la compatibilité avec le Pascal Standard, le caractère nul n'est pas représenté par "; il faut utiliser CHR(0).

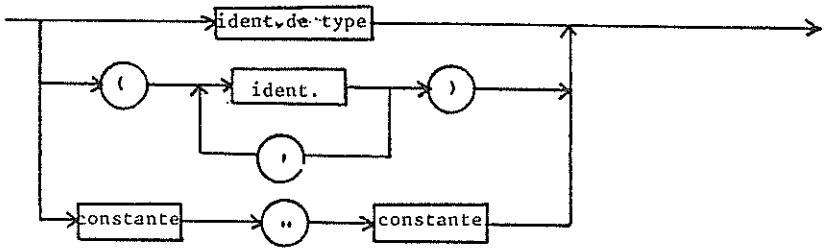
1.5 CONSTANTES



La construction CHR non standard est donnée ici pour permettre l'emploi de constantes comme caractères de commande. Dans ce cas, la constante entre parenthèses doit être du type entier.

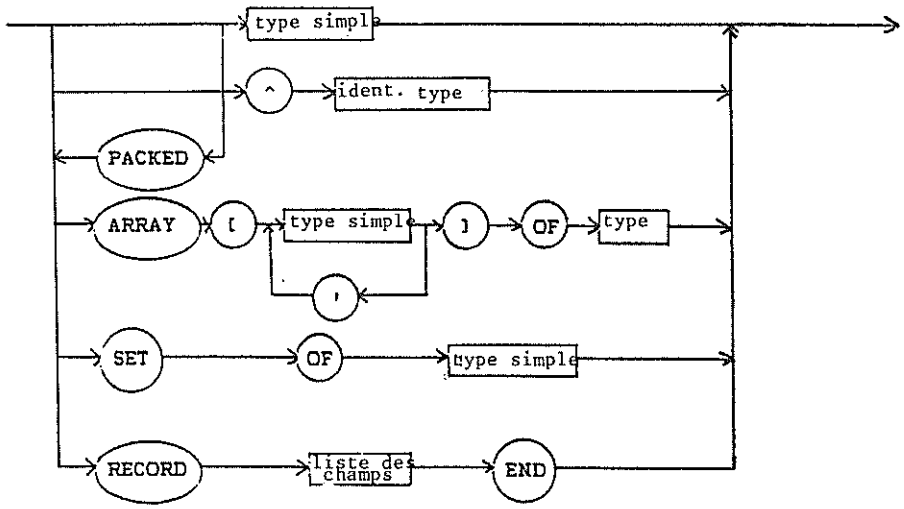
Exemple : CONST bs=CHR(10);
cr=CHR(13);

1.6 TYPE SIMPLE



Les types scalaires énumérés (identificateur, identificateur...) ne peuvent pas contenir plus de 256 éléments.

1.7 TYPES



Le mot réservé PACKED est accepté, mais il est ignoré puisque la compression a déjà lieu pour les tableaux de caractères, etc. Le seul cas dans lequel la compression des tableaux serait avantageuse est celui d'un tableau de Booléens; mais ceci s'exprime plus naturellement comme un ensemble, lorsque la compression est requise.

1.7.1 TABLEAUX et ENSEMBLES

Le type de base d'un ensemble peut avoir jusqu'à 256 éléments. Ceci permet de déclarer des ensembles de caractères avec des ensembles de n'importe quel type énuméré par l'utilisateur. Toutefois, il faut noter que seuls des sous-intervalles d'entiers peuvent être utilisés comme types de base. Tous les sous-ensembles d'entiers sont traités comme des ensembles de 0..255.

Les tableaux complets de tableaux, les tableaux d'ensembles, les enregistrements d'ensembles, etc. sont supportés.

Deux types de TABLEAUX sont seulement traités comme équivalents si leur définition découle du même emploi du mot réservé ARRAY. Ainsi, les types suivants ne sont pas équivalents :

```
TYPE
     tablea = ARRAY(1..100) OF INTEGER;
     tableb = ARRAY(1..100) OF INTEGER;
```


Ainsi, une variable du type tablea ne peut pas être affectée à une variable de type tableb. Ceci permet de détecter les erreurs comme le fait d'affecter deux tables représentant des données différentes. La limitation précédente ne s'applique pas au cas spécial des tableaux du type chaîne puisque les tableaux de ce type sont toujours utilisés pour représenter des données similaires.

1.7.1 Pointeurs

Pascal 4 Hisoft permet de créer des variables dynamiques grâce à la Procédure Standard NEW (Voir le Chapitre 2). Contrairement à une variable statique à laquelle de l'espace mémoire est alloué dans le bloc dans lequel elle est déclarée, une variable dynamique ne peut pas être citée directement par un identificateur puisqu'elle n'a pas d'identificateur; dans ce cas, une variable pointeur est utilisée. Cette variable - qui est une variable statique - contient l'adresse de la variable dynamique et la variable dynamique elle-même est acceptée en insérant un '^' après la variable pointeur. Des exemples d'emploi des types pointeur peuvent être étudiés à l'Annexe 4.

Quelques limitations s'appliquent à l'emploi des pointeurs dans Pascal 4 Hisoft. Ces limitations sont :

Des pointeurs de types qui n'ont pas été déclarés sont interdits. Ceci n'empêche pas de construire des structures de listes liées puisque les définitions des types peuvent contenir des pointeurs vers eux-mêmes, par exemple :

```
TYPE
     élément = RECORD
               valeur : INTEGER;
               prochain : ^élément
               END;

     lien = ^élément;
```

Les pointeurs vers les pointeurs sont interdits.

Les pointeurs désignant le même type sont considérés comme équivalents, par exemple

```
VAR
     premier : lien;
     courant : ^élément;
```

Les variables premier et courant sont équivalentes (c'est-à-dire que l'équivalence structurelle est utilisée) et peuvent être affectées l'une à l'autre ou comparées.

La constante pré-définie NIL est supportée et, lorsqu'elle est affectée à une variable pointeur, la variable pointeur est considérée comme ne contenant pas d'adresse.

1.7.4 RECORDS

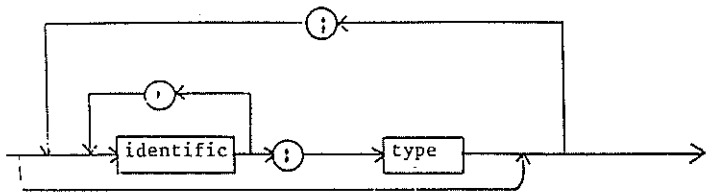
La mise en oeuvre des "RECORDS", c'est-à-dire de variables structurées composées d'un nombre fixe de constituants appelés champs, est dans Pascal 4 Hisoft comme dans le Pascal Standard, sauf que la partie variante de la liste des champs n'est pas supportée.

Deux types de "RECORDS" sont traités comme équivalents si leur déclaration provient de la même apparition du mot réservé RECORD. Voir le Paragraphe 1.7.1 ci-dessus.

L'instruction WITH peut être utilisée pour accéder aux différents champs d'un enregistrement, dans une forme plus compacte.

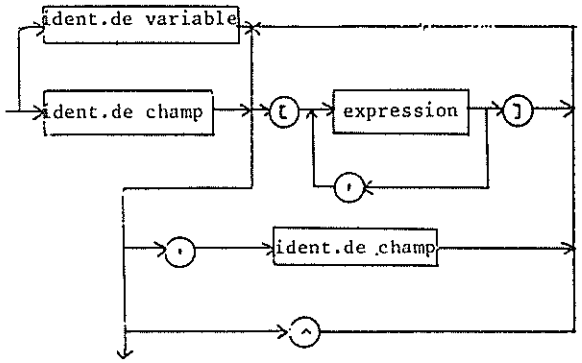
Voir à l'Annexe 4 un exemple d'emploi général de WITH avec RECORD.

1.8 LISTES DE CHAMPS



S'utilisent avec les ENREGISTREMENTS (RECORD); voir le Paragraphe 1.7.4 ci-dessus et l'Annexe 4 où figure un exemple.

1.9 VARIABLE



Deux types de variables sont supportés par Pascal 4 Hisoft; ces deux types sont les variables statiques et dynamiques. Les variables statiques se déclarent explicitement par VAR et de la mémoire leur est allouée pendant toute l'exécution du bloc dans lequel elles ont été déclarées.

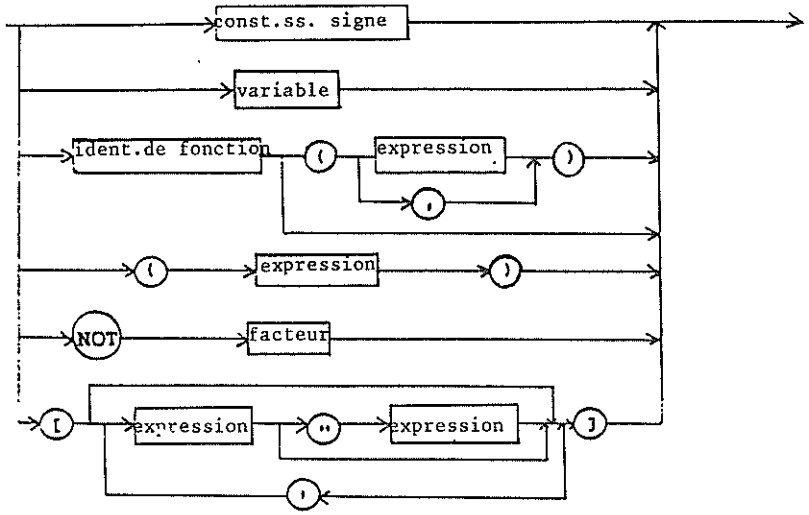
En revanche, les variables dynamiques sont créées dynamiquement par la procédure NEW, pendant l'exécution du programme. Elles ne sont pas déclarées explicitement et ne peuvent pas être citées par un identificateur. Elles sont citées indirectement par une variable statique du type pointeur contenant l'adresse de la variable dynamique.

Pour plus de détails sur l'emploi des variables dynamiques, voir le Paragraphe 1.7.2 et le Chapitre 2; un exemple d'utilisation de ces variables dynamiques est donné à l'Annexe 4.

Lorsqu'il spécifie les éléments de tableaux multi-dimensionnels, le programmeur n'est pas obligé d'employer la même forme de spécification d'index de la référence que celle utilisée dans la déclaration; ceci constitue un changement par rapport à Pascal 3 Hisoft.

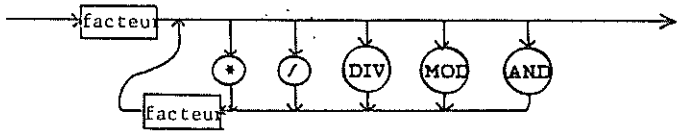
Exemple : si la variable a est déclarée ARRAY(1..10) OF ARRAY(1..10) OF INTEGER, a(1)(1) ou a(1,1) peut s'utiliser pour accéder à 1'élément (1,1) du tableau.

FACTEURS



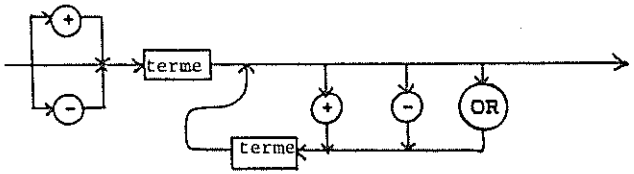
Pour plus de détails, voir EXPRESSIONS au Paragraphe 1.13 et FONCTIONS au Chapitre 3.

1.11 TERMES



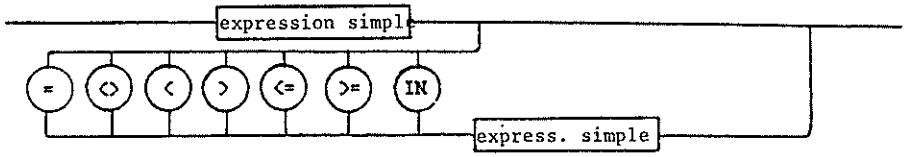
La limite inférieure d'un ensemble est toujours zéro et la taille d'un ensemble est toujours le maximum du type de base de l'ensemble. Ainsi, un JEU DE CARACTERES occupe toujours 32 octets (256 éléments possibles à raison d'un bit pour chaque élément). De même, SET OF 0..10 est équivalent à SET OFF 0..255.

1.12 EXPRESSION SIMPLE



Les commentaires du paragraphe 1.11 au sujet des ensembles s'appliquent aux expressions simples.

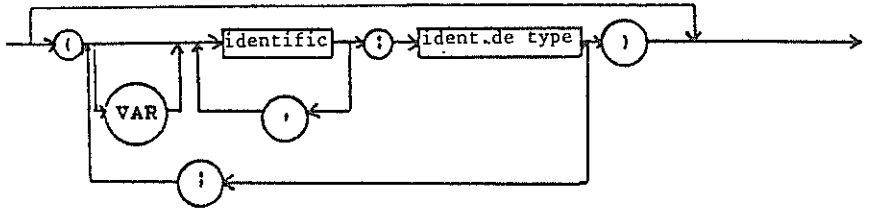
1.13 EXPRESSIONS



Avec IN, les attributs d'un ensemble sont l'intervalle complet du type de l'expression simple à l'exception des arguments entiers pour lesquels les attributs sont pris comme si le système avait détecté (0..255).

