ATARI ST+STE

MISE À JOUR

GFA BASIC 3.5

DU GFA BASIC 3.0 ET DU GFA COMPILATEUR 3.0 VERS LA VERSION 3.5.

UN LOGICIEL GFA

EDITIONS MICRO APPLICATION

ATARI ST+STE

MISE À JOUR

GFA BASIC 3.5.F

DU GFA BASIC 3.0 ET DU GFA COMPILATEUR 3.0 VERS LA VERSION 3.5.

UN LOGICIEL GFA

Le vieux Manuel

EDITIONS MICRO APPLICATION

Frank OSTROWSKI
Gottfried P. ENGELS

Le vieux Manuel

UPGRADE

GFA Basic
GFA Compilateur
3.5

Editions MICRO APPLICATION

MICRO APPLICATION
58, Rue du Faubourg Poissonnière
75010 PARIS

© Reproduction interdite sans l'autorisation de
MICRO APPLICATION

'Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (Loi du 11 Mars 1957, article 40, 1er alinéa).

Cette représentation ou reproduction illicite, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La Loi du 11 Mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration'.

© 1989 GFA Systemtechnik GmbH
Heerdter Sandberg 30-32
D-4000 Düsseldorf 11

© 1989 MICRO APPLICATION
58 Rue du Faubourg Poissonnière
75010 PARIS

Auteurs : Frank Ostrowski - Gottfried P. Engels

Collection dirigée par Mr Philippe OLIVIER
Edition réalisée par Frédérique BEAUDONNET

GFA Basic 3.xx est une marque déposée de GFA Systemtechnik GmbH.
Atari, ST et TOS sont des marques déposées par Atari Corp.

Les programmes compilés en GFA Basic peuvent être distribués sans verser de royalties aux sociétés GFA Systemtechnik et Micro Application.

Dans le programme, sur la disquette ou dans le manuel, doit être indiqué que l'application est programmée en GFA Basic 3.xx.

Le vieux Manuel

Sommaire

1. Les nouvelles possibilités de l'éditeur 9

2. Opérations linéaires sur des vecteurs et des matrices 11

2.1. Initialisation 11

2.2. Opérations unaires (sur une matrice) 12

2.3. Opérations d'entrée-sortie 13

2.4. Opérations binaires (sur deux matrices) 15

3. Autres fonctions de la version 3.5 39

3.1. Combinatoire 39

3.2. Pointeurs sur DATA 40

Le vieux Manuel

Introduction

Vous venez d'acquérir la version 3.5 des nouveaux interpréteurs et compilateurs GFA Basic. Vous êtes donc déjà en possession des versions précédentes pour chacun des deux logiciels (de la 3.01 à la 3.07 pour le GFA Basic, et 3.01 à 3.03 pour le compilateur).

Lorsque vous découvrirez ce langage, votre première réaction pourrait être de penser : pourquoi une version 3.5 et non pas 3.8 ou 4.0 ?

Disons que les numéros de version des logiciels sont à l'image des gammes que proposent les constructeurs d'automobiles. Un changement d'unité correspond à un modèle radicalement différent de ses prédécesseurs : nouveau moteur, nouvelle carrosserie, nouveaux équipements.

Dans le cas du GFA Basic, nous avons ajouté des équipements de choix un éditeur encore plus confortable, des fonctions d'édition plus puissantes, mais aussi gonflé sensiblement son moteur pour des performances plus marquées en matière de programmation : près de 40 instructions dédiées aux manipulations des matrices, de nouvelles fonctions statistiques mathématiques.

Ces outils peuvent jouer un rôle fondamental dans de nombreux domaines, de la gestion aux maths/physique, du graphisme au traitement du signal.

Le programmeur en verra immédiatement les avantages : matrices pour le calcul vectoriel, matrices représentant les coordonnées d'un objet 3D.

Dans le domaine du calcul, les étudiants, mathématiciens, physiciens, enseignants et bien d'autres, découvriront que le GFA Basic peut résoudre très rapidement des problèmes ardus, en simplifiant considérablement les tâches de programmation.

Dernière parenthèse, le GFA Compilateur 3.5 que vous possédez dorénavant grâce à cette mise à jour est bien entendu destiné à prendre en charge les nouvelles instructions du GFA Basic, qu'il optimisera aux mêmes titres que les 400 autres commandes du langage.

Et maintenant, hâtons-nous d'en découvrir les trésors !

Les auteurs.

Le vieux Manuel

Chapitre 1

Les nouvelles possibilités de l'éditeur

L'éditeur utilise 2 octets de plus pour chaque ligne de programme, ce qui rend possible le pliage des fonctions et augmente la vélocité du scrolling arrière.

La recherche d'une séquence de caractères se fait maintenant même dans les procédures et fonctions repliées.

Lors du listage, les labels sont décalés de 2 cm vers la gauche.

TAB : Le curseur saute à la prochaine tabulation vers la droite.

Ctrl + TAB : Le curseur revient à la tabulation précédente vers la gauche.

Shift-gauche + TAB : Insère des espaces jusqu'à la prochaine tabulation.

Shift-droit + TAB : Supprime les espaces entre le curseur et la précédente tabulation.

Le vieux Manuel

Chapitre 2

Opérations linéaires sur des vecteurs et des matrices

Le Vieux Manuel

Toutes les fonctions définies dans ce chapitre font références à des tableaux de valeurs réelles ayant 1 ou 2 dimensions.

2. 1. Initialisation

MAT BASE 0
MAT BASE 1

L'instruction MAT BASE n'a de sens que si OPTION BASE 0 figure dans le programme. On peut alors définir le début de la numérotation des lignes et colonnes de matrices ou vecteurs à 1 pour MAT BASE 1 et à 0 pour MAT BASE 0.

MAT BASE n influe sur les commandes suivantes

- MAT READ
- MAT PRINT
- MAT CPY
- MAT XCPY
- MAT ADD
- MAT SUB
- MAT MUL

La valeur prise par défaut est MAT BASE 1

Exemple

```
OPTION BASE 0
MAT BASE 1
DATA 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16
DIM a(3.3)
MAT READ a()
PRINT a(1.1)

   Renvoie la valeur 1
```

2.2. Opérations unaires (sur une matrice)

MAT CLR a()
MAT SET a()=x
MAT ONE a()

a : nom du tableau de valeurs numériques
x : expression réelle

MAT CLR a() correspond à ARRAYFILL a(), 0, ce qui initialise à la valeur 0 tous les éléments de a (matrice ou vecteur).

