GAMS:Data Exchange, Operators

Zeliha ERGÜL AYDIN
zergul@eskisehir.edu.tr

Department of Industrial Engineering
Eskisehir Technical University

December 10, 2020

Data Exchange with Application

- Text file
- Microsoft Excel
- Database

Read Data with Text Files

The $include compile time command is used to instruct the GAMS compiler to include the context of a different file at the current position of the input stream. (usually in ASCII format: Be careful about Turkish Characters!).

table a(i,j) amount of nut. j in food i

$include food_info.txt;

Read Data with Text Files .csv

The $ondelim command only enables the use of commas as a separation symbol!

```
table a(i,j) amount of nut. j in food i

$ondelim

$include food_info.csv

$offdelim ;
```

Write Results to Text Files

The $ondelim command only enables the use of commas as a separation symbol! File results / results.txt /;

```
put results;

put "Model status", diet.modelstat /;

put "Cost", z.l /;

put "Decision variables" /;

loop((i),

put x.l(i)," number of foood " i.tl, " eaten daily" /);

putclose;
```

Data Exchange with GDX

A GDX file is a file that stores the values of one or more GAMS symbols such as sets, parameters variables and equations. GDX files can be used to prepare data for a GAMS model, present results of a GAMS model, store results of the same model using different parameters etc.

Scalar

Single (scalar) data entry.

Scalar c /10/;

or

Scalar c;

c=10

card operator

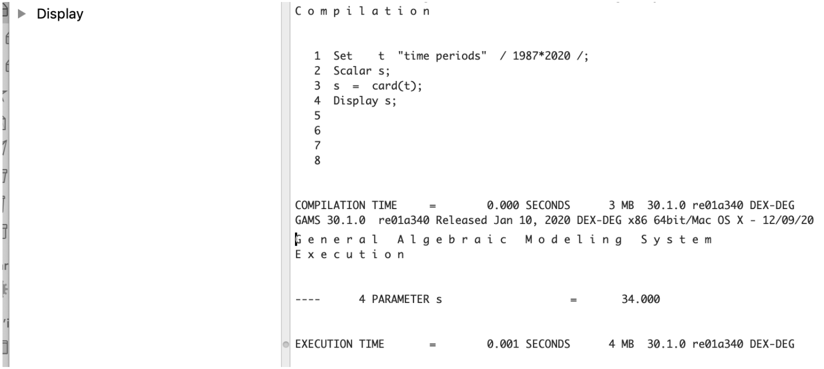
The operator card returns the number of elements in a set.

```
Set    t  "time periods"  / 1987*2020 /;
Scalar s;
s  =  card(t);
Display s;


```

card operator

The operator card returns the number of elements in a set.

```
Set     t  "time periods"  / 1987*2020 /;
Scalar s;
s  =  card(t);
Display s;
```

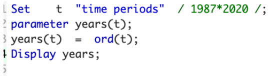


Display

```
Compilation


   1  Set     t  "time periods"  / 1987*2020 /;
   2  Scalar s;
   3  s  =  card(t);
   4  Display s;
   5
   6
   7
   8


COMPILATION TIME     =        0.000 SECONDS      3 MB  30.1.0 re01a340 DEX-DEG
GAMS 30.1.0  re01a340 Released Jan 10, 2020 DEX-DEG x86 64bit/Mac OS X - 12/09/20
General Algebraic Modeling System
Execution


----      4 PARAMETER s                    =       34.000


EXECUTION TIME       =        0.001 SECONDS      4 MB  30.1.0 re01a340 DEX-DEG
```

ord operator

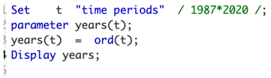
The operator ord returns the relative position of a member in a set.



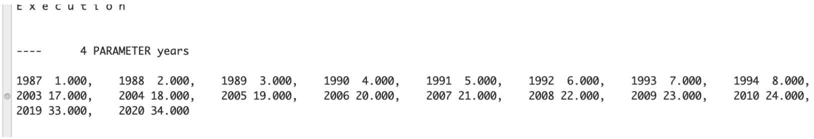
```
Set    t  "time periods"  / 1987*2020 /;
parameter years(t);
years(t)  =  ord(t);
Display years;
```

ord operator

The operator ord returns the relative position of a member in a set.