La syntaxe ci-dessus s'applique à la comparaison de chaînes de même longueur, de pointeurs et de tous les types scalaires. Les ensembles peuvent être comparés à l'aide des opérateurs >=, <=, <> ou = . Pour comparer les pointeurs, on utilise seulement = et <>.

1.14 LISTE DE PARAMETRES



Un identificateur de type doit être utilisé après le symbole deux points ; autrement une *ERREUR*44 est lancée.

Les paramètres de variables ainsi que les paramètres de valeurs sont intégralement supportés.

Les procédures et les fonctions ne peuvent pas être utilisées comme paramètres.

1.15 INSTRUCTIONS

Voir le schéma de syntaxe de la page 17 .

Instructions d'affectation :

Voir au paragraphe 1.17 les informations précisant les instructions d'affectation interdites.

Instructions CASE :

Une liste de CASE entièrement nulle est interdite, par exemple CASE OF END qui génère une erreur (*ERREUR*13).

La clause ELSE qui constitue une alternative à END est exécutée si le sélecteur ('expression' page suivante) n'est pas trouvé dans l'une des listes de CASE ('constante' page suivante).

Si le caractère de terminaison END est utilisé et que le sélecteur n'est pas trouvé, le contrôle est transmis à l'instruction qui se trouve après END.

Instructions FOR :

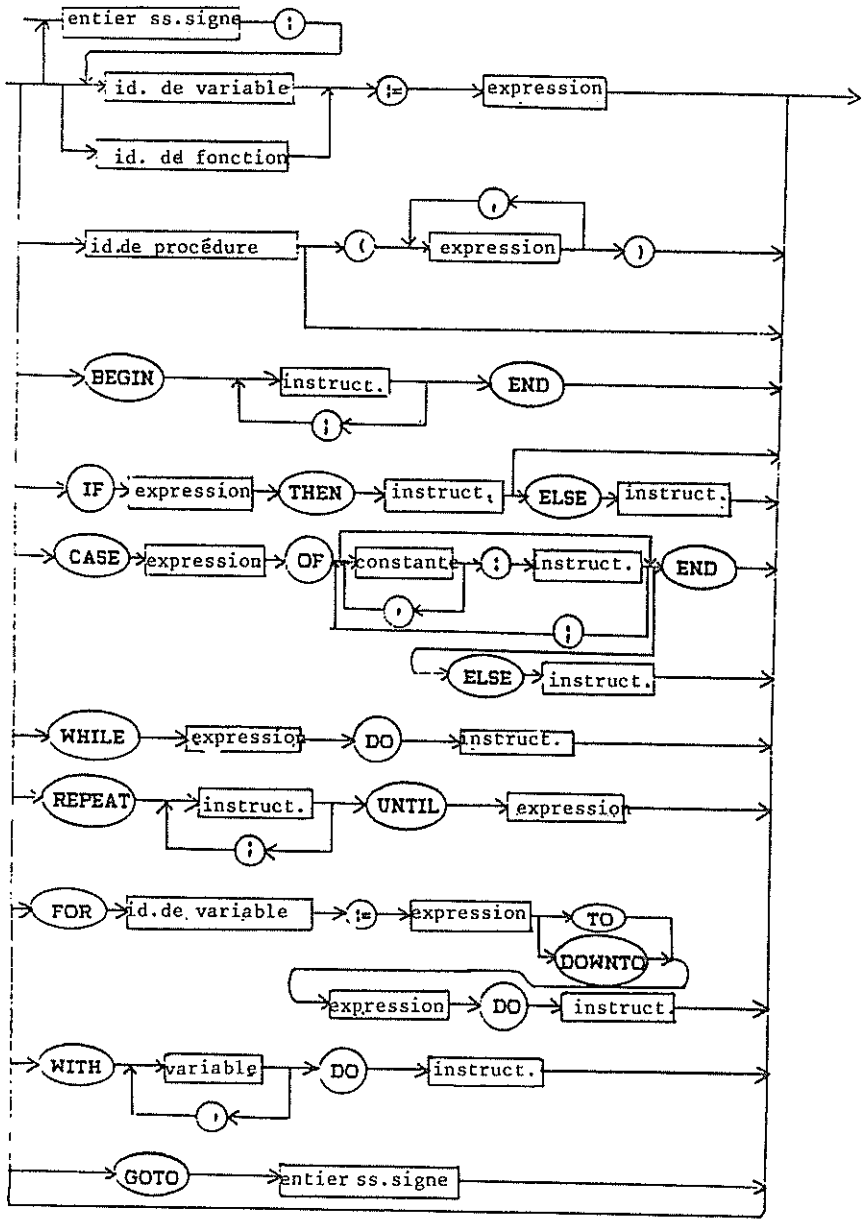
La variable de contrôle d'une instruction FOR peut seulement être une variable non structurée, mais pas un paramètre. Cette disposition est à mi-chemin entre les définitions de Jensen/wirth et du projet de norme de l'ISO.

Instructions GOTO :

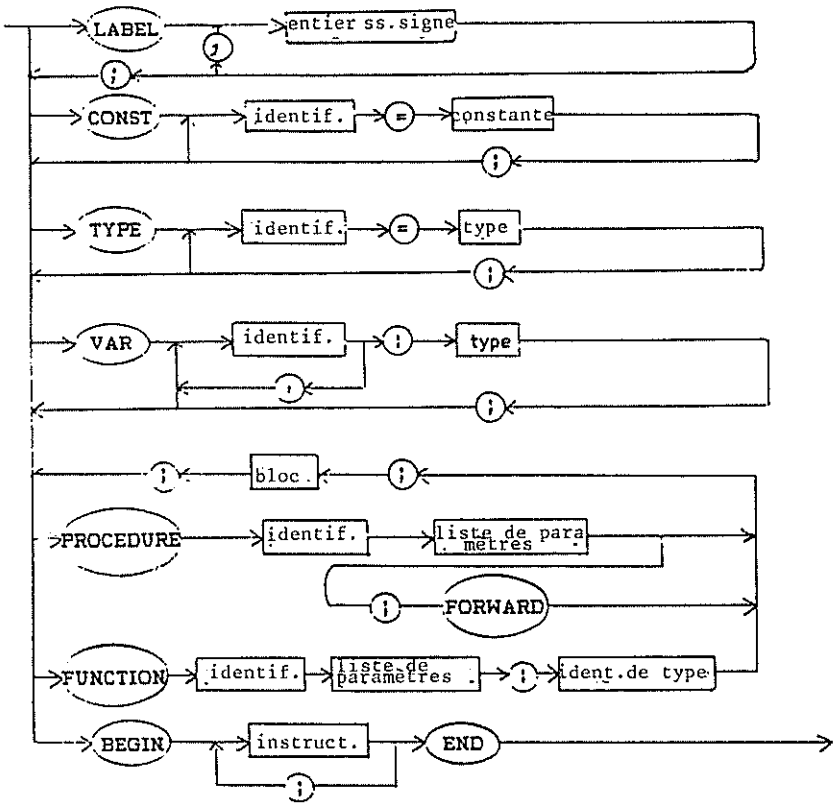
Il est seulement possible d'aller à (GOTO) une étiquette qui est présente dans le même bloc que l'instruction GOTO et au même niveau.

Les étiquettes doivent être déclarées (au moyen du mot réservé LABEL) dans le bloc dans lequel elles sont utilisées; une étiquette se compose d'un chiffre au moins et de quatre chiffres au plus. Lorsqu'une étiquette est utilisée pour désigner une instruction, elle doit être au début de l'instruction et être suivie du symbole ':'.

INSTRUCTION



1.16 BLOCS



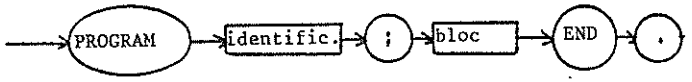
Références avant

Comme dans le Rapport et Manuel de l'Utilisateur de Pascal (Paragraphe 11.C.1), les procédures et fonctions peuvent être citées avant d'être déclarées, grâce au mot réservé FORWARD. Exemples :

```
PROCEDURE a(y:t) : FORWARD;  (déclaration de procédure à venir)
PROCEDURE b(x:t);            (après cette instruction)
     BEGIN
     ...
     a(p);                   (procédure a citée)
     ...
     END;
PROCEDURE a;          (déclaration effective de la procédure a)
     BEGIN
     ...
     b(q);
     ...
     END;
```

Noter que les paramètres et le type de résultat de la procédure a sont déclarés avec FORWARD et ne sont pas répétés dans la déclaration principale de la procédure. Il ne faut aps oublier que FORWARD est un mot réservé.

1.17 PROGRAMME



Puisque les fichiers ne sont pas mis en oeuvre, il n'y a pas de paramètres formels du programme

CHAPITRE 2 - IDENTIFICATEURS PREDEFINIS

2.1 CONSTANTES

MAXINT	Le plus grand entier disponible, c'est-à-dire 32767.
TRUE, FALSE	Les constantes de type booléen.

2.2 TYPES

INTEGER	Voir le Paragraphe 1.3
REAL	Voir le Paragraphe 1.3
CHAR	Le jeu de caractères ASCII étendus de 256 éléments
BOOLEAN	(TRUE, FALSE = vrai, faux). Ce type est utilisé dans les opérations logiques, y compris le résultat des comparaisons.

2.3 PROCEDURES ET FONCTIONS

2.3.1 Procédures d'entrée et de sortie

2.3.1.1 WRITE

La procédure WRITE sert à transmettre des données vers l'écran ou l'imprimante. Lorsque l'expression à écrire est simplement du type caractère, WRITE(e) transmet à l'écran ou à l'imprimante selon le cas la valeur à 8 bits représentée par la valeur de l'expression e.

Note :

CHR(8) (CTRL H) provoque un espacement arrière destructif sur l'écran.
CHR(19) (CTRL L) efface l'écran ou provoque un saut de page sur l'imprimante.
CHR(13) (CTRL M) provoque un retour chariot et un saut de ligne.
CHR(16) (CTRL P) provoque normalement l'acheminement de la sortie vers l'imprimante si l'écran est utilisé et vice versa.

Toutefois, de façon général :

WRITE(P1,P2,.....Pn); est équivalent à :

BEGIN WRITE(P1); WRITE(P2);.....; (WRITE(Pn) END;

Les paramètres P1, P2,...Pn de WRITE peuvent revêtir l'une des formes suivantes :

<e> ou <e:m> ou <e:m:n> ou <e:m:H>

où e, m et n sont des expressions et H est une constante littérale.

Il y a cinq cas à examiner :

1. e est du type entier: et soit <e> soit <e:m> est utilisée.

La valeur de l'expression entière e est convertie en une chaîne de caractères avec un espace de fin. La longueur de la chaîne peut être augmentée (avec des espaces de début) en employant m pour préciser le nombre total de caractères à sortir. Si m ne suffit pas pour le e à écrire ou si m n'est pas présent, e est écrit entièrement avec un espace de fin et m est ignoré. Il faut noter que si m est spécifié comme étant la longueur de e sans l'espace de fin, un espace de fin n'est pas sorti.

2. e est du type entier et la forme <e:m:H) est utilisée.

Dans ce cas, e est sorti en hexadécimal. Si m=1 et m=2, la valeur (e MOD 16^m) est sortie dans une largeur de m caractères exactement. Si m=3 ou m=4, la valeur complète de e est sortie en hexadécimal, en largeur 4 caractères. Si m>4, des espaces de début sont insérés avant la valeur hexadécimale complète de e si nécessaire. Des zéros de début sont insérés si nécessaire. Exemples :

```
WRITE(1025:m:H);

m = 1  donne en sortie : 1
m = 2  donne en sortie : 01
m = 3  donne en sortie : 0401
m = 4  donne en sortie : 0401
m = 5  donne en sortie : _0401
```

3. e est du type réel. Les formes <e>, <e:m> ou <e:m:n> peuvent être utilisées.

La valeur de e est convertie en une chaîne de caractères représentant un nombre réel. Le format de la représentation est déterminé par n.

Si n n'est pas présent, le nombre est sortie en notation scientifique, avec une mantisse et un exposant. Si le nombre est négatif, un signe moins est sorti avant la mantisse; autrement, un espace est fourni en sortie. Le nombre est toujours sorti avec au moins une position décimale et au maximum 5 positions décimales; l'exposant est toujours accompagné d'un signe (le signe plus ou le signe moins). Ceci veut dire que la largeur minimale de la représentation scientifique est de 8 caractères; si la largeur du champ m est inférieure à 8, la largeur complète de 12 caractères est toujours fournie en sortie. Si m>=8, une ou plusieurs positions décimales sont fournies en sortie, à concurrence d'un maximum de 5 positions décimales (m=12). Pour m>12, des espaces de début sont insérés avant le nombre. Exemples :

```
WRITE(-1,23E 10:m)

m=7  donne : -1.23000E+10
m=8  donne : -1.2E+10
m=9  donne : -1.23E+10
m=10 donne : -1.230E+10
m=11 donne : -1.2300E+10
m=12 donne : -1.23000E+10
m=13 donne :  -1.23000E+10
```

Si la forme <e:m:n> est utilisée, une représentation à marque décimale fixe du nombre e est écrite avec n spécifiant le nombre de positions décimales à sortir. Aucun espace de début n'est sorti, sauf si la largeur du champ m est assez grande. Si n est égal à zéro, e est sorti sous forme d'un entier. Si e est trop grand pour être sorti dans la largeur spécifiée pour le champ, la sortie se fait en format scientifique avec un champ de largeur m (voir ci-dessus). Exemples :

```
WRITE(1E2:6:2       donne: 100.00
WRITE(1E2:8:2)      donne: _100.00
WRITE(23.455:6:1)   donne: _23.5
WRITE(23.455:4:2)   donne: _2.34550E+01
WRITE(23.455:4:0)   donne: _23
```

4. e est du type caractère ou chaîne.

