به نام خدا

آموزش نرم افزار Gmas

General Algebraic Modeling System

حسام پرند

Hesam_nm_1988@yahoo.com

Ie86.blogfa.com

فهرست مطالب :

مقدمه:

Gams قوی ترین و مجهز ترین برنامه ی حل مسائل تحقیق در عمیات است و در مسائل برنامه ریزی خطی، غیر خطی، عدد صحیح، مختلط و ... بکار می رود و انواع محدودیت ها را در بر می گیرد. البته لازم به ذکر است که علاوه بر تحقیق در عملیات در علوم زیر نیز بکار می رود:

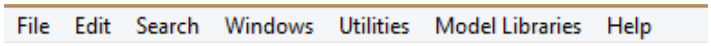
- اقتصاد
- مهندسی شیمی
- امور مالی و ...

معرفی محیط برنامه

- نوار منو
- نوار ابزار
- نوار وضعیت
- پنجره ویرایش
- پنجره اجرا
- پنجره خروجی

معرفی محیط برنامه : نوار منو

با توجه به شکل منوهای موجود در نوار منو را مشاهده می کنید

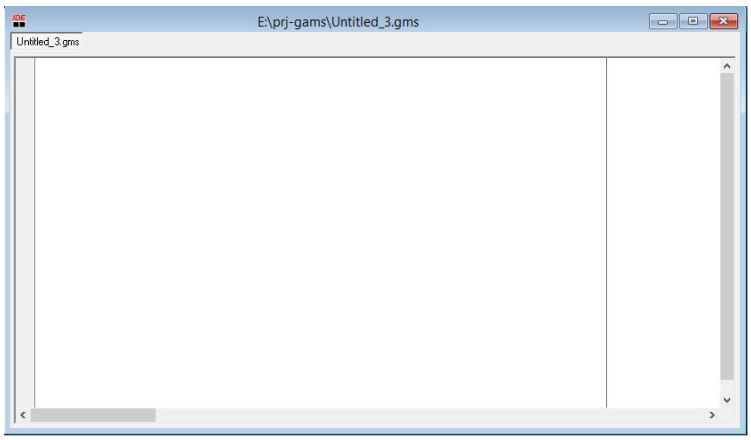


نوار ابزار نیز مطابق شکل زیر است

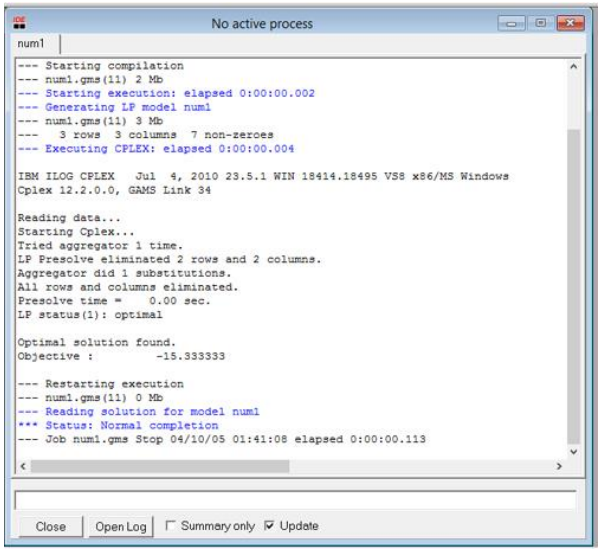
نوار وضعیت :

معرفی محیط برنامه :

پنجره ویرایش مکانی است که برنامه نویسی مدل در آن صورت می گیرد :



معرفی محیط برنامه



پنجره اجرا :

بعد از اجرای مدل در پنجره ویرایش از طریق فرمان Run در منوی فایل یا کلید میانبر F9 پنجره جدیدی به نام پنجره اجرا باز می شود که مراحل اجرای مدل و همچنین خطاهای احتمالی به همراه کد خطا در آن به نمایش میابد.