➥ Abréviation : m cl a()

MAT SET a()=x correspond à ARRAYFILL a(), x, ce qui initialise à la valeur x tous les éléments de a (matrice ou vecteur).

➥ Abréviation : m se a()=x

MAT ONE a() crée, à partir d'une matrice carrée, une matrice unité dont les éléments a(1,1), a(2,2), ..., a(n,n) sont égaux à 1, les autres étant nuls.

➥ Abréviation : m o a()

Exemple

```
DATA 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16
DIM a(3.3)
MAT READ a()
PRINT a(1.1)
MAT CLR a()
PRINT a(1.1)
```

Le vieux Manuel

```
   Renvoie la valeur 1. puis la valeur 0

DIM a(5,7)
FOR i%=1 TO 5
  FOR j%=1 TO 7
    a(i%,j%)=RAND(10)
  NEXT j%
NEXT i%
MAT SET a().5.3
FOR i%=1 TO 5
  FOR j%=1 TO 7
    PRINT a(i%,j%)
  NEXT j%
NEXT i%

Renvoie 35 fois la valeur 5.3

DIM a(3,3)
MAT ONE a()
MAT PRINT a()

Renvoie      1.0.0

             0.1.0
             0.0.1
```

2.3. Opérations d'entrée-sortie

MAT READ a()
MAT PRINT [#i,]a()[,g,n]
MAT INPUT #i,a()

i, g, n : expression entière
a : nom du tableau de valeurs numériques

MAT READ a() lit une matrice ou un vecteur dans une zone de DATA.

➥ Abréviation : m r a()

Exemple

```
DATA 1,2,3,4,5,6,7,8,9,10
DIM a(2,5)
MAT READ a()
PRINT a(2,4)

      Renvoie la valeur 9
```

Le vieux Manuel

MAT PRINT [#i,]a()[,g,n]

affiche une matrice (sous forme de tableau) ou un vecteur (sur une ligne) formatés comme avec STR$(x,g,n). De plus, la sortie peut être redirigée vers un fichier ou périphérique avec #i.

➥ Abréviation : m p [#i,]a()[,g,n] ou m ? [#i,]a()[,g,n]

Exemple

```
DATA 1.2.33333.3
DATA 7.5.25873.9.376
DATA 3.23.7.2.8.999
DIM a(3.3)
MAT READ a()
MAT PRINT a()
PRINT"--------------"
MAT PRINT a().7.3

   Renvoie    1.2.33333.3
              7.5.25873.9.376
              3.23.7.2.8.999
              --------------
              1.000.2.333.3.000
              7.000.5.259.9.376
              3.230.7.200.8.999
```

MAT INPUT #1,a()

lit une matrice ou un vecteur dans un fichier ASCII. Les séparateurs et retours à la lignes peuvent être redéfinis, comme pour INPUT #.

➥ Abréviation : m i #i,a()

Exemple

```
OPEN "o".#1."Test.DAT"
DIM a(3.3)
MAT ONE a()
MAT PRINT #1.a()
CLOSE #1
MAT CLR a()
OPEN "i".#1."Test.DAT"
MAT INPUT #1.a()
CLOSE #1
MAT PRINT a()
   Renvoie    1.0.0
              0.1.0
              0.0.1
```

Le vieux Manuel

2.4. Opérations binaires (sur deux matrices)

MAT CPY a([I,J])=b([k,l])[,h,w]
MAT XCPY a([I,J])=b([k,l])[,h,w]
MAT TRANS a()[=b()]

a, b : nom des tableaux de valeurs numériques
i, j, k, l, h, w : expression entière

MAT CPY a([I,J])=b([k,l])[,h,w]

copie dans la matrice a, à partir de la ligne i et de la colonne j, h lignes de la matrice b à partir de la ligne k et w colonnes à partir de la colonne l.

➥ Abréviation : m c a(i,j)=b(k,l),h,w
m x a(i,j)=b(k,l),h,w
m t a()=b()

Exemple

```
DIM a(5.5).b(4.4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%.j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a(2.2)=b(2.2).3.3
MAT PRINT a()

   Renvoie     1.1.1.1.1
               1.1.1.1.1
               1.1.1.1.1
               1.1.1.1.1
               1.1.1.1.1
               -------------
               2.2.2.2
               3.3.3.3
               4.4.4.4
               5.5.5.5
               -------------
               1.1.1.1.1
               1.3.3.3.1
```

Le vieux Manuel

```
1.4.4.4.1
1.5.5.5.1
1.1.1.1.1
```

Cas particuliers

MAT CPY a()=b() copie dans la matrice a, la matrice b, complètement si elles sont de même ordre, les éléments d'indices équivalents sinon.

➥ Abréviation : m c a()=b()

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "------------"
MAT PRINT b()
PRINT "------------"
MAT CPY a()=b().3.3
MAT PRINT a()

   Renvoie     1.1.1
               1.1.1
               1.1.1
               1.1.1
               1.1.1
               ------------
               2.2.2.2
               3.3.3.3
               4.4.4.4
               5.5.5.5
               ------------
               2.2.2
               3.3.3
               4.4.4
               1.1.1
               1.1.1
```

MAT CPY a(i,j)=b() copie dans la matrice a à partir de la ligne i et de la colonne j, la matrice b, complète si possible, les éléments d'indices équivalents sinon. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a(i,j)=b()

Le vieux Manuel

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "------------"
MAT PRINT b()
PRINT "------------"
MAT CPY a(2,2)=b(2,2),3,3
MAT PRINT a()
   Renvoie    1.1.1
              1.1.1
              1.1.1
              1.1.1
              1.1.1
              ------------
              2.2.2.2
              3.3.3.3
              4.4.4.4
              5.5.5.5
              ------------
              1.1.1
              1.3.3
              1.4.4
              1.5.5
              1.1.1
```

MAT CPY a()=b(i,j) copie dans la matrice a, la matrice b à partir de la ligne i et de la colonne j, complète si possible, les éléments d'indices équivalents sinon. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a()=b(i,j)