<e> ou <e:m> peut être utilisée et le caractère ou la chaîne de caractères sera sorti dans un champ de largeur minimale (1 pour les caractères) ou la longueur de la chaîne (pour les types chaîne). Des espaces de début sont insérés si m est assez grand.

5. e est de type Booléen.

La forme <e> ou <e:m> peut être utilisée et 'TRUE' ou 'FALSE' est fourni en sortie, selon la valeur booléenne de e, avec une largeur minimale de champ de 4 ou 5 respectivement.

2.3.1.2 WRITELN

WRITELN provoque un saut de ligne en sortie. Ceci est équivalent à WRITE(CHR(13)). Il faut remarquer qu'une avance de ligne est incluse.

WRITELN(P1,P2.....P3) est équivalent à :

```
BEGIN WRITE(P1,P2....P3); WRITELN END;
```

2.3.1.3 PAGE

La procédure PAGE est équivalente à WRITE(CHR(12)); elle provoque l'effacement de l'écran vidéo ou le passage à une nouvelle page de l'imprimante.

2.3.1.4 READ

La procédure READ sert à accéder des données à partir du clavier. L'opération se fait à l'aide d'un tampon conservé avec les programmes d'exécution - ce tampon est initialement vide (à l'exception d'une marque de fin de ligne). Nous pouvons considérer que tous les accès à ce tampon se font par une fenêtre de texte sur le tampon, par laquelle l'opérateur peut voir un caractère à la fois. Si la fenêtre de texte est positionnée sur une marque de fin de ligne, avant de terminer l'opération de lecture (READ), une nouvelle de ligne de texte sera mise dans le tampon à partir du clavier. Pendant la lecture de cette ligne, les différents codes de commande du paragraphe 0.0 sont reconnus par le système. Ainsi :

READ(V1,V2,....Vn); est équivalent à :

```
BEGIN READ(V1); READ(V2);...; READ(Vn) END;
```

où V1, V2 etc. peuvent être du type caractère, chaîne, entier ou réel.

L'instruction READ(V); a des effets différents selon le type de V. Quatre cas sont à envisager :

1. V est du type caractère.

Dans ce cas, READ(V) lit simplement un caractère du tampon d'entrée et l'affecte à V. Si la fenêtre de texte du tampon est positionnée sur une marque de ligne (un caractère CHR(13)), la fonction EOLN renvoie la valeur TRUE et une nouvelle ligne de texte est lue à partir du clavier. Lorsqu'une opération READ est exécutée ultérieurement, la fenêtre de texte est positionnée au début de la nouvelle ligne.

Note importante : Il faut se rappeler que EOLN est TRUE au début du programme. Ceci veut dire que si la première READ est du type caractère, une valeur CHR(13) sera renvoyée suivie par la lecture d'une nouvelle ligne à partir du clavier; une lecture ultérieure du type caractère renverra le premier caractère de cette nouvelle ligne, si ce n'est pas un blanc. Voir aussi la procédure READLN ci-dessous.

2. V est du type chaîne.

Une chaîne de caractères peut être lue à l'aide de READ et, dans ce cas, une série de caractères est lue jusqu'au moment où le nombre de caractères définis par la chaîne a été lu ou jusqu'au moment où EOLN = TRUE. Si la chaîne n'est pas remplie par la lecture (c'est-à-dire si la fin de la ligne est atteinte avant que toute la chaîne ait été assignée), la fin de la chaîne est remplie de caractères nuls (CHR(0)) - ceci permet au programmeur d'évaluer la longueur de la chaîne qui a été lue.

La note concernant 1. ci-dessus s'applique également ici.

3. V est du type entier.

Dans ce cas, une série de caractères qui représente un entier tel qu'il est défini au paragraphe 1.3 est lue. Tous les blancs précédents et toutes les marques de fin de ligne sont sautés (ceci signifie que les entiers peuvent être lus immédiatement, cf. la Note 1. ci-dessus).

Si l'entier lu a une valeur absolue supérieure à MAXINT (32567), l'erreur d'exécution 'Nombre trop grand' est lancée et l'exécution est clôturée.

Si le premier caractère lu, après le saut des espaces et des caractères de fin de ligne, n'est pas un chiffre ou un signe ('+' ou '-'), l'erreur d'exécution 'Nombre attendu' est lancée et le programme est clôturé.

4. V est du type réel.

Dans ce cas, une série de caractères représentant un nombre réel (conformément à la syntaxe du paragraphe 1.3) est lue.

Tous les espaces de début et les marques de fin de ligne sont sautés et, comme pour les entiers ci-dessus, le premier caractère suivant doit être un chiffre ou un signe. Si le nombre lu est trop grand ou trop petit (Voir le Paragraphe 1.3), une erreur 'Dépassement de capacité' est lancée; si 'E' est présent sans être suivi d'un signe ou d'un chiffre, le message d'erreur 'Exposant attendu' est généré et si une marque décimale est présente sans être suivie d'un chiffre, une erreur 'Nombre attendu' est lancée.

Comme les entiers, les réels peuvent être lus immédiatement; voir 1. et 3. ci-dessus.

2.3.1.5 READLN

READLN(V1,V2,.....Vn); est équivalent à :
BEGIN READ(V1,V2,....Vn); READLN END;

READLN lit simplement dans un nouveau tampon à partir du clavier; pendant l'entrée vers le tampon, l'utilisateur peut employer les différentes fonctions de commande du Paragraphe 0.0. Ainsi, EOLN devient FALSE après l'exécution de READLN, sauf si la ligne suivante est composée de blancs.

READLN peut être utilisée pour sauter la ligne vierge qui se trouve au début de l'exécution du code objet, c'est-à-dire que l'effet est une lecture d'un nouveau tampon. Ceci est intéressant si l'utilisateur souhaite lire un composant du type caractère au début d'un programme, mais n'est pas nécessaire pour lire un entier ou un réel (puisque les marques de fin de ligne sont sautées) ou pour lire des caractères provenant de lignes suivantes.

2.3.2 Fonctions d'entrée

2.3.2.1 EOLN

La fonction EOLN est une fonction booléenne qui renverrait la valeur TRUE si le caractère suivant à lire est un caractère de fin de ligne (CHR(13)). Autrement, la fonction renvoie la valeur FALSE.

2.3.2.2 INCH

La fonction INCH provoque un examen du clavier de l'ordinateur et, si une touche a été manoeuvrée, renvoie le caractère représenté par la touche utilisée. Si l'opérateur n'a appuyé sur aucune touche, CHR(0) est renvoyée. Par conséquent, cette fonction renvoie un résultat de type caractère.

2.3.3 Fonctions de transfert

2.3.3.1 TRUNC(X)

Le paramètre X doit être de type réel ou entier et la valeur renvoyée par TRUNC est le plus grand entier inférieur ou égal à X si X est positif et le plus petit entier supérieur ou égal à X si X est négatif. Exemples :

TRUNC(-1.5) renvoie -1 TRUNC(1.9) renvoie 1

2.3.3.2 ROUND(X)

X doit être de type réel ou entier et la fonction renvoie l'entier 'le plus proche' de X (conformément aux règles standard d'ajustement). Exemples :

```
ROUND(-6.5)  renvoie -6
ROUND(-651)  renvoie -7
ROUND(11.7)  renvoie 12
ROUND(23.5)  renvoie 24
```

2.3.3.3 ENTIER(X)

X doit être de type réel ou entier - ENTIER renvoie le plus grand entier inférieur ou égal à X, pour tous les X. Exemples :

```
ENTIER(-6.5)  renvoie -7
ENTIER(11.7)  renvoie 11
```

Note : ENTIER n'est pas une fonction du Pascal standard, elle est équivalente à la fonction INT de BASIC. Cette fonction est utile pour écrire des routines rapides destinées à de nombreuses applications mathématiques.

2.3.3.4 ORD(X)

X peut être de n'importe quel type scalaire à l'exception de réel. La valeur renvoyée est un entier représentant le nombre ordinal de la valeur de X dans l'ensemble définissant le type de X.

Si X est de type entier, ORD(X) = X; ceci doit habituellement être évité.

Exemples :

```
ORD('a') renvoie 97   ORD('a ') renvoie 64
```

2.3.3.5 CHR(X)

X doit être de type entier. CHR renvoie une valeur de caractère correspondant à la valeur ASCII de X. Exemples :

```
CHR(49) renvoie '1'    CHR(91) renvoie '('
```


2.3.4 Fonctions arithmétiques

Pour toutes les fonctions de ce sous-paragraphe, le paramètre X doit être de type réel ou entier.

2.3.4.1 ABS(X)

Renvoie la valeur absolue de X (par exemple ABS(-4.5) donne 4.5). Le résultat est du même type que X.

2.3.4.2 SQR(X)

Renvoie la valeur X*X, c'est-à-dire le carré de X. Le résultat est du même type que X.

2.3.4.3 SQRT(X)

Renvoie la racine carrée de X; la valeur renvoyée est toujours de type réel. Un message 'Erreur Appel Mathématique' est générée si l'argument X est négatif.

2.3.4.4. FRAC(X)

Renvoie la partie fractionnaire de X: FRAC(X) = X - ENTIER(X).

Comme pour ENTIER, cette fonction est utile pour écrire de nombreuses routines mathématiques rapides :

```
FRAC(1.5) renvoie 0.5  FRAC(-12.56) renvoie 0.44
```

2.3.4.5 SIN(X)

Renvoie le sinus de X, X étant en radians. Le résultat est toujours de type réel.

2.3.4.6 COS(X)

Renvoie le cosinus de X, X étant en radians. Le résultat est de type réel.

2.3.4.7 TAN(X)

Renvoie la tangente de X, X étant en radians. Le résultat est toujours de type réel.

2.3.4.8 ARCTAN(X)

Renvoie l'angle en radians dont la tangente est égale au nombre X. Le résultat est de type réel.

2.3.4.9 EXP(X)

Renvoie la valeur de e^X où e = 2.71828. Le résultats est toujours de type réel.

2.3.4.10 LN(X)

Renvoie le logarithme naturel (c'est-à-dire dans la base e) de X. Le résultat est de type réel. Si X <= 0, un message 'Erreur Appel Mathématique' est lancé.

2.3.5 Autres procédures prédéfinies

2.3.5.1 NEW(p)

La procédure NEW(p) alloue de l'espace pour une variable dynamique. La variable p est une variable pointeur et, après exécution de NEW(p), p contient l'adresse de la variable dynamique nouvellement allouée. Le type de la variable dynamique est le même que le type de la variable pointeur; tous les types sont autorisés.

Pour accéder à la variable dynamique, on utilise p^. Voir à l'annexe 4 un exemple d'emploi de pointeurs pour citer des variables dynamiques.

Pour ré-affecter l'espace utilisé pour des variables dynamiques, il faut employer les procédures MARK et RELEASE (voir ci-dessous).

2.3.5.2 MARK(v1)

Cette procédure sauvegarde l'état de la pile de variables dynamiques à sauvegarder dans la variable pointeur v1. L'état de la pile peut être remis à ce qu'il était au moment de l'exécution de la procédure MARK, en employant la procédure RELEASE (voir ci-dessous).

Le type de variable désigné par v1 est indifférent puisque v1 doit seulement être utilisé avec MARK et RELEASE, mais jamais avec NEW.

Voir à l'Annexe 4 un exemple de programme utilisant MARK et RELEASE.

2.3.5.3 RELEASE(v1)

Cette procédure libère de l'espace sur la pile à l'intention des variables dynamiques. La pile est remise dans l'état dans lequel elle se trouvait au moment de l'exécution de MARK(v1); il y a donc destruction réelle de toutes les variables dynamiques créées depuis l'exécution de la procédure MARK. De ce fait, cette fonction doit être utilisée avec précaution.

Pour plus de détails, voir ci-dessus et l'Annexe 4.

2.3.5.4 INLINE(C1,C2,C3,.....)

Cette procédure permet d'insérer dans le programme Pascal du code machine Z80; les valeurs (C1 MOD 256, C2 MOD 256, C3 MOD 256, ...) sont insérées dans le programme objet, à l'adresse courante du compteur de position tenu par le compilateur. C1, C2, C3, etc. sont des constantes entières dont le nombre est indifférent. Un exemple d'emploi de INLINE se trouve à l'annexe 4.

2.3.5.5 USER(V)

USER est une procédure contenant un argument entier V. Cette procédure provoque un appel de l'adresse mémoire donnée par V. Les entiers étant dans Pascal 4 Hisoft sous forme de complément à deux (Voir l'Annexe 3), pour citer des adresses supérieures à £7FFF (32767), il faut utiliser des valeurs négatives pour V. Par exemple £C000 est -16384 et, par conséquent, USER (-16384) appelle l'adresse mémoire £C000. Toutefois, lorsqu'on utilise une constante pour citer une adresse mémoire, il est plus pratique de travailler en hexadécimal.