معرفی محیط برنامه

در نوار وضعیت چهار کادر وجود دارد :

-اولین کادر نماینگر موقعیت مکان نماست

-کادر بعدی ابتدا خالی است اما بعد اصلاح پرونده پیغام modified مشاهده می شود

-کادر سوم وقتی در حالت insert می باشد کاربر می تواند هر عبارت دلخواهی را در مدل بین دو عبارت دیگر درج کند. گاهی لازم است عبارتی جایگزین عبارت دیگری شود.در این صورت باید از وضعیت overwrite استفاده کرد (با فشردن کلید insert)

- کادر چهارم وظایف دکمه های نوار ابزار و برخی کلید های میانبر را نشان می دهد.

معرفی محیط برنامه

پنجره خروجی :

پنجره ی خروجی در زبانه ای با پسوند lst. در محیط برنامه مشاهده می شود. و شامل اطلاعات کلیدی است.

که دارای عبارات بازتاب مدل ؛ فهرست عناصر ؛ فهرست قیود؛ فهرست ستون؛ آمار مدل؛ گزارش وضعیت؛

گزارش جواب و .. می باشد. با دوبار کلیک بر روی هر یک از عبارات متن مربوط به آن به نمایش می آید.

مدل ابتدایی : حل مسئله بهینه سازی

$$\max \quad 109x_1 + 90x_2 + 115x_3$$
$$s.t. \quad x_1 + x_2 + x_3 \le 100 \quad : equation1$$
$$6x_1 + 4x_2 + x_3 \le 500 \quad : equation2$$
$$x_1, x_2, x_3 \quad \ge 0$$

```
VARIABLES             Z;
POSITIVE VARIABLES    X1 , X2 , X3;
EQUATIONS     OBJ,  eq1 ,  eq2;
OBJ..  Z =E= 109 * X1 + 90 * X2 + 115 * X3;
eq1..             X1 +      X2 +       X3=L= 100;
eq2..          6*X1 +  4 * X2 +  8  * X3 =L= 500;
MODEL opt /ALL/;
SOLVE opt USING LP MAXIMIZING Z;
```

مدل ابتدایی : متغیر ها

$$\max \quad 109x_1 + 90x_2 + 115x_3$$
$$s.t. \quad x_1 + x_2 + x_3 \le 100 \quad : equation1$$
$$6x_1 + 4x_2 + x_3 \le 500 \quad : equation2$$
$$x_1, x_2, x_3 \quad \ge 0$$

انواع متغیرها :

```
Positive variable (nonnegative)
negative variable
Binary variable
Integer variable
(free) variable
```

در مدل بهینه سازی

```
VARIABLES            Z;
POSITIVE VARIABLES   X1 , X2 , X3;
```

مدل ابتدایی : قیود

در نرم افزار گمز قیود شامل محدودیت ها و تابع هدف می باشد

برای پیاده سازی قیود در نرم افزار گمز ابتدا قیود بیان می شوند، سپس تعریف می شوند .

بیان قیود:

کلمه کلیدی equation => نام قید => , => نام قید => , => ... => ;

```
EQUATIONS     OBJ,  eq1 ,  eq2;
```

تعریف قیود : برای تعریف هر قید:

نام قید => (..) => مدل ریاضی سمت چپ => عملگر رابطه ای => فرمول سمت راست ;

```
OBJ..          Z =E= 109 * X1 + 90 * X2 + 115 * X3;
eq1..          X1 + X2 + X3 =L= 100;
eq2..          6*X1 +  4 * X2 +  8  * X3 =L= 500;
```

مدل ابتدایی : قیود

عملگر های قیود (رابطه ای) :

عملگر رابطه ای	مفهوم
=l=	کوچکتر یا مساوی
=g=	بزرگتر یا مساوی
=e=	مساوی

```
OBJ..        Z =E= 109 * X1 + 90 * X2 + 115 * X3;
eq1..        X1 + X2 + X3 =L= 100;
eq2..        6*X1 +  4 * X2 +  8  * X3 =L= 500;
```

مدل ابتدایی : عبارت model و solve

مفهوم مدل: کلمه ی مدل دارای مفهوم گروهی از قیود می باشد.