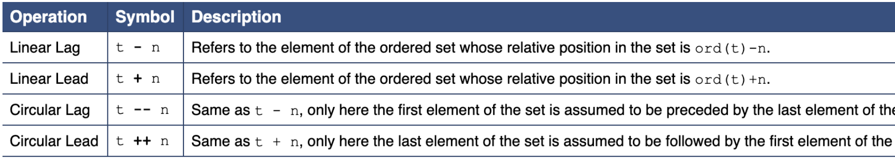


```
Set     t  "time periods"  / 1987*2020 /;
parameter years(t);
years(t)  =  ord(t);
Display years;
```



```
E x e c u t i o n


----      4 PARAMETER years

1987  1.000,    1988  2.000,    1989  3.000,    1990  4.000,    1991  5.000,    1992  6.000,    1993  7.000,    1994  8.000,
2003 17.000,    2004 18.000,    2005 19.000,    2006 20.000,    2007 21.000,    2008 22.000,    2009 23.000,    2010 24.000,
2019 33.000,    2020 34.000
```

Lag and Lead Operators



Operation	Symbol	Description
Linear Lag	`t - n`	Refers to the element of the ordered set whose relative position in the set is `ord(t)-n`.
Linear Lead	`t + n`	Refers to the element of the ordered set whose relative position in the set is `ord(t)+n`.
Circular Lag	`t -- n`	Same as `t - n`, only here the first element of the set is assumed to be preceded by the last element of th
Circular Lead	`t ++ n`	Same as `t + n`, only here the last element of the set is assumed to be followed by the first element of the

Lag and Lead Operators

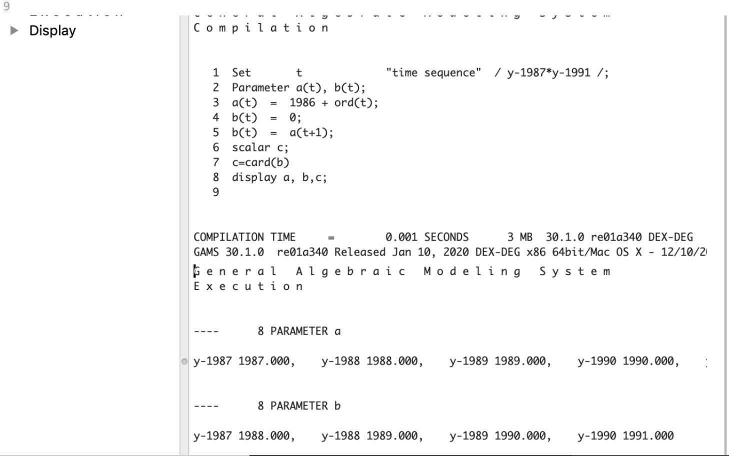
The operator ord returns the relative position of a member in a set.

```
Set       t             "time sequence"  / y-1987*y-1991 /;
Parameter a(t), b(t);
a(t)  =  1986 + ord(t);
b(t)  =  0;
b(t)  =  a(t+1);
scalar c;
c=card(b)
display a, b,c;

```

Lag and Lead Operators

The operator ord returns the relative position of a member in a set.

```
Set       t             "time sequence"  / y-1987*y-1991 /;
Parameter a(t), b(t);
a(t)  =  1986 + ord(t);
b(t)  =  0;
b(t)  =  a(t+1);
scalar c;
c=card(b)
display a, b,c;

```



▶ Display

```
C o m p i l a t i o n


   1  Set       t             "time sequence"  / y-1987*y-1991 /;
   2  Parameter a(t), b(t);
   3  a(t)  =  1986 + ord(t);
   4  b(t)  =  0;
   5  b(t)  =  a(t+1);
   6  scalar c;
   7  c=card(b)
   8  display a, b,c;
   9


COMPILATION TIME     =        0.001 SECONDS      3 MB  30.1.0 re01a340 DEX-DEG
GAMS 30.1.0  re01a340 Released Jan 10, 2020 DEX-DEG x86 64bit/Mac OS X - 12/10/2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----      8 PARAMETER a

y-1987 1987.000,    y-1988 1988.000,    y-1989 1989.000,    y-1990 1990.000,


----      8 PARAMETER b

y-1987 1988.000,    y-1988 1989.000,    y-1989 1990.000,    y-1990 1991.000
```

Conditional Expressions and Assignments

The operator $ may be read as under the condition that the following logical condition evaluates to TRUE (or is unequal 0).

$if\,(b > 1.5)$, then a $= 2$;

$a\$(b > 1.5) = 2$;

Conditional Expressions and Assignments

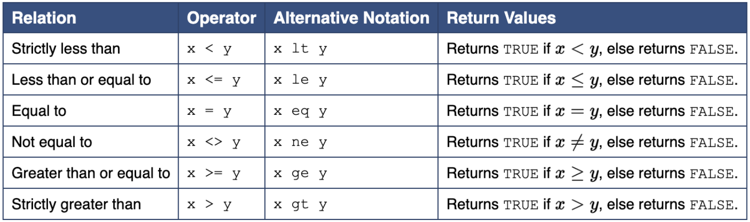
The operator $ may be read as under the condition that the following logical condition evaluates to TRUE (or is unequal 0).