La routine appelée doit se terminer par une instruction RET (£C9) Z80 et doit préserver le contenu du registre IX.

2.3.5.6 HALT

Cette procédure provoque l'arrêt de l'exécution du programme avec le message 'Arrêt à PC=XXXX' où XXXX est l'adresse mémoire en hexadécimal de la position de laquelle HALT a été lancée. Avec la liste de la compilation, HALT peut être utilisée pour déterminer lequel de deux chemins ou plus d'un programme est pris. Cette fonction s'utilise habituellement pendant la mise au point.

2.3.5.7 POKE(X,V)

POKE stocke l'expression V dans la mémoire de l'ordinateur, en commençant à l'adresse mémoire X. X est de type entier et V peut être de type quelconque sauf SET. Voir au Paragraphe

2.3.5.5 une discussion de l'emploi d'entiers pour représenter des adresses mémoire. Exemples :

POKE(£6000,'A') met £41 à la position £6000.
POKE(-16384,3.6E3 met 00 0B 80 70 (en hexadécimal) à la position £C000.

2.3.5.8 TOUT(NOM, DEBUT, TAILLE)

TOUT est la procédure utilisée pour sauvegarder des variables sur bande. Le premier paramètre est de type ARRAY(1..8) de CHAR et est le nom du fichier à sauvegarder. TAILLE octets de mémoire sont vidés à partir de l'adresse de DEBUT. Ces deux paramètres sont de type entier.

Par exemple, pour sauvegarder sur bande la variable V sous le nom 'VAR ' il faut utiliser :

```
TOUT('VAR    ',ADDR(V),SIZE(V))
```

L'emploi d'adresses mémoire réelles donne à l'utilisateur une souplesse beaucoup plus grande que la seule possibilité de sauvegardes des tableaux. Par exemple, si l'écran d'un système comporte une correspondance mémoire, des écrans complets peuvent être sauvegardés directement. Voir à l'Annexe 4 un exemple d'utilisation de la procédure TOUT.

2.3.5.9 TIN (NOM, DEBUT)

Cette procédure permet de charger à partir d'une bande des variables, etc. qui ont été sauvegardées à l'aide de TOUT. NOM est du type ARRAY(1..8) de CHAR et DEBUT est de type entier. La bande est examinée pour y trouver un fichier appelé NOM qui est ensuite chargé à l'adresse mémoire DEBUT. Le nombre d'octets à charger est pris sur la bande (sauvegardé sur la bande par la procédure TOUT).

Par exemple, pour charger la variable sauvegardée dans l'exemple du Paragraphe 2.3.5.8, ci-dessus, il faut utiliser :

```
TIN('VAR    ',ADDR(V))
```

Puisque les fichiers source sont enregistrés par l'éditeur en utilisant le même format que celui employé par TIN et TOUT, TIN peut servir à charger des fichiers de textes dans ARRAY de CHAR pour traitement (voir le Guide des Modifications HP4T).

Voir à l'Annexe 4 un exemple d'emploi de TIN.

2.3.5.10 OUT(P,C)

Cette procédure est utilisée pour accéder directement aux accès de sortie du Z80 sans employer la procédure INLINE. La valeur du paramètre entier P est mise dans le registre BC, le paramètre caractère C est chargé dans le registre A et l'instruction d'assemblage OUT(C),A est exécutée.

Exemple : OUT(1,'A') fait sortir le caractère 'A' par l'accès 1 du Z80.

2.3.6 Autres fonctions prédéfinies

2.3.6.1 RANDOM

Cette fonction renvoie un nombre pseudo-aléatoire de l'intervalle 0 - 255, bornes comprises. Bien que cette routine soit très rapide, elle donne des résultats médiocres lorsqu'elle est utilisée de façon répétée dans des boucles ne comportant pas d'opérations d'E/S.

Si l'utilisateur souhaite des résultats meilleurs que ceux de cette fonction, il doit écrire une routine (en Pascal ou en code machine) adaptée à l'application concernée.

2.3.6.2 SUCC(X)

X peut être de n'importe quel type scalaire sauf réel et SUCC(X) renvoie le successeur de X. Exemples :

SUCC('A') renvoie 'B' SUCC('5') renvoie '6'

2.3.6.3 PRED(X)

X peut être de n'importe quel type scalaire sauf réel; le résultat de la fonction est le prédécesseur de X. Exemples :

PRED('j') renvoie 'i' PRED(TRUE) renvoie FALSE

2.3.6.4 ODD(X)

X doit être de type entier. ODD renvoie un résultat booléen qui est TRUE si X est impair et FALSE si X est pair.

2.3.6.6 ADDR(V)

Cette fonction prend un identificateur variable de n'importe

quel type comme paramètre et renvoi un résultat entier qui est l'adresse mémoire de l'identificateur variable V. Pour plus de détails sur la façon dont les variables sont contenues à l'exécution dans Pascal 4 Hisoft, voir l'Annexe 3. Un exemple d'emploi de la fonction ADDR est donné à l'Annexe 4.

2.3.6.7 PEEK(X,T)

Le premier paramètre de cette fonction est de type entier, il sert à spécifier une adresse mémoire (voir le Paragraphe 2.3.5.5). Le deuxième argument est un type; c'est le type du résultat de la fonction.

PEEK sert à ressortir des données de la mémoire de l'ordinateur et son résultat peut être de n'importe quel type.

Dans toutes les opérations PEEK et POKE (l'opposé de PEEK), les données sont déplacées dans la représentation interne de Pascal 4 Hisoft (Voir l'annexe 3). Par exemple, si la mémoire à partir de £5000 contient les valeurs 50 61 73 63 61 6C (en hexadécimal), il vient :

```
WRITE(PEEK(£5000,ARRAY(1..6) OF CHAR)) donne 'Pascal'
WRITE(PEEK(£5000,CHAR)) donne 'P'
WRITE(PEEK(£5000,INTEGER)) donne 24912
WRITE(PEEK(£5000,REAL)) donne 2.46227E+29
```

Pour plus de détails sur la représentation des types dans Pascal 4 Hisoft, voir l'Annexe 3.

2.3.6.7 SIZE(V)

Le paramètre de cette fonction est une variable. Le résultat entier est la quantité de mémoire (en octets) occupée par cette variable.

2.3.6.8 INP(P)

INP permet d'accéder directement aux accès du Z80, sans utiliser la procédure INLINE. La valeur du paramètre entier P est chargée dans le registre BC et le résultat caractère de la fonction s'obtient en exécutant l'instruction IN A,(C) du langage d'assemblage.

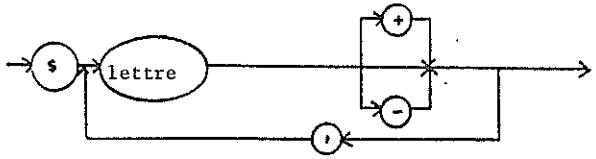
CHAPITRE 3 - COMMENTAIRES ET OPTIONS DU COMPILATEUR

3.1 Commentaires

Un commentaire peut apparaître entre deux mots réservés, nombres, identificateurs ou symboles spéciaux quelconques; voir l'annexe 2. Un commentaire commence par un caractère '{' ou par une paire de caractère '(* '. Sauf dans le cas où le caractère suivant est '$', tous les caractères sont ignorés jusqu'au prochain caractère '{' ou jusqu'à la prochaine paire de caractères '*)'. Si un caractère '$' a été trouvé, le compilateur recherche une série d'options compilateur (voir ci-dessous, après laquelle les caractères sont sautés jusqu'à détection de '{' ou de '*)'.

3.2 Options du compilateur

La syntaxe de spécification des options du compilateur est :



Options disponibles :

Option L :

Commande le listage par le compilateur du texte de programme et de l'adresse du code objet.

Si L+, la liste fournie est complète.
Si L-, les seules lignes listées sont celles pour lesquelles une erreur est détectée.

Valeur par défaut : L+

Option O :

Cette option détermine si certains contrôles de dépassement de capacité sont faits. Les multiplications et divisions d'entiers ainsi que toutes les opérations arithmétiques sur réels sont toujours l'objet d'un contrôle de dépassement de capacité.

Avec O+, les additions et soustractions d'entiers sont contrôlées.

Avec O-, les contrôles ci-dessus ne sont pas effectués.

Valeur par défaut : O+

Option C :

Cette option détermine si les contrôles clavier sont faits ou non pendant l'exécution des programmes en code objet. Avec C+, si l'opérateur appuie sur CC, l'exécution provoque un renvoi avec un message ARRET - Voir le Paragraphe 2.3.5.6.

Ce contrôle est fait au début de toutes les boucles, procédures et fonctions. Ainsi, l'utilisateur peut employer cette fonction pour détecter la boucle, etc. qui ne se termine pas correctement pendant la mise au point. Cette fonction doit être invalidée pour que le programme objet s'exécute rapidement.

Avec C-, le contrôle ci-dessus n'est pas fait.

Valeur par défaut : C+.

Option S :

Cette option détermine si les contrôles de la pile sont exécutés ou non.

Avec S+, au début de chaque appel de fonction et de procédure, un contrôle est exécuté pour savoir si un dépassement de capacité de la pile interviendra probablement dans ce bloc. Si la pile d'exécution déborde par rapport à la pile des variables dynamiques ou au programme, le message 'Hors RAM à PC=XXXX' est lancé et l'exécution est clôturée. Bien entendu, cette option n'est pas absolue; si une procédure fait largement appel à la pile dans elle-même, il se peut que le programme 's'effondre'. Alternativement, si une fonction fait très peu appel à la pile tout en utilisant des récurrences, il est possible que la fonction soit arrêtée inutilement.

Avec S-, aucun contrôle de la pile n'est exécuté.

Valeur par défaut : S+

Option A :

Cette option détermine si des contrôles sont faits pour s'assurer que les indices des tableaux sont dans les limites spécifiées dans la déclaration de chaque tableau.

Avec A+ et avec un indice de tableau trop haut ou trop bas, le message 'Indice trop haut' ou 'Indice trop bas' est lancé et l'exécution du programme est arrêtée.

Avec A-, ces contrôles ne sont pas faits.

Valeur par défaut : A+

Option I :

Lorsqu'on emploie l'arithmétique entière en complément à deux sur 16 bits, des dépassements de capacité interviennent à l'exécution d'opérations <, <, >=, ou <= si les arguments diffèrent de plus que MAXNT (32767). Dans ce cas, le résultat de la comparaison est incorrect. Ceci ne présente normalement pas de difficultés; toutefois, si l'utilisateur souhait comparer de tels nombres, l'emploi de I+ garantit que les résultats de la comparaison seront corrects. Une situation équivalente peut intervenir en arithmétique sur réels et, dans ce cas, une erreur de dépassement de capacité est lancée si la différence entre les arguments est supérieure à 3.4E38 environ; ceci est inévitable.

Avec I-, le résultat des comparaisons précédentes n'est pas vérifié.

Valeur par défaut : I-

Option P :

Avec l'option P, le dispositif auquel est transmis la liste de compilation est modifié; en d'autres termes, si l'écran a été utilisé, l'imprimante est mise en fonction et vice-versa. Il faut remarquer que cette option n'est pas suivie d'un signe '+' ou '-'.

Valeur par défaut : l'écran vidéo.

Option F :

La lettre de cette option doit être suivie d'un espace et d'un nom de fichier de huit caractères. Si le nom de fichier a moins de huit caractères, il doit être comblé avec des espaces.

La présence de cette option provoque l'inclusion du texte source Pascal du fichier spécifié, à partir de la fin de la ligne courante - ceci est utile si le programmeur souhaite construire sur bande une 'bibliothèque' de procédures et fonctions très utilisées pour les inclure ensuite dans certains programmes.

Le programme doit être sauvegardé à l'aide de la commande 'P' de l'éditeur. Dans la plupart des systèmes, l'option liste L doit être utilisée; autrement, la compilation ne sera pas assez rapide.

Exemple : { $L-,F MATRIX inclut le texte d'un fichier bande appelé MATRIX};

Lorsque le programmeur écrit de très grands programmes, il se peut qu'il n'y ait pas assez de place dans la mémoire de

l'ordinateur pour que le code objet et le code source y soient présents en même temps. Il est toutefois possible de compiler ces programmes en les sauvegardant sur bande et en employant l'option 'D'; dans ce cas, seulement 128 octets de la source sont dans la RAM à un moment quelconque, ce qui laisse beaucoup plus de place pour le code objet.

Les options du compilateur peuvent être utilisées sélectivement. Ainsi, les sections mise au point du code peuvent être accélérées et comprimées en inhibant les contrôles applicables tout en les préservant pour les parties non testées du code.

CHAPITRE 4 - EDITEUR INTEGRE

4.1 Présentation de l'Editeur

L'éditeur accompagnant toutes les versions de Pascal 4T Hisoft est un éditeur simple, ligne par ligne, conçu pour travailler avec tous les systèmes d'exploitation Z80 tout en demeurant facile à utiliser et capable d'éditer rapidement et efficacement les programmes.
Le texte est gardé en mémoire en forme comprimée. Le nombre d'espaces de début d'une ligne est conservé sous forme d'un caractère au début de la ligne et tous les mots réservés de HP4T sont rassemblés en un jeton d'un caractère. Ceci entraîne une réduction typique de la longueur du texte de 25% environ.