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "------------"
MAT PRINT b()
PRINT "------------"
```

Le vieux Manuel

```
MAT CPY a()=b(2,2),3,3
MAT PRINT a()

   Renvoie     1.1.1
               1.1.1
               1.1.1
               1.1.1
               1.1.1
               -------------
               2.2.2.2
              3.3.3.3
               4.4.4.4
               5.5.5.5
               -------------
               3.3.3
               4.4.4
               5.5.5
               1.1.1
               1.1.1
```

MAT CPY a(I,J)=b(k,l) copie dans la matrice a à partir de la ligne i et de la colonne j, la matrice b à partir de la ligne k et de la colonne l, complète si possible, les éléments d'indices équivalents sinon.

➥ Abréviation : m c a(i,j)=b(k,l)

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a(2,2)=b(2,2)
MAT PRINT a()

   Renvoie     1.1.1
               1.1.1
               1.1.1
               1.1.1
               1.1.1
               -------------
               2.3.4.5
               2.3.4.5
               2.3.4.5
               2.3.4.5
```

Le Vieux Manuel

```
            ------------
            1.1.1
            1.3.4
            1.3.4
            1.3.4
            1.1.1
```

MAT CPY a()=b(),h,w

copie dans la matrice a, h lignes et w colonnes de la matrice b, les éléments d'indices équivalents s'il y a dépassement. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a()=b(),h,w

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "------------"
MAT PRINT b()
PRINT "------------"
MAT CPY a()=b()
MAT PRINT a()
```

```
Renvoie     1.1.1
            1.1.1
            1.1.1
            1.1.1
            1.1.1
            ------------
            2.3.4.5
            2.3.4.5
            2.3.4.5
            2.3.4.5
            ------------
            2.3.4
            2.3.4
            2.3.4
            2.3.4
            1.1.1
```

Le Vieux Manuel

MAT XCPY a([i,j])=b([k,l]),[h,w]

fonctionne selon le même principe que MAT CPY a([i,j])=b([k,l]),[h,w], à cela près qu'une transposée de la matrice b est faite pour l'opération, c'est à dire que les lignes et colonnes de b deviennent respectivement les colonnes et les lignes de la matrice servant au calcul. La matrice b reste inchangée aprés l'exécution de la fonction.

➥ Abréviation : m x a([i,j])=b([k,l]),[h,w]

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "------------"
MAT PRINT b()
PRINT "------------"
MAT XCPY a(2,2)=b(2,2),3,3
MAT PRINT a()
```

```
 Renvoie    1.1.1
            1.1.1
            1.1.1
            1.1.1
            1.1.1
            ------------
            2.3.4.5
            2.3.4.5
            2.3.4.5
            2.3.4.5
            ------------
            1.1.1
            1.3.3
            1.4.4
            1.5.5
            1.1.1
```

Le vieux Manuel

Cas particuliers

Semblables à MAT CPY a(I,J)=b(k,l),w,h

Pour MAT CPY comme pour MAT XCPY, i, j et h sont ignorés dans le cas d'utilisation de vecteurs. Après un DIM a(n),b(m), a et b seront définis comme

vecteurs ligne, c'est à dire des matrices de types (1,n) et (1,m). Pour travailler avec des vecteurs colonne, il vous faut des matrices de type (n,1) et (m,1), donc faire un DIM a(n,1),b(m,1). Si les deux vecteurs sont de même ordre (vecteurs ligne ou colonne), vous devez utiliser MAT CPY. Indépendamment du type des vecteurs a et b, MAT CPY considère les deux comme des vecteurs colonne. La syntaxe correcte est MAT CPY a(n,1)=b(m,1).

Exemple

```
DIM a(10),b(5)        ' a et b sont des vecteurs ligne
MAT SET a()=1
FOR i%=1 TO 5
  b(i%)=SUCC(i%)
NEXT i%
PRINT "a():    ";
MAT PRINT a()
PRINT "b():    ";
MAT PRINT b()
PRINT STRING$(45,"-")
MAT CPY a(3,1)=b(1,1)  ! Considère a et b comme vecteurs
                       ! colonne
PRINT "MAT CPY a(3,1)=b(1,1):  ";
MAT PRINT a()

   Renvoie   a():  1.1.1.1.1.1.1.1.1.1
             b():  2.3.4.5.6
             ---------------------------------------
             MAT CPY a(3,1)=b(1,1):  1.1.2.3.4.5.6.1.1.1
```

L'utilisation de MAT XCPY avec des vecteurs nécessite un vecteur ligne et un vecteur colonne. Le deuxième argument de la fonction est transposé (pour le calcul seulement, il n'est pas modifié) avant d'être copié dans le premier. MAT XCPY avec des vecteurs fonctionnera avec les dimensionnements suivants :

- DIM a(1,n),b(m,1) : a()=Vecteur ligne, b()=Vecteur colonne
- DIM a(n,1),b(1,m) : a()=Vecteur colonne, b()=Vecteur ligne

Exemple

```
DIM a(1,10),b(5,1)
MAT SET a()=1
FOR i%=1 TO 5
  b(i%,1)=SUCC(i%)
NEXT i%
MAT PRINT a()
PRINT
MAT PRINT b()
MAT XCPY a(1,2),b(1,1)
PRINT
MAT PRINT a()
```

Le vieux Manuel

```
Renvoie    1.1.1.1.1.1.1.1.1.1

           2
           3
           4
           5
           6

       1.1.2.3.4.5.6.1.1.1
```

On peut utiliser des paramètres optionnels h et w lors d'une copie de vecteurs avec MAT CPY et MAT XCPY mais :

➢ **Pour MAT CPY**

Si w >= 1, h seul est pris en compte
Si w = 0, pas de copie

➢ **Pour MAT XCOPY**

Si w >= 1, seul h est pris en compte, si b est un vecteur colonne qui est copié après la transposition dans un vecteur ligne
Si w = 0 , pas de copie
Si h >= 1, seul w est pris en compte, si b est un vecteur ligne qui est copié après la transposition dans un vecteur colonne
Si h = 0, pas de copie

MAT TRANS a()=b() copie dans a, la transposée de la matrice b avec le nombre de lignes de a égal au nombre de colonnes de b et le nombre de colonnes de a égal au nombre de lignes de b. La dimension des matrices est du type DIM a(n,m),b(m,n).