اگر بخواهیم همه ی قیود تعریف شده در محاسبات قرار گیرند از عبارت all استفاده می کنیم :

```
MODEL opt /ALL/;
```

عبارت opt نام مدل است (نام مدل بستگی به کاربر دارد)

در مثال بهینه سازی، اگر بخواهیم محدودیت دوم در محاسبات حذف شود و فقط محدودیت اول باشد به صورت زیر عمل می کنیم :

```
MODEL opt /eq1/;
```

در حالت کلی :

```
MODEL opt /eq1, eq2, eq3, . . . /;
```

مدل ابتدایی : عبارت model و solve

حل کننده solve:

چون نوع مدل خطی و از نوع ماکزیمم سازی است ،تابع هدف Z با کلمات زیر تعریف و حل می شود :

کلمه کلیدی solve => نام مدل => کلمه کلیدی using => نوع مدل => max یا min => نام تابع هدف ;

```
SOLVE opt USING LP MAXIMIZING Z;
```

مدل ابتدایی : عبارت model و solve

تمرین:

(1) ----------

(2) ----------

(3) ----------

چند نکته عمومی :

- قاعده مهم در ترتیب عبارت این است که هیچ عبارتی در مدل پیش از بیان ،قابل ارجاع نیست.
- اختصاص چندین خط به هر عبارت، وجود خطوط خالی، و بیان چند عبارت در یک خط همگی مجازند.
- کامپایلر گمز نسبت به حروف کوچک و بزرگ حساس نیست.
- مستند سازی برای فهم بیشتر مدل ریاضی ضروری است. برای مستند سازی دست کم دو روش وجود دارد. روش اول هر سطری که با «*» (ستاره) در اولین ستون آن آغاز شود، آن سطر توسط کامپایلر گمز نادیده گرفته می شود. روش دوم متن توضیحی که شامل چند سطر می باشد در بلوک «$ontext-$offtext» درج می شود.
- نام مدل باید با یک حرف آغاز می شود

مدل استاندارد :

در نرم افزار گمز برنامه نویسی آن به گونه ای است که تبدیل کردن مسئله به مدل استاندارد بسیار کارا تر می باشد. لذا در ادامه مدل ها را به روش جدید در گمز پیاده می کنیم .

نوشتن مدل ابتدایی در حالت غیر استاندارد در نرم افزار گمز ،به 3 دلیل کارایی ندارد :

1- در مدل های با متغیر ها و پارامتر های بیشتر وقت و هزینه زیادی صرف می شود.

2-امکان خطای بیشتری وجود دارد و عیب یابی مشکل تر می شود.

3- به روز کردن آن دشوارتر است.

مدل استاندارد :

در مدل بهینه سازی ابتدا مدل را به صورت پارامتری تبدیل می کنیم :

$$\begin{aligned} \max \quad & c_1 x_1 + c_2 x_2 + c_3 x_3 \\ s.t. \quad & a_{11} x_1 + a_{12} x_2 + a_{13} x_3 \le b_1 \qquad : equation1 \\ & a_{21} x_1 + a_{22} x_2 + a_{23} x_3 \le b_2 \qquad : equation2 \\ & x_1, x_2, x_3 \quad \ge 0 \end{aligned}$$

و در نهایت مدل را استاندارد می کنیم :

$$\begin{aligned} \max \quad & \sum_{j=1}^{3} C_j X_j \\ s.t. \quad & \sum_{j=1}^{3} A_{ij} X_j \le B_i \qquad \forall i \\ & x_j \ge 0 \qquad \forall j \end{aligned}$$

مدل استاندارد :

```
SET j    /1,2,3/
    i    /1 ,2/;
PARAMETER
c(j) / 1   109,2   90 ,3   115/
b(i) /1 100 ,2 500/;
TABLE a(i,j)
     1    2    3
1    1    1    1
2    6    4    8;
POSITIVE VARIABLES     x(j);
VARIABLES PROFIT;
EQUATIONS          OBJective ,eq(i) ;
OBJective..        PROFIT=E= SUM(J,(c(J))*x(J)) ;
eq(i)..            SUM(J,a(i,J) *x(J))   =L= b(i);
MODEL     opt /ALL/;
SOLVE opt USING LP MAXIMIZING PROFIT;
```

مجموعه ها

اندیس ها در مدل، با کلمه set در برنامه گمز معرفی می شوند :