NOTE: Dans ce chapitre, la touche DELETE est citée à la place du code de commande CH. Ce remplacement a semblé plus naturel.

L'éditeur est accédé automatiquement lorsque HP4T est chargé à partir d'une bande; le message suivant est affiché :

Copyright Hisoft 1982
Tous droits réservés

suivi par le message d'édition '>'.

En réponse à cette interrogation, l'utilisateur peut entrer une ligne de commande dans le format :

C N1, N2, S1, S2

suivi de RETURN où :

C est la commande à exécuter (voir Paragraphe 4.2 ci-dessous).
N1 est un nombre entre 1 et 32767, bornes incluses.
N2 est un nombre entre 1 et 32767, bornes incluses.
S1 est une chaîne de caractères de longueur maximale 20.
S2 est une chaîne de caractères de longueur maximale 20.

La virgule est utilisée pour séparer les arguments différents (bien que ceci puisse être modifié - voir la commande 'S') et les espaces sont ignorés, sauf dans les chaînes. Aucun argument n'est obligatoire, mais certaines des commandes (par exemple la commande 'D') ne peuvent pas opérer si N1 et N2 ne sont pas spécifiés. L'éditeur se souvient des différents nombres et chaînes entrés et utilise ces valeurs antérieures, le cas échéant, si l'opérateur ne spécifie pas un argument particulier dans la ligne de commande. Les valeurs de N1 et N2 sont mises initialement à 10 et les chaînes sont initialement vides. Si la ligne de commande entrée est interdite, par exemple F-1,100,HELLO, elle est ignorée et le message 'Pardon?' est affiché; il faut alors refrapper la ligne correctement,

c'est-à-dire F1,100,HELLO. Ce message d'erreur est également affiché si la longueur de S2 est supérieure à 20; si la longueur de S1 dépasse 20, les caractères en excès sont ignorés.

Les commandes peuvent être entrées en minuscules ou en majuscules.

Pendant l'entrée d'une ligne de commande, toutes les fonctions de commande applicables décrites au Paragraphe 0.0 peuvent être utilisées, par exemple CX pour effacer jusqu'au début de la ligne.

Le paragraphe suivant donne la description détaillée des différentes commandes utilisables avec l'éditeur; il faut remarquer que dès qu'un argument est entre les symboles '<>', cet argument doit être présent pour que la commande puisse s'exécuter.

4.2 Commandes de l'éditeur

4.2.1 Insertion de textes

Des textes peuvent être insérés dans le fichier des textes en frappant un numéro de ligne, un espace puis le texte requis ou en employant la commande 'I'. Il faut remarquer que si l'utilisateur frappe un numéro de ligne suivi de RETURN (c'est-à-dire sans texte), cette ligne est supprimée du texte si elle existe. Pendant une entrée de texte, les fonctions de commande CX (effacement jusqu'au début de la ligne), CI (aller au prochain arrêt de tabulation), CC (revenir à la boucle de commande) et CP (basculer l'imprimante) peuvent être utilisées. La touche DELETE (ou BACKSPACE) provoque un espacement arrière destructeur (mais pas au delà du début de la ligne de texte). Le texte est mis dans un tampon interne de HP4T et, si ce tampon est plein, l'opérateur ne peut pas y entrer d'autres textes; il doit employer DELETE ou CX pour libérer de l'espace dans le tampon.

Commande: I n,m

L'emploi de cette commande permet de passer en mode insertion automatique: le système interroge l'utilisateur par des numéros de lignes, en commençant à n et en augmentant ensuite par pas de m. Il faut enter le texte requis après le numéro de ligne affiché, en utilisant les différents codes de commande si nécessaire et en terminant la ligne de texte par RETURN. Pour sortir de ce mode, il faut utiliser la fonction de commande CC (voir le paragraphe 0.0 et la note de mise en oeuvre applicable).

Si l'opérateur entre une ligne avec un numéro qui existe déjà dans le texte, la ligne existante est supprimée pour être remplacée par la nouvelle, dès que l'opérateur a appuyé sur RETURN. Si l'incrémentation automatique des numéros de lignes donne un numéro supérieur à 32767, le mode insertion est terminé automatiquement.

Si, pendant une entrée de texte, l'opérateur parvient à la fin d'une ligne de l'écran sans avoir entré 128 caractères (la taille du tampon), l'écran défile vers le haut et la frappe peut se poursuivre sur la ligne suivante; un décalage automatique entre alors en jeu pour le texte afin que les numéros des lignes soient effectivement séparés du texte.

4.2.2 Listage des textes

La commande 'L' permet d'inspecter les textes; le nombre de lignes affichées à un moment quelconque pendant l'exécution de cette commande est fixé initialement (voir votre Note de Mise en Oeuvre) mais peut être modifié à l'aide d'une commande 'K'.

Commande: L n,m

Cette commande provoque le listage du texte courant sur le terminal d'affichage, entre le numéro de ligne n et le numéro de ligne m, ces deux numéros inclus. La valeur par défaut de n est toujours 1 et la valeur par défaut de m est toujours 32767; en d'autres termes, les valeurs par défaut ne proviennent pas des arguments entrés antérieurement. Pour lister tout le contenu du fichier de textes, il suffit d'utiliser 'L' sans argument. Les lignes de l'écran sont formattées avec une marge de gauche afin que le numéro de chaque ligne soit affiché clairement. Le nombre de lignes de l'écran listées sur le dispositif d'affichage peut être contrôlé à l'aide de la commande 'K' - après avoir listé un certain nombre de lignes, la liste s'arrête (si elle n'est pas encore à la ligne numéro m); il faut alors lancer la fonction de commande CC pour revenir à la boucle de l'éditeur ou appuyer sur une autre touche pour continuer le listage.

Commande : K n

'K' précise le nombre de lignes de l'écran à lister sur le dispositif d'affichage avant la pause décrite en 'L' ci-dessus. La valeur (n MOD 256) est calculée et stockée. Par exemple, il faut utiliser K5 si l'on souhaite qu'une liste ultérieure produise cinq lignes d'écran à la fois.

4.2.3 Edition des textes

Lorsqu'un texte a été créé, il est inévitablement nécessaire d'éditer certaines lignes. Différentes commandes permettent de modifier, supprimer, déplacer et renuméroter les lignes :

Commande: D <n,m>

Toutes les lignes de n à m inclus sont supprimées du fichier des textes. Si m<m ou si moins de deux arguments sont spécifiés, aucune action n'intervient; cette particularité permet d'éviter les fautes d'inattention. Pour supprimer une seule ligne, il faut que m=n. Cette suppression peut également être faite en entrant simplement le numéro de la ligne et en appuyant ensuite sur RETURN.

Commande: M n,m

Ceci a pour effet que le texte de la ligne n est entré à la ligne m, supprimant le texte qui se trouve déjà à cet endroit. Il faut remarquer que la ligne n n'est pas modifiée. Cette commande permet donc de déplacer une ligne de texte pour la mettre dans une autre position du fichier des textes. Si la ligne numéro n n'existe pas, aucune action n'intervient.

Commande: N<n,m>

L'emploi de la commande 'N' a pour effet que le fichier des textes est renuméroté, le numéro de sa première ligne devenant n et les pas suivants de numérotation étant égaux à m. n et m doivent être présents et, si la renumérotation va avoir pour effet que le numéro d'une ligne devient supérieur à 32767, la numérotation initiale est préservée.

Commande: F n,m,f,s

Le texte existant dans l'intervalle n < x < m est examiné pour y rechercher une apparition de la chaîne f - la chaîne 'find'. Si cette apparition est trouvée, la ligne de texte correspondante est affichée et le système passe en mode Edition - voir plus loin. L'opérateur peut alors employer des commandes du mode Edition pour rechercher d'autres apparitions de la chaîne f dans l'intervalle de lignes définies ou pour remplacer l'apparition courante de f par la chaîne s (la chaîne de substitution) et rechercher l'apparition suivante de f; pour plus de détails, voir ci-dessous.

Nota : l'intervalle de lignes et les deux chaînes ont pu être établis précédemment par une autre commande de sorte qu'il peut suffire d'entrer 'F' pour lancer la recherche; pour mieux comprendre, voir l'exemple du Paragraphe 4.3.

Commande: E n

Cette commande permet d'éditer la ligne dont le numéro est n. Si n n'existe pas, le système ne fait rien; autrement, la ligne est copiée dans un tampon et affichée sur l'écran (avec le numéro de ligne), le numéro de ligne est affiché à nouveau sur la ligne et le système passe en mode Edition. Toutes les opérations suivantes d'édition ont lieu dans le tampon et non dans le texte lui-même. Ainsi, la ligne originale peut être récupérée à tout moment.

Dans ce mode, un pointeur imaginaire se déplace dans la ligne (en commençant par le premier caractère) et différentes sous-commandes sont supportées pour vous permettre d'éditer la ligne. Les sous-commandes sont :

' '(espace) - incrémente le pointeur du texte d'une unité, c'est-à-dire pour désigner le caractère suivant de la ligne. Il est impossible de dépasser la fin de la ligne.

DELETE (ou BACKSPACE) - décrémente d'une unité le pointeur du texte pour qu'il désigne le caractère précédent de la ligne. Il est impossible de reculer au delà du premier caractère de la ligne.

CI (fonction de commande) - fait avancer le pointeur du texte jusqu'à la prochaine position de tabulation, mais pas au delà de la fin de la ligne.

RETURN - termine l'édition de cette ligne en préservant toutes les modifications faites.

Q - arrête l'édition de cette ligne, c'est-à-dire sort du mode Edition en ignorant toutes les modifications effectuées et en laissant la ligne dans l'état dans lequel elle était avant le commencement des opérations d'édition.

R - recharge le tampon d'édition à partir du texte, c'est-à-dire oublie toutes les modifications faites sur cette ligne et remet la ligne dans l'état où elle était initialement.

L - liste le reste de la ligne en cours d'édition, c'est-à-dire le reste de la ligne au delà de la position courante du pointeur. Le système demeure en mode Edition, le pointeur étant repositionné au début de la ligne.

K - écrase (supprime) le caractère qui se trouve à la position courante du pointeur.

Z - supprime tous les caractères à partir de (et incluse) la position courante du pointeur jusqu'à la fin de la ligne.

F - trouve la prochaine apparition de la chaîne 'find' définie précédemment dans une ligne de commande (voir la commande 'F' ci-dessus). Cette commande provoque la sortie automatique de l'édition de la ligne courante (en conservant les modifications) si elle ne trouve pas une autre apparition de la chaîne 'find' dans la ligne courante. Si une apparition de cette chaîne est détectée dans une ligne ultérieure (dans l'intervalle de lignes spécifié précédemment), le mode Edition entre en jeu pour la ligne dans laquelle se trouve la chaîne. Nota : le pointeur du texte est toujours positionné au début de la ligne après qu'une recherche ait abouti.

S - provoque le remplacement de l'apparition courante de la chaîne 'string' par la chaîne de remplacement et exécute ensuite la sous-commande 'F', c'est-à-dire une recherche de l'apparition suivante de la chaîne 'find'. Ceci permet, avec la sous-commande 'F' ci-dessus, de progresser dans le fichier des textes en remplaçant optionnellement les apparitions de la chaîne 'find' par la chaîne de 'remplacement'; voir l'exemple du paragraphe 4.3.

Note Importante Dans la version courante de HP4T, il existe une erreur connue dans le fonctionnement de la sous-commande 'S' : cette sous-commande doit seulement être utilisée immédiatement après une commande 'F', une sous-commande 'F' ou une sous-commande 'S'. En pratique, ceci ne devrait poser aucun problème.

I - insère des caractères à la position courante du pointeur. Le système demeure dans ce sous-mode jusqu'au moment où l'opérateur appuie sur RETURN; dans ce cas, le système revient en mode Edition, le pointeur étant placé après le dernier caractère qui a été inséré. Si l'utilisateur appuie sur DELETE (ou BACKSPACE) dans ce sous-mode, le caractère qui est à gauche du pointeur est effacé du tampon tandis que l'emploi de la fonction de commande CI fait avancer le pointeur jusqu'à la prochaine position de tabulation, avec insertion d'espaces.

X - fait avancer le pointeur jusqu'à la fin de la ligne et provoque l'accès automatique au sous-mode insertion détaillé ci-dessus.

C - sous-mode changement. Ceci permet à l'utilisateur de surcharger le caractère qui se trouve à la position courante du pointeur et de faire ensuite avancer le pointeur d'un espace. Le système demeure en sous-mode changement jusqu'à la prochaine manoeuvre de RETURN qui remet le système en mode Edition, le pointeur étant alors positionné après le dernier caractère qui a été modifié. Dans ce sous-mode DELETE (ou BACKSPACE) décrémente seulement le pointeur d'un espace, c'est-à-dire le déplace vers la gauche alors que CI est sans effet.

4.2.4 Commandes de bande

Un texte peut être sauvegardé sur bande ou chargé à partir d'une bande au moyen des commandes 'P' et 'G' :

Commande: P n,m,s

L'intervalle de lignes défini par n < x < m est sauvegardé sur bande en format HP4T sous le nom de fichier spécifié par la chaîne s. Nota : ces arguments ont pu être établis par une commande précédente. Avant d'entrer cette commande, il faut

s'assurer que le magnétophone est sous-tension et en mode ENREGISTREMENT. Pendant la sauvegarde du texte, le message 'Occupé...' est affiché.