Exemple

```
DIM a(3.4).b(4.3)
MAT SET b()=4
MAT SET a()=1
MAT PRINT a()
PRINT STRING$(10."-")
MAT PRINT b()
PRINT STRING$(10."-")
MAT TRANS a()=b()
MAT PRINT a()

   Renvoie    1.1.1.1
              1.1.1.1
```

Le vieux Manuel

```
              1.1.1.1
              ----------
             4.4.4
              4.4.4
              4.4.4
              4.4.4
              ----------
              4.4.4.4
              4.4.4.4
              4.4.4.4
```

Si la matrice est carrée (même nombre de lignes et de colonnes), on peut utiliser MAT TRANS a(). Cette instruction intervertit les lignes et les colonnes de la matrice a. On peut récupérer le contenu initial en faisant un nouveau MAT TRANS a().

➥ Abréviation : m t a()

Exemple

```
DIM a(5,5)
FOR i%=1 TO 5
  FOR j%=1 TO 5
    a(i%,j%)=j%
  NEXT j%
NEXT i%
MAT PRINT a()
PRINT STRING$(10,"-")
MAT TRANS a()
MAT PRINT a()

   Renvoie    1.2.3.4.5
              1.2.3.4.5
              1.2.3.4.5
              1.2.3.4.5
              1.2.3.4.5
              ----------
              1.1.1.1.1
              2.2.2.2.2
              3.3.3.3.3
              4.4.4.4.4
              5.5.5.5.5
```

Le vieux Manuel

2.5. Instructions de calcul

MAT ADD a()=b()+c()
MAT ADD a(),b()
MAT ADD a(),x

```
MAT SUB a()=b()-c()
MAT SUB a(),b()
MAT SUB a(),x

MAT MUL a()=b()*c()
MAT MUL x=a()*b()
MAT MUL x=a()*b()*c()
MAT MUL a(),x

MAT NORM a(),0
MAT NORM a(),1

MAT DET x=a([i,j])[,n]
MAT QDET x=a([i,j])[,n]
MAT RANG x=a([i,j])[,n]
MAT INV a()=b()
```

a, b, c : nom des tableaux de valeurs numériques
x : expression numérique, scalaire
i, j, n : expression numérique

MAT ADD a()=b()+c() est seulement définie pour des matrices du même ordre, par exemple de dimensions DIM a(n,m),b(n,m),c(n,m) ou DIM a(n),b(n),c(n). Le résultat dans a est la somme des deux éléments de mêmes indices de b et c.

➥ Abréviation : m a()=b()+c()

Exemple

```
DIM a(3,5),b(3,5),c(3,5)
MAT SET b()=3
MAT SET c()=4
MAT PRINT b()
PRINT STRING$(10,"-")
MAT PRINT c()
PRINT STRING$(10,"-")
MAT ADD a()=b()+c()
MAT PRINT a()

   Renvoie   3.3.3.3.3
             3.3.3.3.3
             3.3.3.3.3
             ----------
             4.4.4.4.4
```

Le vieux Manuel

```
 4.4.4.4.4
 4.4.4.4.4
 ----------
 7.7.7.7.7
 7.7.7.7.7
 7.7.7.7.7
```

MAT ADD a(),b() équivaut à MAT ADD a()=a()+b().

➥ Abréviation : m a a(),b()

Exemple

```
DIM a(3.5).b(3.5)
MAT SET a()=1
MAT SET b()=3
MAT PRINT a()
PRINT STRING$(10,"-")
MAT PRINT b()
PRINT STRING$(10,"-")
MAT ADD a().b()
MAT PRINT a()

   Renvoie   1.1.1.1.1
             1.1.1.1.1
             1.1.1.1.1
             ----------
             3.3.3.3.3
             3.3.3.3.3
             3.3.3.3.3
             ----------
             4.4.4.4.4
             4.4.4.4.4
             4.4.4.4.4
```

MAT ADD a(),x est définie pour toutes matrices et vecteurs. Le scalaire x est ajouté élément par élément à ceux de a. Le résultat est disponible dans a.

➥ Abréviation : m a a(),x

Exemple

```
DIM a(3.5)
MAT SET a()=1
MAT PRINT a()
PRINT STRING$(10,"-")
MAT ADD a().5
MAT PRINT a()

   Renvoie   1.1.1.1.1
```


Vieux Manuel

```
          1.1.1.1.1
          1.1.1.1.1
          ----------
          6.6.6.6.6
          6.6.6.6.6
          6.6.6.6.6
```

MAT SUB a()=b()-c() est seulement définie pour des matrices du même ordre, par exemple de dimensions DIM a(n,m),b(n,m),c(n,m) ou DIM a(n),b(n),c(n). Le résultat dans a est la différence des deux éléments de mêmes indices de b et c.

➥ Abréviation : m a()=b()-c()

Exemple

```
DIM a(3.5),b(3.5),c(3.5)
MAT SET b()=5
MAT SET c()=3
MAT PRINT b()
PRINT STRING$(10,"-")
MAT PRINT c()
PRINT STRING$(10,"-")
MAT SUB a()=b()-c()
MAT PRINT a()

Renvoie   5.5.5.5.5
          5.5.5.5.5
          5.5.5.5.5
          ----------
          3.3.3.3.3
          3.3.3.3.3
          3.3.3.3.3
          ----------
          2.2.2.2.2
          2.2.2.2.2
          2.2.2.2.2
```

MAT SUB a(),b() équivaut à MAT SUB a(),b().

➥ Abréviation : m a a(),b()

Exemple

```
DIM a(3.5),b(3.5)
MAT SET a()=3
MAT SET b()=1
MAT PRINT a()
PRINT STRING$(10,"-")
MAT PRINT b()
```

Le vieux Manuel

```
PRINT STRING$(10,"-")
MAT SUB a(),b()
MAT PRINT a()

   Renvoie    3.3.3.3.3
              3.3.3.3.3
              3.3.3.3.3
              ----------
              1.1.1.1.1
              1.1.1.1.1
              1.1.1.1.1
              ----------
              2.2.2.2.2
              2.2.2.2.2
              2.2.2.2.2
```

MAT SUB a(),x est définie pour toutes matrices et vecteurs. Le scalaire x est soustrait élément par élément à ceux de a. Le résultat est disponible dans a.