قالب set به صورت زیر می باشد :

```
SET setname optional defining text
/ firstset    optional defining text
secondset    optional defining text
...  /;
```

مثال :

```
Set i sukht
/i1 benzin
 i2 gaz
 i3 naft
 i4 metan/;
```

```
Set i
/i1
 i2
 i3
 i4 /;
```

```
Set i/i1, i2, i3, i4/;
```

```
Set i/i1*i4/;
```

مجموعه ها

برای مدل هایی که دارای دو یا چند مجموعه با اعضای یکسانند از عبارت alias به صورت زیر استفاده می کنیم :

```
Alias(j,l);
```

با اجرای این عبارت با فرض تعریف مجموعه ی j از قبل، اعضای مجموعه l دقیقا مانند اعضای مجموعه j خواهد بود.

(به طور مثال ؟ مسافت- ...)

در حالت کلی می توان بیش از دو مجموعه را نیز در alias تعریف کرد :

```
ALIAS(knownset, newset1, newset2,...);
```

زیر مجموعه ها

فرض کنیم مجموعه i به صورت زیر تعریف شده است :

```
Set i/i1*i10/;
```

اگر مجموعه p زیر مجموعه ی مجموعه i باشد ، این گونه تعریف می گردد :

```
Set i/i1*i10/
    P(i) /i1*i5,i7,i8,i10/;
```

یا

```
Set p(i);
    p(i)=yes;
    p( 'i6' , 'i9' )= no;
```

```
set setname /e1,e2,e3,e4/
    sub_set1(setname) /e1,e2/
    sub_set2(setname) /e2,e3,e4/;
```

```
set material /w,m,s,g/
    m_light(material) /w,g/
    m_heavy(material) /m,s/;
```

مجموعه ها

مثال :

```
SETs  j /x1,x2,x3/
      i /r1 ,r2/;
SET  PROCESS   PRODUCTION PROCESSES  /X1,X2,X3/;
SET  commodities Crop commodities  / corn   in bushels
                                     Wheat  in metric tons
                                     milk   in pounds
                                    cost   "cost/unit"
                                   "long-complex-*&$name"
                                    'element name'/ ;
Set  jj(j)       set to b e computed later without entries ;
$onmulti
set    i  additional entries for i  /i1,i2/;
set    composite(i,j) mullltidimentsional     /r1.set.j/;
set    kk    kk has all of i and j in it /set.i,set.j/;
```

مجموعه ها

در تعریف مجموعه ، بر چسب ها مقدار ندارند. مثلا برچسب 1970 شامل مقدار عددی 1970 نیست. و بر چسب '01' متفاوت از بر چسب '1' است .

در این قسمت دو عملگر که جایگاه و تعداد برچسب ها را نشان می دهد معرفی می کنیم :

ORD : با مثال این عملگر را توضیح می دهیم :

```
set t time periods / 2013*2017 /;

parameter val(t) ;

val(t) = ord(t);
```

نتیجه مثال بالا به این صورت است که ارزش ('2013')val مقدار 1و ('2014')val مقدار 2 و ....

کاربرد آن در محاسبه ی جمعیت : `population(t) = 75*(1.015**(ord(t) - 1)) ;`

کاربرد دیگر ؟

مجموعه ها

Card: تعداد عناصر یک مجموعه را بر می گرداند.

```
set t time periods / 2012*2014 /

parameter s,p(t),a ;

p(t)/2012 100,2013 120,2014 150/;

s = card(t);

a=(p('2012')+p('2013')+p('2014'))/s;
```

که به طور مثال میانگین تولید را در سال 2012 تا 2014 محاسبه کرد.

خطا در مدل

در صورت وجود خطا در مدل، برای هر خطا سه خط در پنجره ایجاد می شود. اولین خطا با علامت «---» آغاز می شود که شامل نام مدل، شماره ی خطی که خطا در آن رخ داده، نمایش مقدار تخصیص حافظه به وسیله کامپایلر برای پردازش مدل می باشد.

دومین خط با علامت «***» آغاز می شود و شامل کد خطا و محل گرفتن پرونده مدل است.

آخرین خط، توضیحی را در مورد کد خطای مربوط بیان می کند.