Commande: G,,s

La bande est examinée pour y trouver un fichier en format bande HP4T dont le nom de fichier est s. Pendant la recherche, le message 'Occupé...' est affiché. Si un fichier bande HP4T valable est trouvé, mais que son nom de fichier est erroné, le message 'Trouvé' suivi du nom de fichier trouvé sur la bande est affiché et la recherche est poursuivie. Lorsque le nom de fichier correct est trouvé, l'indication 'Trouvé' apparaît sur le' dispositif de listage et le fichier est mis en mémoire. Si une erreur est détectée pendant le chargement, le message 'erreur de total de contrôle' est lancé et le chargement est clôturé prématurément. Dans ce cas, il faut rebobiner la bande, appuyer sur PLAY et frapper à nouveau 'G'.

Si la chaîne s est vide, le premier fichier HP4T de la bande est chargé, indépendamment de son nom de fichier.

Pendant que la recherche sur bande, il est possible de clôturer le chargement en maintenant CC en position abaissée; cette manoeuvre interrompt le chargement et permet de revenir à la boucle principale de l'éditeur.

Nota : Si un fichier de textes est déjà présent, le fichier de textes qui est chargé à partir de la bande sera annexé au fichier existant et tout le fichier sera renuméroté à partir de la ligne 1, par pas de 1.

4.2.5 Compilation et exécution à partir de l'Editeur

Commande: C n

Ceci provoque la compilation du texte qui commence à la ligne portant le numéro n. Si l'opérateur ne spécifie pas un numéro de ligne, le texte est compilé à partir de la première ligne existante. Pour plus de détails, voir le Paragraphe 0.2.

Commande: R

Le code objet compilé précédemment sera exécuté mais seulement si la source n'a pas été agrandie entre temps - pour plus de détails, voir le paragraphe 0.2.

Commande: T n

Cette commande est appelée commande de "TRANSLATION'. Le source courant est compilé à partir de la ligne n (ou à partir du début si n est omis) et, si la compilation réussit, le système lance le message 'OK?'; si l'opérateur répond 'Y', le code objet produit par la compilation est mis à la fin des

programmes d'exécution (détruisant le compilateur) et les programmes d'exécution ainsi que le code objet sont vidés sur bande avec un nom de fichier égal à celui défini précédemment pour la chaîne 'find'. Il est alors possible, ultérieurement, de mettre ce fichier en mémoire à l'aide du programme de chargement HP4T; dans ce cas, il y a exécution automatique du programme objet. Il faut remarquer que le code objet est placé à et déplacé jusqu'à la fin des programmes d'exécution de sorte que, après une commande 'T', il faut recharger le compilateur; toutefois, ceci ne devrait pas poser de problèmes puisqu'il est vraisemblable que la traduction d'un programme est faite seulement lorsque celui-ci est entièrement opérationnel.

Si l'opérateur décide de ne pas continuer le vidage sur bande, il lui suffit d'entrer un caractère quelconque autre que 'Y' en réponse à la question 'OK?'; le contrôle est rendu à l'éditeur qui fonctionne toujours correctement puisque le code objet n'a pas été déplacé.

4.2.6 Autres commandes

Commande: B

Ceci rend simplement le contrôle au système d'exploitation. Pour plus de détails sur la façon de ré-accéder au compilateur, il faut se reporter au Guide des Modifications HP4T et à votre Note de Mise en Oeuvre.

Commande: O n,m

Il faut se souvenir que le texte est contenu en mémoire sous forme de jetons, les espaces de début étant comprimés en un seul compte de caractères et tous les Mots Réservés de HP4T étant réduits à un jeton d'un caractère. Toutefois, s'il y a quand même du texte en mémoire, peut-être en provenance d'un autre éditeur, et que ce texte n'est pas sous forme de jetons, on peut employer la commande 'O' pour le transformer en jetons. Le texte est alors lu dans un tampon dans sa forme normale puis remis dans le fichier sous forme de jetons; bien entendu, ceci peut demander un peu de temps. Un intervalle de lignes doit être spécifié; autrement, les valeurs entrées précédemment sont utilisées par défaut.

Commande: S,,d

Cette commande permet de changer le caractère de séparation qui sert à séparer les arguments de la ligne de commande. Au moment de l'accès à l'éditeur, la virgule ',' est considérée comme caractère de séparation. Elle peut être modifiée par l'emploi de la commande 'S' au premier caractère de la chaîne spécifiée d. Nota : il faut se souvenir du fait que lorsqu'un nouveau séparateur a été défini, il faut l'utiliser (même dans la commande 'S') jusqu'au moment où un autre séparateur est spécifié.

Nota : l'espace ne peut pas jamais être utilisé comme caractère de séparation.

4.3 Exemple d'utilisation de l'éditeur

Imaginons que l'utilisateur ait frappé le programme suivant (à l'aide de I10,10):

```
 10 PROGRAM BUBBLESORT
 20 CONST
 30   Size = 2000;
 40 VAR
 50   Number# : ARRAY [1..Size] OF INTEGER;
 60   I, Temp : INTEGER;
 70 BEGIN
 80   FOR I:=1 TO Size DO Number[I] := RANDOM;
 90   REPEAT
100     FOR I:=1 TO Size DO
110     Noswaps := TRUE;
120     IF Number[I] > Number[I+1] THEN
130       BEGIN
140         Temp := Number[I];
150         Number[I] := Number[I+1];
160         Number[I+1] := Temp;
170         Noswaps := FALSE
180       END
190   UNTIL Noswapss
200 END.
```

Ce programme contient les erreurs suivantes :

Ligne 10	Point virgule omis.
Ligne 30	Pas vraiment une erreur, mais imaginons que la taille voulue soit 100.
Ligne 100	La taille devrait être taille-1.
Ligne 110	Ceci devrait être à la ligne 95.
Ligne 190	Noswapss devrait être Noswaps.

De plus, des nombres variables ont été déclarés, mais toutes les citations portent sur des nombres. Enfin, la variable booléenne Noswaps n'a pas été déclarée.

Pour rectifier le tir, on peut opérer comme suit :

F60,200,Nombres,Nombres	et utiliser ensuite à plusieurs reprises la sous-commande 'S'.
E10	puis la séquence X; RETURN RETURN
E30	puis _____ K C 1; RETURN RETURN
F100,100,Taille,Taille-1	suivie de la sous-commande 'S'.
M110,95 110	. suivie de RETURN.
E190	puis X DELETE RETURN RETURN
65 Noswaps:BOOLEAN;	.
N10,10	pour renuméroter par pas de 10.

Il est fortement recommandé de bien travailler l'exemple précédent en utilisant l'éditeur; les choses vous paraîtront un petit peu embêtante au début si vous avez l'habitude des éditeurs sur écran, mais il ne vous faudra pas beaucoup de temps pour vous adapter.

ANNEXE 1 : ERREURS

A.1.1 Numéros des erreurs générées par le compilateur

1. Nombre trop grand
2. Point virgule attendu
3. Identificateur non déclaré
4. Identificateur attendu
5. Utiliser '=' et non ':=' dans une déclaration de constante
6. '=' attendu
7. Cet identificateur ne peut pas être au début d'une instruction
8. ':=' attendu
9. ')' attendu
10. Type erroné
11. '.' attendu
12. Facteur attendu
13. Constante attendue
14. Cet identificateur n'est pas une constante
15. 'THEN' attendu
16. 'DO' attendu
17. 'TO' ou 'DOWNTO' attendus
18. '(' attendu
19. Il est interdit d'écrire ce type d'expression
20. 'OF' attendu
21. ',' attendu
22. ':' attendu
23. 'PROGRAM' attendu
24. Variable attendue puisque le paramètre est un paramètre de variable
25. 'GEGIN' attendu
26. Variable attendue dans un appel de READ
27. Impossible de comparer des expressions de ce type
28. Devrait être type INTEGER ou type REAL
29. Impossible de libre ce type de variable
30. Cet identificateur n'est pas un type
31. Exposant attendu dans un nombre réel
32. Expression scalaire (non numérique) attendue
33. Les chaînes nulles sont interdites (il faut utiliser CHR(0)).
34. '}' attendu
35. '{' attendu
36. Le type d'indice des tableaux doit être scalaire
37. '..' attendu
38. '}' ou ',' attendu dans une déclaration de tableau (ARRAY)
39. Limite inférieure supérieure à limite supérieure
40. Ensemble trop grand (plus de 256 éléments possibles)
41. Le résultat de la fonction doit être un identificateur de type
42. ',' ou '}' attendu dans l'ensemble.
43. '..' ou ',' ou ' ' attendu dans l'ensemble
44. Le type du paramètre doit être un identificateur de type
45. Un ensemble nul ne peut pas être le premier facteur d'une instruction autre que d'affectation
46. Scalaire (y compris un réel) attendu

47. Scalaire (mais pas de réel) attendu
48. Ensembles incompatibles
49. '<' et '>' ne peuvent pas être utilisés pour comparer des ensembles.
50. 'FORWARD', 'LABEL', 'CONST', 'VAR', 'TYPE' ou 'BEGIN' attendu.
51. Chiffre hexadécimal attendu
52. Impossible d'exécuter POKE sur des ensembles
53. Tableau trop grand (> 64K)
54. 'END' ou ';' attendu dans une définition d'enregistrement
55. Identificateur de champ attendu
56. Variable attendue après 'WITH'
57. La variable dans WITH doit être du type RECORD
58. Un identificateur de champ n'a pas eu d'instruction WITH associée
59. Entier sans signe attendu après 'LABEL'
60. Entier sans signe attendu après 'GOTO'
61. Cette étiquette est au niveau erroné
62. Etiquette non déclarée
63. Le paramètre de SIZE doit être une variable
64. Les tests d'égalité peuvent seulement être utilisés pour les pointeurs
67. Le seul paramètre d'écriture pour les entiers avec deux ':' est e:m:H.
68. Une chaîne ne peut pas contenir un caractère de fin de ligne
69. Le paramètre de NEW, MARK ou RELEASE doit être une variable de type pointeur
70. Le paramètre de ADDR doit être une variable

A.1.2 Messages d'erreur à l'exécution

Lorsqu'une erreur d'exécution est détectée, l'un des messages suivants est lancé, suivi de 'à PC=XXXX' ou XXXX est la position de mémoire à laquelle l'erreur est intervenue. Souvent, la source de l'erreur est évidente; dans les autres cas, il faut consulter la liste de la compilation pour savoir à quel endroit du programme l'erreur est intervenue, en utilisant XXXX comme référence croisée. De temps à autre, ceci ne permet pas d'obtenir le résultat correct.

1. Arrêt
2. Dépassement de capacité
3. Hors RAM
4. / par zéro également produit par DIV
5. Indice trop bas
6. Indice trop haut
7. Erreur Appel Mathématique
8. Nombre trop grand
9. Nombre attendu
10. Ligne trop longue
11. Exposant attendu

Des erreurs d'exécution provoquent l'arrêt de l'exécution du programme.

ANNEXE 2 - MOTS RESERVES ET IDENTIFICATEURS PREDEFINIS

A.2.1 Mots réservés

AND	ARRAY	BEGIN	CASE	CONST	DIV	DO
DOWNTO	ELSE	END	FORWARD	FUNCTION	GOTO	IF
IN	LABEL	MOD	NIL	NOT	OF	OR
PACKED	PROCEDURE	PROGRAM	RECORD	REPEAT	SET	THEN
TO	TYPE	UNTIL	VAR	WHILE	WITH	

A.2.2 Symboles spéciaux

Les symboles suivants sont utilisés par Pascal 4 Hisoft, ils ont une signification particulière :

+	-	*	/		
=	<>	<	<=	>=	>
(	)	[	]		
{	}	(*	*)		
^	:=	.	,	;	:
'	..				

A.2.3 Identificateurs prédéfinis

Les entités suivantes peuvent être interprétées comme déclarées dans un bloc entourant tout le programme et elles sont donc disponibles dans toute le programme, sauf si elles sont redéfinies par le programmeur à l'aide d'un bloc intérieur.

Pour plus de détails, voir le Chapitre 2.

```
CONST       MAXINT = 32767;

TYPE        BOOLEAN = (FALSE, TRUE);
            CHAR (Jeu étendu de caractères ASCII);
            INTEGER = -MAXINT..MAXINT;
            REAL (Un sous-ensemble des nombres réels, voir le
            paragraphe 1.3)
```

PROCEDURE WRITE; WRITELN; READ; READLN; PAGE; HALT; USER; POKE; INLINE; OUT; NEW; MARK; RELEASE; TIN; TOUT;

FUNCTION ABS; SQR; ODD; RANDOM; ORD; SUCC; PRED; INCH; EOLN; PEEK; CHR; SQRT; ENTIER; ROUND; TRUNC; FRAC; SIN; COS; TAN; ARCTAN; EXP; LN; ADDR; SIZE; INP;

ANNEXE 3 - REPRESENTATION ET STOCKAGE DES DONNEES

A.3.1 Représentation des données

L'exposé suivant précise la façon dont les données sont représentées en interne par Pascal 4 Hisoft.

Les informations données sur la capacité de mémoire requise dans chaque cas devraient être utiles à la plupart des programmeurs (la fonction SIZE peut être utilisée, voir le Paragraphe 2.3.6.7); d'autres détails peuvent être nécessaires si l'on tente de fusionner des programmes Pascal et en langage machine.