$if\,(b > 1.5)$, then a $=$ 2 else a$=$0;

$a = 2\$(b > 1.5)$;

Conditional Expressions Numerical Relational Order

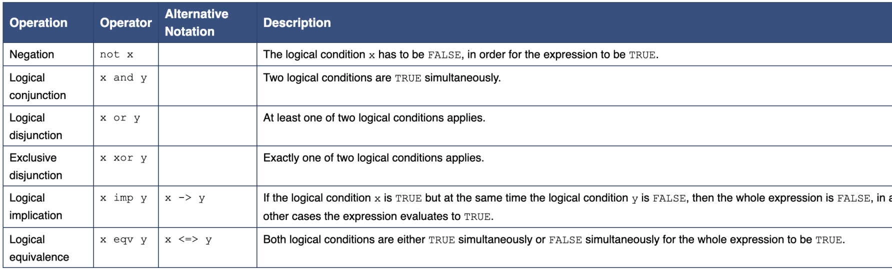
The operator $ may be read as under the condition that the following logical$_{c}onditionevaluatestoTRUE(orisunequal0)$.



Relation	Operator	Alternative Notation	Return Values
Strictly less than	`x < y`	`x lt y`	Returns `TRUE` if $x < y$, else returns `FALSE`.
Less than or equal to	`x <= y`	`x le y`	Returns `TRUE` if $x \leq y$, else returns `FALSE`.
Equal to	`x = y`	`x eq y`	Returns `TRUE` if $x = y$, else returns `FALSE`.
Not equal to	`x <> y`	`x ne y`	Returns `TRUE` if $x \neq y$, else returns `FALSE`.
Greater than or equal to	`x >= y`	`x ge y`	Returns `TRUE` if $x \geq y$, else returns `FALSE`.
Strictly greater than	`x > y`	`x gt y`	Returns `TRUE` if $x > y$, else returns `FALSE`.

Conditional Expressions Logical Conditions

The operator $ may be read as under the condition that the following logical condition evaluates to TRUE (or is unequal 0).



Operation	Operator	Alternative Notation	Description
Negation	`not x`		The logical condition `x` has to be `FALSE`, in order for the expression to be `TRUE`.
Logical conjunction	`x and y`		Two logical conditions are `TRUE` simultaneously.
Logical disjunction	`x or y`		At least one of two logical conditions applies.
Exclusive disjunction	`x xor y`		Exactly one of two logical conditions applies.
Logical implication	`x imp y`	`x -> y`	If the logical condition `x` is `TRUE` but at the same time the logical condition `y` is `FALSE`, then the whole expression is `FALSE`, in a other cases the expression evaluates to `TRUE`.
Logical equivalence	`x eqv y`	`x <=> y`	Both logical conditions are either `TRUE` simultaneously or `FALSE` simultaneously for the whole expression to be `TRUE`.

Conditional Equations

$$y_i \geq x_i, \quad \forall i, i > 1$$

Conditional Equations

$$y_i \geq x_i, \quad \forall i, i > 1$$

contraint(i)$(ord(i)>1).. y(i)=g=x(i);

Conditional Equations

$$a = \sum_{i=1}^{n-1} x_i$$

Conditional Equations

$$a = \sum_{i=1}^{n-1} x_i$$

```
contraint.. a=e=sum(i$(ord(i)<card(i)),x(i))
```

Assignment to Decision Variables

A fixed value for the variable. $x_0 = 20$

```
x.fx('0')=20;
```

Example: An Inventory Model

XYZ company must determine how many trucks should be produced during each of the next four months. The demands are the following: 20 trucks in April, 10 trucks in May, 30 trucks in June, 50 trucks in July. XYZ company must meet demands on time. At the beginning of April, the company has an inventory of 2 trucks. At the beginning of each month, XYZ must decide how many trucks should be produced during that month. For simplicity, we assume that trucks manufactured during a quarter can be used to meet the demand for that quarter. XYZ can produce up to 30 trucks monthly. The production cost of a truck is 540000 TL. At the end of each month (after production has occurred and the current month's demand has been satisfied), a holding cost of 12000 TL per truck is incurred. Use linear programming to determine a production schedule to minimize the sum of production and inventory costs during the next four months.

Example: An Inventory Model

Sets:

T={April,May,June,July} months

Parameters:

a=12000 holding cost

b=540000 production cost

c=30 upper limit for production

d_t: of trucks demand in month t

$$d = \begin{bmatrix} 20 \\ 10 \\ 30 \\ 50 \end{bmatrix}$$

Example: An Inventory Model

Decision variables x_t: of trucks produced in month t

s_t: of trucks in inventory of month t

Example: An inventory model

$$Minz = \sum_{t=1}^{4} as_t + bx_t$$

s.t

$$s_t = s_{t-1} + x_t - d_t, \quad \forall t, \quad t = \overline{1,4}$$

$$s_{t-1} + x_t \geq d_t, \quad \forall t, \quad t = \overline{1,4}$$

$$x_t \leq c, \quad \forall t, \quad t = \overline{1,4}$$

$$s_0 = 20$$

$$x_t, s_t \geq 0, \quad \forall t, t = \overline{1,4}$$

Example: An inventory model

Write GAMS code of this model by using following set:

Set t period /April,May, June, July/;

Example: An inventory model

Write GAMS code of this model by using following set:

Set t period /March,April,May, June, July/;