➥ Abréviation : m s a(),x

Exemple

```
DIM a(3,5)
MAT SET a()=6
MAT PRINT a()
PRINT STRING$(10,"-")
MAT SUB a(),5
MAT PRINT a()

   Renvoie    6.6.6.6.6
              6.6.6.6.6
              6.6.6.6.6
              ----------
              5.5.5.5.5
              5.5.5.5.5
              5.5.5.5.5
              ----------
              1.1.1.1.1
              1.1.1.1.1
              1.1.1.1.1
```

MAT MUL a()=b()*c() met dans a le produit des matrices b et c. Pour que le résultat soit défini, il faut que le nombre de lignes de b soit égal au nombre de colonnes de c et que le nombre de colonnes de b soit égal au nombre de lignes de c. La matrice résultat a doit avoir un nombre de lignes équivalent à celui de b et le même nombre de colonnes que c, par exemple comme le dimensionnement DIM a(2,2),b(2,3),c(3,2). On obtient le

Le vieux Manuel

calcul d'un élément de position i,j par a(i,j)=SOMME(n=0;n=x)[b(i,x)*c(x,j)].

➥ Abréviation : m a()=b()*c()

Exemple

```
DIM a(2,2),b(2,3),c(3,2)
MAT SET b()=1
DATA 1.2,-3.4,5,-1
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL a()=b()*c()
MAT PRINT a(),5,1

   Renvoie   1.0   1.0   1.0
             1.0   1.0   1.0
             ------------------
              1.0   2.0
             -3.0   4.0
              5.0  -1.0
             ------------------
              3.0   5.0
              3.0   5.0
```

Si, au lieu de matrices, on a des vecteurs, alors on obtient un produit matriciel de vecteurs.

Exemple

```
DIM a(3,3),b(3),c(3)
DATA 1,2,-3,4,5,-1
MAT READ b()
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL a()=b()*c()
MAT PRINT a(),5,1

   Renvoie    1.0   2.0  -3.0
             ------------------
              4.0   5.0  -1.0
             ------------------
              4.0   5.0  -1.0
              8.0  10.0  -2.0
            -12.0 -15.0   3.0
```

Vieux Manuel

MAT MUL x=a()*b() est défini avec b et c vecteurs. Le résultat du produit scalaire de a et de b est un scalaire x. On le trouve en faisant le calcul x=SOMME((n=0;n=i)[a(n)*b(n)].

➥ Abréviation : m x=a()*b()

Exemple

```
DIM b(3),c(3)
DATA 1,2,-3,4,5,-1
MAT READ b()
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL x=b()*c()
PRINT x

   Renvoie     1.0.   2.0.  -3.0
          ------------------
               4.0.   5.0.  -1.0
             ------------------
             17.0
```

MAT MUL x=a()*b()*c()

est défini pour des vecteurs a, c et une matrice b particuliers. Le résultat est un scalaire x qui se calcule en multipliant le vecteur a à la matrice b puis ce produit au vecteur c. Le vecteur a doit avoir le même nombre d'éléments que la matrice b de lignes et le vecteur b le même nombre d'éléments que la matrice b de colonnes, comme le dimensionnement DIM a(5),b(5,3),c(3).

➥ Abréviation : m x=a()*b()*c()

Exemple

```
DIM a(2),b(2,3),c(3)
DATA 1,2,-3,4,5
MAT READ a()
MAT READ c()
MAT SET b()=1
MAT PRINT a(),5,1
PRINT STRING$(18,"-")
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
```

Le vieux Manuel

```
PRINT STRING$(18,"-")
MAT MUL x=a()*b()*c()
PRINT x

   Renvoie    1.0.  2.0
              ------------------
                 1.0.  1.0.  1.0
                 1.0.  1.0.  1.0
                 1.0.  1.0.  1.0
              ------------------
                 -3.0.  4.0.  5.0
              ------------------
              18.0
```

MAT NORM a(),0 ou MAT NORM a(),1

sont définies pour des matrices ou des vecteurs. MAT NORM a(),0 normalise une matrice ou un vecteur par ligne, MAT NORM a(),1 le fait par colonne. Une matrice normalisée par ligne (par colonne) doit avoir la somme des carrés des éléments d'une ligne (d'une colonne) égale à 1.

➥ Abréviation : m no a(),0 ou m no a(),1.

Exemple

```
DIM a(10,10),b(10,10),v(10)
DATA 1,2,3,4,5,6,7,8,9,-1
DATA 3,2,4,-5,2,4,5,1,6,2,7,2,8,1,6,-5
DATA -2,-5,-6,-1,2,-1,5,-6,7,4,5,8,1,3,4,10
DATA 5,-2,3,4,5,6,12,2,18,2,14,1,16,-21,-13
DATA 4,1,5,2,16,7,18,4,19,1,20,2,13,6,14,8,19,4,18,6
DATA 15,2,-1,8,13,6,-4,9,5,4,19,8,16,4,-20,9,21,4,13,8
DATA -3,6,6,-8,2,-9,1,4,-2,5,2,3,4,6,7,8,4
DATA 4,7,8,3,9,4,10,5,11,19,15,4,18,9,-20,12,6
DATA 5,3,-4,7,6,1,6,5,6,9,-9,2,-10,8,4,3,5,6,9,1
DATA 21,4,19,5,28,4,19,3,24,6,14,9,71,3,23,5,14,5,-12,3
'
CLS
MAT READ a()
MAT CPY b()=a()           ! Sauvegarder matrice originale
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
INP(2)
'
' Normalisation par ligne
'
CLS
MAT NORM a(),0
PRINT
PRINT "Normaliser par ligne : "
PRINT
```

Le Vieux Manuel

```
MAT PRINT a().7.2
INP(2)
'
' Tester la normalisation
'
PRINT
PRINT "Vérification : "
PRINT
FOR i%=1 TO 10
  MAT XCPY v()=a(i%,1)  ! Copier les lignes dans des vecteurs
  MAT MUL x=v()*v()     ! Calcule le produit scalaire entre v
                        ! et v
  PRINT x'
NEXT i%
PRINT
INP(2)
'
' Normaliser par colonne
'
CLS
MAT CPY a()=b()              ! Restituer matrice originale
MAT NORM a().1
PRINT "Normalise par colonne : "
PRINT
MAT PRINT a().7.2
INP(2)
'
' Tester la normalisation
'
PRINT
PRINT "Vérification : "
PRINT
FOR i%=1 TO 10
  MAT CPY v()=a(1,i%)   ! Copier les colonnes dans des vecteurs
  MAT MUL x=v()*v()     ! Calcule le produit scalaire entre v
                        ! et v
  PRINT x'
NEXT i%
INP(2)
```