A.3.1.1 Entiers

Un entier occupe deux octets de mémoire; les entiers sont représentés en forme de complément à 2.
Exemples :

```
   1   ≡   £0001
 256   ≡   £0100
-256   ≡   £FF00
```

Le registre standard du Z80 utilisé parle compilateur pour contenir les entiers est HL.

A.3.1.2 Caractères, booléens et autres scalaires

Ces éléments occupent chacun un octet de mémoire, en forme binaire pure, sans signe.

Caractères : Le code ASCII étendu à 8 bits est utilisé.

```
'E' ≡ £45
'[' ≡ £5B
```

Booléens :

```
ORD(TRUE) = 1     de sorte que TRUE est représenté par 1
ORD(FALSE) = 1    de sorte que FALSE est représenté par 0
```

Le registre standard du Z80 utilisé par le compilateur pour ce qui précède est A.

A.3.1.3 Réels

La forme (mantisse, exposant) utilisée est semblable à celle de la notation scientifique habituelle; toutefois, il s'agit de la forme binaire et non de la forme propre à la notation scientifique. Exemples :

$2 = 2*10^0$ ou $1.0_2 * 2$

$1 = 1*10^0$ ou $1.0_2 * 2^0$

$12.5 \equiv -1.25\times10^1$ or $\equiv -25\times2^{-1}$
$\equiv -11001_2 \times2^{-1}$
$\equiv -1.1001_2 \times2^3$
après normalisation.

$0.1 \equiv 1.0\times10^{-1}$ or $\frac{1}{10} \equiv \frac{1}{1010_2} \equiv \frac{0.1_2}{101_2}$

il faut donc faire maintenant des divisions binaires longues.

```
          0.0001100
      ------------------
101 | 0.100000000000000
        101
        110
        101
         1000
          101
```

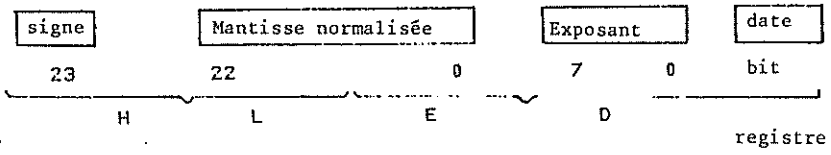
il apparait donc que la partie fractionnaire devient récurrente

$$= \frac{0.1_2}{101_2} = 0.000\dot{1}100\dot{0}_2$$

$$1.100\dot{1}100\dot{0} \times 2^{-4}$$

Réponse.

Comment utiliser les résultats ci-dessus pour représenter ces nombres dans l'ordinateur? Eh bien, il faut tout d'abord réserver quatre octets de mémoire pour chaque réel, dans le format suivant :



Signe : Le signe de la mantisse; 1 = négatif, 0 = positif
Mantisse normalisée : La mantisse normalisée a la forme 1.xxxxxx dont le bit haut (bit 22) est toujours à 1, sauf lorsqu'il représente zéro (HL=0,DE=0).
Exposant : Exposant en binaire en complément à 2.

Ainsi :

```
  2  = 0  1000000  00000000  00000000  00000001   (£40,£00,£00,£01)
  1  = 0  1000000  00000000  00000000  00000000   (£40,£00,£00,£00)
-12.5= 1  1100100  00000000  00000000  00000011   (£E4,£00,£00,£03)
  0.1= 0  1100110  01100110  01100110  11111100   (£66,£66,£66,£FC)
```

Si on se souvient que HL et DE servent à contenir des nombres réels, il faut charger les registres de la façon suivante pour représenter chacun des nombres précédents :

```
  2    =   LD   HL,£4000
           LD   DE,£0100

  1    =   LD   HL,£4000
           LD   DE,£0000

-12.5  =   LD   HL,£E400
           LD   DE,£0300

  0.1  =   LD   HL,£6666
           LD   DE,£FC66
```

Le dernier exemple montre pourquoi les calculs comportant des fractions binaires peuvent être imprécis; 0.1 ne peut pas être représenté de façon précise par une fraction binaire, avec un nombre fini de positions décimales.

N.B. Les réels sont stockés en mémoire dans l'ordre ED LH.

A.3.1.4 Enregistrements et tableaux

Un enregistrement utilise la même capacité de mémoire que le total de ses composants.

Tableaux : si n = nombre des éléments du tableau et
s = taille de chaque élément, il vient

le nombre d'octets occupés par le tableau et n*s.

Exemple
un ARRAY(1...10) OF INTEGER exige 10*2 = 20 octets
un ARRAY(2..12,1..10) OF CHAR exige 11*10=110 éléments, il a donc besoin de 110 octets.

A.3.1.5 Ensembles

Les ensembles sont stockés sous forme de chaînes binaires de sorte que si le type de base a n éléments, le nombre d'octets utilisés est le suivant :

(n-1) DIV 8 + 1. Exemples :

un SET OF CHAR exige (256-1) DIV 8 + 1 = 32 octets.
un SET OF (bleu, vert, jaune) exige (3-1) DIV 8 + 1 = 1 octet.

A.3.1.6 Pointeurs

Les pointeurs occupent deux octets qui contiennent l'adresse (en format Intel, c'est-à-dire octet bas d'abord) de la variable qu'ils désignent.

A.3.2 Stockage des variables à l'Exécution

Dans trois cas l'utilisateur doit avoir des informations sur la façon dont les variables sont stockées à l'exécution :

a. Variables globales - déclarées dans le bloc principal du programme
b. Variables locales - déclarées dans un bloc interne
c. Paramètres et valeurs renvoyées - transmis par et vers les procédures et fonctions.

Ces différents cas sont examinés ci-dessous et un exemple de la façon d'utiliser cette information est donnée à l'Annexe 4.

Variables globales

Les variables globales sont allouées du haut vers le bas de la pile d'exécution; par exemple, si la pile d'exécution est à £B000 et que les variables du programme principal sont :

```
VAR  i : INTEGER;
     ch : CHAR;
     x : REAL;
```

alors

i (qui occupe 2 octets - voir le chapitre précédent) est stockée aux positions µB000-2 et £B000-1, c'est-à-dire à £AFFE et £AFFF.

ch (un octet) est mise à la position £AFFE-1, c'est-à-dire à £AFFD.

x (quatre octets) est mise à £AFFF9, £AFFB et £AFFC.

Variables locales

Les variables locales ne peuvent pas être accédées très facilement par la pile. De ce fait, on établit le registre IX au début de chaque bloc interne pour que (IX-4) désigne le début des variables locales du bloc. Exemples :

```
PROCEDURE     test;
VAR           i,j: INTEGER;
```

il vient :

i (entier - c'est-à-dire 2 octets) est mise à IX-4-2 et IX-4-1, c'est-à-dire IX-6 et IX-5.
j est mise à IX-8 et IX-7.

Paramètres et valeurs renvoyées

Les paramètres des valeurs sont traités comme des variables locales et comme ces variables, plus un paramètre est déclaré tôt et plus haute est son adresse en mémoire. Toutefois, et contrairement aux variables, l'adresse la plus basse (pas la plus haute) est fixée, elle est fixée à (IX+2). Exemple :

```
PROCEDURE test(i ; REAL; j: INTEGER);
```

il vient :

j (alloué d'abord) est à IX+2 et IX+3.
i est à IX+4, IX+5, IX+6, et IX+7.

Les paramètres de variables sont traités comme les paramètres de valeurs sauf que deux octets leur sont toujours alloués et que ces deux octets contiennent l'adresse de la variable. Exemple :

```
PROCEDURE test(i : INTEGER; VAR x ; REAL);
```

il vient :

La citation de X est mise à IX+2 et IX+3; ces positions contiennent l'adresse à laquelle x est stocké. La valeur de i est à IX+4 et IX+5.

Les valeurs renvoyées des fonctions sont mises au dessus du premier paramètre en mémoire. Exemple :

```
FUNCTION test(i : INTEGER) : REAL:
```

il vient :

i est à IX+2 et IX+3 et un espace est réservé pour la valeur renvoyée à IX+4, IX+5, IX+6 et IX+7.

ANNEXE 4 - QUELQUES EXEMPLES DE PROGRAMMES HP4T

Les programmes suivants doivent être étudiés soigneusement, si vous avez un doute quelconque sur la façon de programmer en Pascal 4T Hisoft.

(Programme d'illustration de l'emploi de TIN et TOUT.
Le programme construit un répertoire téléphonique simple sur bande et le relit ensuite. Les recherches à effectuer doivent être écrites).

```
PROGRAM TAPE;
CONST
     Taille=10;

TYPE
     Entrée = RECORD
                Nom : ARRAY (1..10) OF CHAR;
                Nombre : ARRAY (1..10) OF CHAR
              END
VAR
     Répertoire : ARRAY (1...Taille) OF Entrée;
     I : INTEGER;
BEGIN
(établir le répertoire
     FOR I:  = 1 TO Taille DO
     BEGIN
        WITH Répertoire(I) DO
        BEGIN
             WRITE('Nom s'il vous plait');
             READLN:
             READ (Nom);
             WRITELN;
             WRITE('Numéro s'il vous plait);
             READLN;
             READ (Numéro)
             WRITELN
        END
     END;

(Pour vider le répertoire sur bande, il faut utiliser..)

     TOUT ('Répertoir', ADDR(Répertoire), SIZE(Répertoire))

(Pour relire le tableau, il faut procéder comme suit..)

     TIN('Répertoir', ADDR(Répertoire))

(Maintenant, le traitement du répertoire se fera en fonction de
vos indications..)
END.
```



```
   (Programme destiné à lister les lignes d'un fichier en
   ordre inversé
10 Ce programme montre les modalités d'emploi des pointeurs,
20 enregistrements ainsi que MARK et RELEASE
30 PROGRAM inversion de ligne;
40
50 TYPE elem=RECORD                    Crée une structure de liste liée)
60           next: ^elem;
70           ch: CHAR
80          END;
90     link=^elem;
100
110 VAR prev,cur,heap: link;           Tous les pointeurs à 'elem'
120
130 BEGIN
140  REPEAT                            Faire ceci plusieurs fois
150    MARK(heap);                     Affecter le haut de la pile à 'pile'
160    prev:=NIL;                      Ne désigne encore aucune variable
170    WHILE NOT EOLN DO
180       BEGIN
190         NEW(cur);                  Crée un nouvel enregistrement dynamique
200         READ(cur^.ch);             et affecte son champ à un
210                                    caractère du fichier.
220         cur^.next:=prev;           Les adresses du pointeur de ce champ
230         prev:=cur                  Enregistrement précédent
240       END;
250
260 (Ecrit les lignes à l'envers en examinant les enregistrements
270 établis en ordre inverse)
280
290    cur:=prev;
300    WHILE cur <> NIL DO             NIL est premier
310      BEGIN
320         WRITE(cur^.ch);            ECRIT ce champ, c'est-à-dire caractère
330         cur:=cur^.next             Adresse le champ précédent
340      END;
350    WRITELN;
360    RELEASE(heap);                  Libère l'espace des variables dynamiques
370    READLN                          Attend une autre ligne
380  UNTIL FALSE                       Utilise CC pour sortir
390 END.
```

```
(Programme pour montrer l'emploi des récurrences)

PROGRAM FACTOR;

(Ce programme calcule la factorielle d'un nombre entré par
le clavier (1) en utilisant une récurrence et (2) par
itération).
TYPE
  POSINT = 0..MAXINT;

VAR
  METHOD : CHAR;
  NUMBER : POSINT;

 Algorithme récurrent

FUNCTION RFAC(N : POSINT) : INTEGER;

 VAR F : POSINT;

 BEGIN
   IF N>1 THEN F:= N * RFAC(N-1)                    RFAC appelée N fois
          ELSE F:= 1;
   RFAC := F
 END;

 Solution itérative

FUNCTION IFAC(N : POSINT) : INTEGER;

  VAR I,F: POSINT;
  BEGIN
   F := 1;
   FOR I := 2 TO N DO F := F*I;                     Boucle simple
   IFAC:=F
  END;

BEGIN
  REPEAT
    (Préciser la méthode (I ou R) et la valeur numérique ')
    READLN;
    READ(METHOD,NUMBER);
    IF METHOD = 'R'
         THEN WRITELN(NUMBER,'! = ',RFAC(NUMBER))
         ELSE WRITELN(NUMBER,'! = ',IFAC(NUMBER))
  UNTIL NUMBER=0
END.
```



```
  (Programme dont le rôle est de vous montrez comment vous
  'salir les mains'! C'est-à-dire comment modifier les
  variables Pascal en utilisant du code machine. Ce
  programme démontre l'emploi de PEEK, POKE, ADDR et
  INLINE.)

PROGRAM divmult2;

VAR r:REAL;

FUNCTION divby2(x:REAL):REAL;                Fonction pour diviser par 2...
                                             ...rapidement
VAR i:INTEGER;
BEGIN
 i:=ADDR(x)+1;                               Désigne l'exposant de x
 POKE(i,PRED(PEEK(i,CHAR)));                 Décrémente l'exposant de x,
                                             voir l'Annexe 3.1.3.
 divby2:=x
END;

FUNCTION multby2(x:REAL):REAL;               Fonction pour multiplier par 2..
                                             ...rapidement
BEGIN
 INLINE(£DD,£34,3);                          INC (IX+3) - l'exposant de x
                                             - voir l'annexe 3.2.3
 multby2:=x
END;

BEGIN
 REPEAT
  WRITE(Entrer le nombre r)
  READ(r);                                   READLN est inutile - voir
                                             Paragraphe 2.3.1.4.
  WRITELN('r divided by two is',divby2(r):7:2);
  WRITELN('r multiplied by two is',multby2(r):7:2)
 UNTIL r=0
END.
```

ANNEXE 5

FONCTIONS SONORES ET GRAPHIQUES DU ZX SPECTRUM AVEC PASCAL 4T HISOFT

Ce document précise les modalités détaillées de la commande des fonctions sonores et graphiques du ZX SPECTRUM à l'aide des procédures Pascal du Pascal 4T Hisoft

1. Fonctions sonores

Les deux procédures suivantes (définies dans l'ordre précisé ci-dessous) sont nécessaires pour produire des sons avec HP4T.