خطا در مدل

نمونه ای از خطاهای رایج :

Set q /spring,sum,fall,wtr/;

وقتی برنامه اجرا شود در پنجره خروجی نتیجه ی :

1 set q /spring,sum,fall,wtr/;

**** **$160**

Error massage

160 unique element expected

علت این خطا، این است که sum، کلمه ذخیره شده برای عمل جمع است و عناصر مجموعه باید دارای نام منحصر به فرد باشند.

خطا در مدل

نمونه ای از خطاهای رایج :

خطای دیگری که معمولا رخ می دهد قرار ندادن سمی کالون در انتهای تعریف یک قید و یا تخصیص مستقیم است. معمولا کد خطاهای ایجاد شده برای این نوع خطا $96,$97,$195,$409 می باشد.

بسیاری از خطاها تنها به دلیل اشتباه در نگارش عبارات رخ می دهند. کد خطا $170

گاهی مدل سازی از نظر ریاضی ایراد دارد و قابل بیان نیست . کد این نوع خطا معمولا $125,$149 می باشد.

به ازای هر i $\sum_i x_{ij} = 100$

داده های ورودی – اسکالر

در حالت کلی :

```
scalar item1 optional text  /numerical value/
       item2 optional text  /numerical value/
       ... ;
```

مثال :

```
Scalar f hazine sabet /1000/
       g hazine moteghayer/120/;
```

```
Scalar f /1000/, g /120/;
```

```
Scalar f /1000/
       g /120/;
```

```
Scalar f /1000/;
Scalar g /120/;
```

داده های ورودی - پارامتر

داده ورودی- پارامتر :

در حالت کلی :

```
Parameter  item(setdependency)  optional text
  / firstset  associated value,
    secondset associated value,
    ... /;
```

پارامتر های چند بعدی ؟؟!

مثال : مجموعه i قبلا تعریف شده است.

```
parameter cp(i) zarfiate anbar
/i1 150
 i2 130/;
```

```
parameter cp(i)/i1 150,i2 130/;
```

```
parameter cp(i)
/i1 150
 i2 130/;
```

```
parameter cp(i) /i1 150/;
parameter cp(i) /i2 130/;
```

داده های ورودی – (تخصیص مستقیم)

مثال :

```
set j /j1,j2,j3/;
scalar  a1;
scalars a2 /11/;
parameter   cc(j) , bc(j) /j2 22/;
a1=10;
a2=5;
cc(j)=bc(j)+10;
cc("j1")=1;
```

پارامترها می توانند بیش از یک بار مقدار دهی شوند و در نهایت آخرین مقدار داده شده را در خود نگه دارد ولی هر پارامتر بیش از یک بار نمی تواند بیان شود.

داده های ورودی

داده ورودی

مثال :

```
Set II /i1*i4000/;
Parameter x(ii),y(ii);
x(ii)=4;
Y(ii)=1;
----------------------
X(ii)=y(ii) +3;
----------------------
x("i344")=4;
----------------------
X("i344")=y("i344") +3;
```

مدل استاندارد :

$$\max \sum_{j=1}^{3} C_j X_j$$

$$s.t. \quad \sum_{j=1}^{3} A_{ij} X_j \le B_i \qquad \forall i$$

$$x_j \ge 0 \qquad \forall j$$

که در آن :

J={1,2,3}

I={1,2}

Xj={x1,x2,x3}

Cj={109,90,115}

Aij= 1 1 1

6 4 8

Bi={100,500}

سیگما (sum)

اگر مدل ریاضی دارای سیگما باشد :

به صورت زیر بیان می شود :

$$\sum_i x_i = x_1 + x_2 + x_3$$

```
Z= sum(i,x(i));
```

برای حالت بیشتر از یک سیگما:

$$\sum_i \sum_j (...) x_{ij...} = x_{11...} + x_{12...} + ... + x_{IJ...}$$

```
Z=sum((i,j,...),x(i,j,...))
```

پرود (prod)

اگر مدل ریاضی به صورت زیر باشد :

به صورت زیر بیان می شود :

$$\prod_i x_{ij...} = x_1 \times x_2 \times x_3 \times ...$$

```
Z= prod(i,x(i));
```