Renvoie

Matrice originale

```
  1.00.   2.00.   3.00.   4.00.   5.00.   6.00.   7.00.   8.00.   9.00.
-1.00
  3.20.   4.00.  -5.00.   2.40.   5.10.   6.20.   7.20.   8.10.   6.00.
-5.00
 -2.00.  -5.00.  -6.00.  -1.20.  -1.50.  -6.70.   4.50.   8.10.   3.40.
10.00
  5.00.  -2.30.   4.00.   5.60.  12.20.  18.20.  14.10.  16.00. -21.00.
-13.00
  4.10.   5.20.  16.70.  18.40.  19.10.  20.20.  13.60.  14.80.  19.40.
18.60
 15.20.  -1.80.  13.60.  -4.90.   5.40.  19.80.  16.40. -20.90.  21.40.
 13.80
```

Le Vieux Manuel

```
  -3.60.    6.00.   -8.20.   -9.10.    4.00.   -2.50.    2.00.    3.40.    6.70.
   8.40
   4.70.    8.30.    9.40.   10.50.   11.00.   19.00.   15.40.   18.90.  -20.00.
  12.60
   5.30.   -4.70.    6.10.    6.50.    6.90.   -9.20.  -10.80.    4.30.    5.60.
   9.10
  21.40.   19.50.   28.40.   19.30.   24.60.   14.90.   71.30.   23.50.   14.50.
 -12.30

Normalisation par ligne

   0.06.    0.12.    0.18.    0.24.    0.30.    0.35.    0.41.    0.47.    0.53.
  -0.06
   0.18.    0.23.   -0.29.    0.14.    0.29.    0.36.    0.42.    0.47.    0.35.
  -0.29
  -0.11.   -0.28.   -0.34.   -0.07.   -0.09.   -0.38.    0.26.    0.46.    0.19.
   0.57
   0.12.   -0.06.    0.10.    0.14.    0.30.    0.45.    0.35.    0.40.   -0.52.
  -0.32
   0.08.    0.10.    0.33.    0.36.    0.38.    0.40.    0.27.    0.29.    0.38.
   0.37
   0.32.   -0.04.    0.29.   -0.10.    0.11.    0.42.    0.35.   -0.44.    0.45.
   0.29
  -0.19.    0.32.   -0.44.   -0.48.    0.21.   -0.13.    0.11.    0.18.    0.36.
   0.45
   0.11.    0.19.    0.21.    0.24.    0.25.    0.43.    0.35.    0.43.   -0.46.
   0.29
   0.23.   -0.21.    0.27.    0.29.    0.31.   -0.41.   -0.48.    0.19.    0.25.
   0.40
   0.23.    0.21.    0.30.    0.21.    0.26.    0.16.    0.76.    0.25.    0.15.
  -0.13

Vérification :

1 1 1 1 1 1 1 1 1 1

Normalisation par colonne :

   0.04.    0.08.    0.05.    0.12.    0.13.    0.14.    0.09.    0.16.    0.20.
  -0.03
   0.11.    0.16.   -0.13.    0.07.    0.14.    0.14.    0.09.    0.18.    0.13.
  -0.14
  -0.07.   -0.21.   -0.15.   -0.04.   -0.04.   -0.15.    0.06.    0.18.    0.07.
   0.28
   0.18.   -0.09.    0.10.    0.17.    0.33.    0.41.    0.18.    0.35.   -0.46.
  -0.36
   0.14.    0.21.    0.42.    0.57.    0.51.    0.46.    0.17.    0.33.    0.42.
   0.52
   0.53.   -0.07.    0.35.   -0.15.    0.15.    0.45.    0.21.   -0.46.    0.47.
   0.38
  -0.13.    0.25.   -0.21.   -0.28.    0.11.   -0.06.    0.03.    0.08.    0.15.
   0.23
   0.17.    0.34.    0.24.    0.33.    0.30.    0.43.    0.20.    0.42.   -0.44.
   0.35
   0.19.    0.19.    0.15.    0.20.    0.19.   -0.21.   -0.14.    0.10.    0.12.
   0.25
   0.75.    0.80.    0.72.    0.81.    0.88.    0.34.    0.90.    0.94.    0.32.
  -0.34
```

Vieux Manuel

```
vérification :

1 1 1 1 1 1 1 1 1 1
```

MAT DET x=a([i,j])[,n]

calcule le déterminant d'une matrice carrée de type (n,n). Les indices commencent à 0 ou 1 en fonction de MAT BASE. Il est aussi possible de calculer le déterminant d'une partie carrée d'une matrice quelconque. Pour cela, indiquer pour i et j le début de la partie de matrice dont vous voulez calculer le déterminant et pour n le nombre de lignes et colonnes. Le calcul se fera de façon interne sans affecter la matrice originale.

➥ Abréviation : m d x=a([i,j])[,n]