(Cette procédure utilise le code machine pour prendre ses paramètres et elle les transmet ensuite à la routine BEEP de la mémoire ROM du SPECTRUM).

```
PROCEDURE

BEGIN
 INLINE(£DD, £6E, 2, £DD, £66, 3,          ( LD  L,(IX+2) : LD  H,(IX+3) )
        £DD, £5E, 4, £DD, £56, 5,          ( LD  E,(IX+4) : LD  D,(IX+5) )
        £CD, £B5, 3, £F3)                  ( CALL £3B5    : DI )

END;
```

(Cette procédure prend une fréquence zéro qu'elle convertit en une période de silence. Pour les fréquences différentes de zéro, la fréquence et la durée de la note sont converties approximativement dans les valeurs requises par la routine qui est dans la ROM du SPECTRUM et qui est ensuite appelée par BEEPER).

```
PROCEDURE BEEP (Fréquence : INTEGER; durée : REAL);
VAR I : INTEGER;

BEGIN
  IF Fréquence= 0 THEN FOR I:=1 TO ENTIER(1200 durée) DO

  ELSE BEEPER( ENTIER(fréquence durée),
ENTIER(437500/Fréquence - 30.125)

FOR I : = 1 TO 100 DO    (délai court entre les notes)
END.
```

Exemple d'emploi de BEEP:

```
BEEP ( 262, 0,5)
BEEP (0,1 ); (fait retentir la note Mi pendant une demi
seconde suivie d'un silence d'une seconde
```


2. Fonctions Graphiques

Trois procédures graphiques sont précisées : la première trace une coordonnée donnée (X,Y) alors que la deuxième et la troisième permettent de dessiner des lignes entre la position courante de traçage et une nouvelle position qui est définie par rapport à la position courante de traçage et qui devient ensuite la position courante de traçage.

Les procédures PLOT et LINE utilisent une variable booléenne, EN, qui, si elle est vraie, a pour effet qu'un point quelconque est tracé, indépendamment de l'état du pixel de cette position de traçage et qui, si elle est fausse, provoque l'inversion du pixel qui est déjà présent à cette position de traçage, c'est-à-dire que s'il est EN, il devient HORS, et vice-versa. Cet effet est identique à celui de la commande OVER SPECTRUM.

(Procédure constituant un "miroir" par rapport à la commande PLOT du BASIC. Elle trace simplement le point X,Y EN/HORS, selon que le premier paramètre est vrai ou faux).

```
PROCEDURE PLOT(  ON: BOOLEAN; X, Y : INTEGER);

BEGIN
 IF ON THEN WRITE( CHR(21), CHR(0))
      ELSE WRITE( CHR(21), CHR(1));

 INLINE(£FD, £21, £3A, £5C,                ( LD   IY,£5C3A )
        £DD, £46, 2, £DD, £4E, 4            ( LD   B,(IX+2) : LD  C,(IX+4) )
        £CD, £E5, £22);                     ( CALL £22E5  ;ROM PLOT routine)

END;
```

(Appelée par la procédure LINE, LINE1 sert à transmettre les arguments corrects à la routine DRAW qui est dans la mémoire ROM du SPECTRUM).

```
PROCEDURE LINE1

BEGIN
 INLINE(£FD, £21, £3A, £5C,                ( LD   IY,£5C3A )
        £DD, £56, 2, £DD, £5E, 4,           ( LD   D,(IX+2) : LD  E,(IX+4) )
        £DD, £46, 6, £DD, £4E, 8,           ( LD   B,(IX+6) : LD  C,(IX+8) )
        £CD, £BA, £24)                      ( CALL £24BA    ;ROM DRAW routine.)
END;
```

(LINE dessine une ligne entre la position courante de traçage (x,y) et (X+x, Y+y). La ligne peut être 'en' ou 'hors' selon la valeur du paramètre BOOLEEN EN)

PROCEDURE LINE

```
VAR
  SGNX, SGNY : INTEGER;

BEGIN
 IF ON THEN WRITE( CHR(21), CHR(0))
      ELSE WRITE( CHR(21), CHR(1));

 IF X<0 THEN SGNX:=-1 ELSE SGNX:=1;
 IF Y<0 THEN SGNY:=-1 ELSE SGNY:=1;

 LINE1( ABS(X), ABS(Y), SGNX, SGNY)
END;
```

(La routine DRAW qui doit être appelée dans LINE1 a besoin des valeurs absolues de X et Y et leur signes)
(Trace la ligne)

Exemple d'emploi de PLOT et de LINE:

```
PLOT (ON,
LINE( ON,      (trace une ligne de (50?50) à (150,0)
```

3. Sortie par la mémoire ROM

Dans certains cas, il est utile d'obtenir une sortie directe par la routine RST £10 de la ROM plutôt qu'avec WRITE(LN). Par exemple, lorsque l'on utilise le code de commande PRINT AT - ce code devrait être suivi de deux valeurs de 8 bits donnant la coordonnée (X,Y) à laquelle la position d'impression va être mise. Toutefois, si cette opération est faite à l'aide d'une instruction WRITE de Pascal, certaines valeurs de X et Y (par exemple 8 est interprété comme un espace arrière par HP4T) ne sont pas transmises à la ROM et, de ce fait, la modification de la position d'impression est incorrecte.

Cette difficulté peut être résolue à l'aide de la procédure suivante :

(SPOUT sort le caractère transmis comme paramètre directement par la routine RST £10 et évite donc le déroutement par HP4T de la valeur sortie)

```
PROCEDURE SPOUT ( C : CHAR );

 BEGIN
  INLINE(£FD, £21, £3A, £5C,          ( LD  IY,£5C3A )
         £DD, £7E, 2,                 ( LD  A,(IX+2) )
         £D7 )                        ( RST £10 )
 END;
```

Exemple d'emploi de SPOUT :

```
SPOUT (CHR(22) ; SPOUT ( CHR(8) ); SPOUT ( CHR(13) )
```

(met la position d'impression à la ligne 8 de la colonne 13).

HISOFT espère que vous trouverez utiles les routines ci-dessus et qu'elles amélioreront votre façon d'utiliser le Pascal 4T Hisoft. Comme d'habitude, nous sommes ouverts à tous commentaires concernant les informations figurant dans cette fiche.

ERRATA

PASCAL 4T HISOFT - VERSION 1.4

Depuis le 31 Octobre 1982, le numéro de version de Pascal 4D et Pascal 4T Hisoft est maintenant 1.4.

Les différences entre la version précédente de Pascal 4 Hisoft et la version 1.4 sont précisées ci-dessous :

1. Une erreur dans l'évaluation des expressions (qui provoquaient l'évaluation incorrecte d'expressions comme 1+SQR(2)) a été corrigée.

2. Une erreur dans l'évaluation des expressions (qui provoquaient l'évaluation incorrecte d'expressions comme I+(I+1)) a été corrigée.

3. Une erreur qui provoquait l'évaluation incorrecte du résultat d'une comparaison entier-réel a été corrigée.

4. Pascal 4T seulement. L'erreur documentée dans la sous-commande 'S' de l'éditeur (voir la page 38 du Manuel du Programmeur) a été corrigée - désormais, la sous-commande 'S' peut être utilisée à tout moment;

5. Pascal 4T seulement. Désormais, l'emploi de la commande 'F' à partir de l'éditeur pour trouver une chaîne de caractères a pour effet de positionner le curseur d'édition au début de toute apparition trouvée de la chaîne, voir les pages 37-38 du Manuel du Programmeur.

6. Pascal 4T seulement. Une nouvelle commande d'édition a été incluse. La commande 'X' affiche, en hexadécimal, l'adresse courante de fin du compilateur. Cette information permet à l'utilisateur de faire un exemplaire de travail du compilateur et minimise donc le risque d'abimer la bande de référence. Nota : HISOFT autorise l'utilisateur à faire un seul exemplaire de travail de HP4T. Pour faire un exemplaire de travail, il faut d'abord trouver l'adresse de début du logiciel en lisant la note de mise en oeuvre et trouver ensuite l'adresse de fin de ce même logiciel à l'aide d'une commande 'X'. Il faut ensuite lancer la commande applicable du système d'exploitation pour vider ce bloc de mémoire sur bande. Il faut remarquer que l'on n'emploie pas le programme de chargement HP4T pour charger le compilateur, etc. lorsque la sauvegarde a été effectuée de cette façon; en revanche, il faut utiliser la commande applicable de chargement à partir d'une bande du système d'exploitation et exécuter ensuite un démarrage à froid de l'éditeur, comme il est précisé dans la note de mise en oeuvre.

Note : Les utilisateurs du ZX SPECTRUM ne peuvent pas utiliser la méthode précédente pour faire une copie, ils doivent consulter leur note de mise en oeuvre qui donne tous les détails permettant de faire un exemplaire de travail de HP4T.

Comme d'habitude, Hisoft accueillera favorablement toutes les demandes concernant la présente note.

PASCAL 4T HISOFT - VERSION 1.5

A partir du ler Avril 1983, le numéro de version de Pascal 4T Hisoft sera 1.5. Les installations dotées de versions antérieures de HP4T peuvent passer à la version 1.5 en renvoyant l'ancienne bande magnétique accompagnée d'un montant forfaitaire de £3 + TVA. Les différences entre la version 1.5 et la version 1.5 sont les suivantes :

1. Les fonctions peuvent maintenant renvoyer un résultat POINTEUR.

2. Une erreur de la procédure pré-établie NEW qui provoquait de temps à autre une attribution incorrecte de mémoire pour une variable dynamique a été corrigée.

3. Une nouvelle commande d'éditeur a été incluse. La commande 'V' ne prend aucun argument, elle affiche simplement les valeurs par défaut courantes de l'intervalle de lignes, la chaîne FIND et la chaîne SUBSTITUTE. L'intervalle courant par défaut est présenté en premier suivi des deux chaînes (qui doivent être vides) sur des lignes séparées. Il faut se souvenir que certaines commandes de l'éditeur (comme 'D' et 'N') n'emploient pas ces valeurs par défaut mais doivent comporter des valeurs spécifiées sur la ligne de commande - voir le Manuel du Programmeur de HP4T.

4. (ZX SPECTRUM seulement). L'option 'INCLUDE' ('$F', voir le Paragraphe 3.2 du Manuel du Programmeur de HP4T) est maintenant disponible sur la version de HP4T destinée au ZX SPECTRUM. Pour utiliser cette option, le texte source qui doit être ultérieurement 'inclus' doit être vidé à l'aide d'une nouvelle commande de l'éditeur - la commande 'W' (SPECTRUM seulement). La commande 'W' opère exactement comme 'P' mais elle ne provoque pas un vidage en format bande standard HP4T, elle commence à sauvegarder le texte sur bande dès que la ligne de commande est terminée - pour cette raison, le magnétophone doit être en mode Enregistrement avant d'appuyer sur ENTER, à la fin de la ligne de commande. Lors d'un vidage de texte à l'aide de 'W' ou lors de lecture de texte à l'aide de l'option 'F' du compilateur, il est possible de manoeuvrer la touche BREAK à un moment quelconque; l'emploi de cette touche permet de revenir à l'éditeur.

Il faut remarquer que si l'on souhaite lire du texte à partir de l'éditeur, par la commande 'G' de l'éditeur, le texte doit avoir été vidé en sortie à l'aide de la commande 'P' et non de 'W'.

Exemple d'emploi :
Pour écrire une section d'un programme, il faut utiliser :

```
W50,120,PLOT      ; écrit la procédure PLOT.
```


Pour 'inclure' cette section dans un autre programme :

```
100 END;
110
120 { $F PLOT  'inclut la procédure PLOT à cet endroit }
130
140 PROCEDURE MORE;   (reste du programme)
150
```

Hisoft espère que les corrections et améliorations précédentes amélioreront l'emploi du HP4T et, comme d'habitude, espère recevoir de la correspondance au sujet de ce document.

BIBLIOGRAPHIE

K,Jensen et N.Wirth	RAPPORT ET MANUEL UTILISATEUR DE PASCAL Springer-Verlag 1975
W. Findlay et D.A. Watt	INTRODUCTION A LA PROGRAMMATION SYSTEMATIQUE - Pitman Publishing 1978
J. Tiberghien	GUIDE DU LANGAGE PASCAL SYBEX 1981
J. Welsh et J. Elder	PRESENTATION DU LANGAGE PASCAL

Le premier ouvrage et le troisième sont utiles comme références, alors que le deuxième et le quatrième sont des présentations du langage destinées aux débutants.