Exemple

```
DIM a(10,10),b(4,4)
DATA 1,2,3,4,5,6,7,8,9,-1
DATA 3,2,4,-5,2,4,5,1,6,2,7,2,8,1,6,-5
DATA -2,-5,-6,-1,2,-1,5,-6,7,4,5,8,1,3,4,10
DATA 5,-2,3,4,5,6,12,2,18,2,14,1,16,-21,-13,3,8
DATA 4,1,5,2,16,7,18,4,19,1,20,2,13,6,14,8,19,4,18,6
DATA 15,2,-1,8,13,6,-4,9,5,4,19,8,16,4,-20,9,21,4,13,8
DATA -3,6,6,-8,2,-9,1,4,-2,5,2,3,4,6,7,8,4,10,9
DATA 4,7,8,3,9,4,10,5,11,19,15,4,18,9,-20,12,6
DATA 5,3,-4,7,6,1,6,5,6,9,-9,2,-10,8,4,3,5,6,9,1
DATA 21,4,19,5,28,4,19,3,24,6,14,9,71,3,23,5,14,5,-12,3
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
PRINT
PRINT "Déterminant : ";
MAT DET x=a()              ! Calculer le déterminant de a
PRINT x;
MAT DET y=a(1,4),4         ! Calculer le déterminant d'une partie ! de la
matrice
PRINT
PRINT "Déterminant de a(1,4),4 : ";
PRINT y
PRINT
PRINT "Vérification :"
PRINT
MAT CPY b()=a(1,4),4,4 ! Copie la partie de matrice dans b
MAT PRINT b(),7,2
MAT DET z=b()              ! Calculer le déterminant de b
PRINT
```

```
PRINT z

   Renvoie

Matrice originale

    1.00.    2.00.    3.00.    4.00.    5.00.    6.00.    7.00.    8.00.    9.00.
   -1.00
    3.20.    4.00.   -5.00.    2.40.    5.10.    6.20.    7.20.    8.10.    6.00.
   -5.00
   -2.00.   -5.00.   -6.00.   -1.20.   -1.50.   -6.70.    4.50.    8.10.    3.40.
   10.00
    5.00.   -2.30.    4.00.    5.60.   12.20.   18.20.   14.10.   16.00.  -21.00.
  -13.00
    3.80.    4.10.    5.20.   16.70.   18.40.   19.10.   20.20.   13.60.   14.80.
   19.40
   18.60.   15.20.   -1.80.   13.60.   -4.90.    5.40.   19.80.   16.40.  -20.90.
   21.40
   13.80.   -3.60.    6.00.   -8.20.   -9.10.    4.00.   -2.50.    2.00.    3.40.
    6.70
    8.40.   10.90.    4.70.    8.30.    9.40.   10.50.   11.00.   19.00.   15.40.
   18.90
  -20.00.   12.60.    5.30.   -4.70.    6.10.    6.50.    6.90.   -9.20.  -10.80.
    4.30
    5.60.    9.10.   21.40.   19.50.   28.40.   19.30.   24.60.   14.90.   71.30.
   23.50

Déterminant : -2549840202186
Déterminante de a(1,4).4 : -57.61200000001

Vérification :

    4.00.    5.00.    6.00.    7.00
    2.40.    5.10.    6.20.    7.20
   -1.20.   -1.50.   -6.70.    4.50
    5.60.   12.20.   18.20.   14.10

-57.61200000001
```

MAT QDET x=a([i,j])[,n]

fonctionne de la même manière que MAT DET mais est optimisée pour la vitesse. En général, les deux procédé donnent le même résultat mais pour des matrices dont le déterminant est proche de 0, il vaut mieux utiliser la fonction MAT DET.

➥ Abréviation : m qd x=a([i,j])[,n].

Exemple

```
DIM a(10,10)
DATA 1,2,3,4,5,6,7,8,9,-1
```

```
DATA 3.2.4.-5.2.4.5.1.6.2.7.2.8.1.6.-5
DATA -2.-5.-6.-1.2.-1.5.-6.7.4.5.8.1.3.4.10
DATA 5.-2.3.4.5.6.12.2.18.2.14.1.16.-21.-13.3.8
DATA 4.1.5.2.16.7.18.4.19.1.20.2.13.6.14.8.19.4.18.6
DATA 15.2.-1.8.13.6.-4.9.5.4.19.8.16.4.-20.9.21.4.13.8
DATA -3.6.6.-8.2.-9.1.4.-2.5.2.3.4.6.7.8.4.10.9
DATA 4.7.8.3.9.4.10.5.11.19.15.4.18.9.-20.12.6
DATA 5.3.-4.7.6.1.6.5.6.9.-9.2.-10.8.4.3.5.6.9.1
DATA 21.4.19.5.28.4.19.3.24.6.14.9.71.3.23.5.14.5.-12.3
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a().7.2
PRINT
PRINT "Déterminant avec MAT DET : ";
MAT DET x=a()
PRINT x:
PRINT
PRINT "Déterminant avec MAT QDET : ";
MAT DET y=a()
PRINT y:
PRINT
PRINT "Ecart : ";x-y

   Renvoie

Matrice originale

   1.00.   2.00.   3.00.   4.00.   5.00.   6.00.   7.00.   8.00.   9.00.
  -1.00
   3.20.   4.00.  -5.00.   2.40.   5.10.   6.20.   7.20.   8.10.   6.00.
  -5.00
  -2.00.  -5.00.  -6.00.  -1.20.  -1.50.  -6.70.   4.50.   8.10.   3.40.
  10.00
   5.00.  -2.30.   4.00.   5.60.  12.20.  18.20.  14.10.  16.00. -21.00.
 -13.00
   3.80.   4.10.   5.20.  16.70.  18.40.  19.10.  20.20.  13.60.  14.80.
  19.40
  18.60.  15.20.  -1.80.  13.60.  -4.90.   5.40.  19.80.  16.40. -20.90.
  21.40
  13.80.  -3.60.   6.00.  -8.20.  -9.10.   4.00.  -2.50.   2.00.   3.40.
   6.70
   8.40.  10.90.   4.70.   8.30.   9.40.  10.50.  11.00.  19.00.  15.40.
  18.90
 -20.00.  12.60.   5.30.  -4.70.   6.10.   6.50.   6.90.  -9.20. -10.80.
   4.30
   5.60.   9.10.  21.40.  19.50.  28.40.  19.30.  24.60.  14.90.  71.30.
  23.50

Déterminant avec MAT DET : -2549840202186
Déterminant avec MAT QDET : -2549840202186
Ecart : 0
```

Vieux Manuel

MAT RANG x=a([i,j])[,n]

donne le rang d'une matrice carrée. On peut, de la même manière que pour MAT QDET, calculer le rang d'une partie de matrice.

➥ Abréviation : m ra x=a([i,j])[,n].

Exemple

```
DIM a(5,5)
DATA 1.2.3.4.5
DATA 3.2.4.-5.2.4.5.1
DATA -2.4.-5.2.4.5.1
DATA 5.-2.3.4.5.6.12.2
DATA 4.1.5.2.16.7.18.4.19.1
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
PRINT
PRINT "Rang de a: ";
MAT RANG x=a()
PRINT x;
PRINT
PRINT "Rang de a(1,2),3 : ";
MAT RANG y=a(1,2),3
PRINT y;
PRINT

   Renvoie

Matrice originale

   1.00.   2.00.   3.00.   4.00.   5.00
   3.20.   4.00.  -5.00.   2.40.   5.10
  -2.00.   4.00.  -5.00.   2.40.   5.10
   5.00.  -2.30.   4.00.   5.60.  12.20
   4.10.   5.20.  16.70.  18.40.  19.10

Rang de a: 5
Rang de a(1,2),3 : 2
```

MAT INV b()=a()

calcule l'inverse d'une matrice carrée. L'inverse de a est placé dans b. La matrice b doit être du même type que a.

➥ Abréviation : m inv b()=a().

Vieux Manuel

Exemple

```
DIM a(5,5),b(5,5),c(5,5)
DATA 1,2,3,4,5
DATA 3,2,4,-5,2,4,5,1
DATA -2,4,-5,2,4,5,1
DATA 5,-2,3,4,5,6,12,2
DATA 4,1,5,2,16,7,18,4,19,1
'
CLS
MAT READ a()
PRINT "Matrice originale a() : "
PRINT
MAT PRINT a(),7,2
'
MAT INV b()=a()
PRINT
PRINT "Inverse de a() : "
PRINT
MAT PRINT b(),7,2
PRINT
PRINT "Verification b()*a() = matrice unité ? "
PRINT
MAT MUL c()=b()*a()
MAT PRINT c(),7,2
   Renvoie

Matrice originale a() :

   1.00.    2.00.    3.00.    4.00.    5.00
   3.20.    4.00.   -5.00.    2.40.    5.10
  -2.00.    4.00.   -5.00.    2.40.    5.10
   5.00.   -2.30.    4.00.    5.60.   12.20
   4.10.    5.20.   16.70.   16.40.   19.10

Inverse de a() :

   0.00.    0.19.   -0.19.   -0.00.   -0.00
   0.97.    0.02.   -0.09.   -0.10.   -0.17
   0.71.   -0.10.   -0.10.   -0.01.   -0.12
  -1.65.    0.17.    0.11.   -0.06.    0.39
   0.71.   -0.12.    0.04.    0.09.   -0.17

Verification b()*a() = matrice unité ?

   1.00.    0.00.    0.00.    0.00.    0.00
   0.00.    1.00.    0.00.    0.00.   -0.00
   0.00.   -0.00.    1.00.    0.00.   -0.00
  -0.00.   -0.00.   -0.00.    1.00.    0.00
  -0.00.    0.00.    0.00.    0.00.    1.00
```

Vieux Manuel

Chapitre 3

Autres fonctions de la version 3.5

Voici, présentées ici, les autres instructions ne traitant pas du calcul matriciel et vectoriel.

3.1. Combinatoire

x=FACT(n)
y=VARIAT(n,k)
z=COMBIN(n,k)

x, y, z: variables
n, k: expressions entières

x=FACT(n) place dans x la factorielle d'un nombre entier n. C'est le produit des n premiers nombres entiers naturels, hormis 0 (0!=1).

y=VARIAT(n,k) met dans y le nombre possible de variations de n éléments de la kième classe sans répétition.

➥ VARIAT(n,k)=FACT(n)/FACT(n-k)

Le Vieux Manuel

z=COMBIN(n,k) calcule dans z le coefficient binomial de (n,k). Par exemple calculer le nombre de possibilités différentes de construire un sous-ensemble de k éléments d'un ensemble de n éléments.

➥ COMBIN(n,k)=FACT(n)/(FACT(k)*FACT(n-k))

3.2. Pointeurs sur DATA

_DATA
_DATA=

_DATA renvoie un pointeur sur les zones de DATA. Si _DATA est à 0 et que l'on tente un READ, on reçoit le message d'erreur "out of data".

_DATA= fixe le pointeur de la zone de DATA sur une valeur déterminée précedemment avec _DATA.

```
DIM dp%(100)
DATA 1,2,3,4,5,6,7,8,9
DATA 13,24,328,3242,1,0
'
i%=0
DO WHILE _DATA
  dp%(i%)=_DAT
  INC i%
  READ a
LOOP
'
DEC i%
'
FOR j%=i% DOWNTO 0
  _DATA=dp%(j%)
  READ a
  PRINT a
NEXT j%
INP(2)
```

Renvoie : 0 1 3242 328 24 13 9 8 7 6 5 4 3 2 1

Achevé d'imprimer sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94) (1) 46.86.73.54
Dépôt légal - Janvier 1990 - N° d'impression: 5219

ATARI ST+STE

GFA BASIC 3.5

Possesseurs du GFA Basic 3.01 à 3.07 et GFA Compilateur 3.01 à 3.03, cette mise à jour vous permet d'acquérir la version 3.5 des nouveaux interpréteurs et compilateurs. C'est pourquoi seules les nouvelles caractéristiques de ces produits sont étudiées dans ce manuel.

L'INTERPRETEUR GFA BASIC 3.5

C'est toute la puissance du GFA Basic 3.0 plus de nouveaux outils et fonctions pour faciliter la tâche des programmeurs :

• Un éditeur encore plus souple et un plus grand confort d'utilisation : scrolling rapide, repliage des fonctions, recherche étendue aux procédures et fonctions repliées.

• Plus de 40 nouvelles instructions pour simplifier votre travail : **fonctions statistiques** : factorielle, calcul des combinaisons et permutations; **fonctions matricielles** : initialisation, déterminant, inverse, opérations arithmétiques, transfert et copie de matrices... Redéfintion des pointeurs sur les zones de données (DATA).

Avec la version 3.5 de nouveaux domaines d'application vous sont aujourd'hui immédiatement accessibles : gestion, maths, physique, graphisme, simulation, traitement du signal...

LE COMPILATEUR GFA 3.5

Le compilateur optimise au même titre que les 400 autres commandes du langage les nouvelles instructions du GFA Basic 3.5.

3 325121 030390

REF. : ST 039. PRIX 290 F.

EDITIONS MICRO APPLICATION
58 RUE DU FAUBOURG POISSONNIERE 75010 PARIS
TEL (1) 47 70 32 44