برای حالت بیشتر از یک اندیس:

$$\prod_i \prod_j ... x_{ij...} = x_{11...} \times x_{12...} \times ... \times x_{IJ...}$$

```
Z=prod((i,j,...),x(i,j,...))
```

(Smax و Smin)

اگر متغیر X به صورت زیر باشد : $x(i) = x(1), x(2), x(3), x(4), \ldots$

و بخواهیم کوچکترین عضو و بزرگترین عضو مجموعه را تعیین کنیم از دستورات زیر استفاده می کنیم :

```
Z= Smax(i,x(i));
Z= Smin(i,x(i));
```

ببیشترین مقدار عضو مجموعه

کمترین مقدار عضو مجمـوعه

```
set i/1*5/;
parameter x(i)/1 2,2 5,3 7,4 3,5 1/;
scalar r,q;
r=smax(i,x(i));
q=smin(i,x(i));
display r,q;
```

ساختار مدل گمز:

ورودی ها	خروجی
-Sets	بازتاب مدل
بیان اندیس ها	پیغام های خطا
تعریف(تخصیص مقادیر) اندیس ها	فهرست عناصر (شناسه ها)
-Data : (parmeter, table, scalar)	فهرست قیود
بیان داده ها (پارامترها، جدول ها، اسکالرها)	فهرست ستون
تعریف مقادیر داده ها	آمار مدل
-Variables	گزارش وضعیت
بیان متغیرها	گزارش جواب
تعیین نوع متغیر ها	
تخصیص کران ها و مقدار اولیه	
- Equations	
بیان تابع هدف و محدودیت ها	
تعریف تابع هدف و محدودیت ها	
-Solve , model	
-Display	

عبارات شرطی:

جدول عملگرها:

Operation	*Operator*
Exponentiation	**
Numerical Operators	
- Multiplication, Division	*, /
- Unary operators - Plus, Minus	+, -
- Binary operators - addition, subtraction	+, -
Numerical Relationship operators	<, <=, =, <>, >=, >
Logical Operators	
- not	not
- and	and
- or, xor	or, xor

عبارات شرطی:

شرایط منطقی:

بیان خاصی از ارزیابی یک مقدار عددی که می تواند True یا False باشند.

- بیان عددی به عنوان شرایط منطقی:

مثال :عبارت 2*a-4 به ازای مقدار a=2 دارای شرایط مقداری false زیرا مقدار عبارت برابر صفر است و به ازای بقیه مقادیر True می باشد.

- عملگرهای رابطه ای عددی(جدول ()) :

مثال :عبارت (sqr(a) gt a) به ازای -1<a<1 false و به ازای مقادیر دیگر True می باشد.

عبارات شرطی:

- شرایط منطقی – عملگر های منطقی{ جدول () } :

Operands		*Results*			
`a`	`b`	`a and b`	`a or b`	`a xor b`	`not a`
0	0	0	0	0	1
0	non-zero	0	1	1	1
non-zero	0	0	1	1	0
non-zero	non-zero	1	1	0	0

- مقدار عددی True و False : طبق قرار داد مقدار عددی True برابر یک و مقدار عددی False مقدار صفر تعیین شد.

x = (1 < 2) + (2 < 3) ===> 2 ; x = (1 < 2) or (2 < 3) ===> 1;

عبارات شرطی:

- شرایط منطقی – ترکیبی:

مثال :

Logical Condition	*Numerical Value*	*Logical Value*
`(1 < 2) + (3 < 4)`	2	True
`(2 < 1) and (3 < 4)`	0	False
`(4*5 - 3) + (10/8)`	17.125	True
`(4*5 - 3) or (10 - 8)`	1	True
`(4 and 5) + (2*3 <= 6)`	2	True
`(4 and 0) + (2*3 < 6)`	0	False

عبارات شرطی

عملگر $: به این معناست که اگر عبارت منطقی صحیح باشد عبارت انجام شود.

```
Term $ logical condition
```

تخصیص مستقیم :

```
A=2;
```

تخصیص شرطی :

```
A$( b is true)= number value;
A$( b>1.5)= 2          ;      if (b > 1.5,A = 2)
```

اگر عبارت منطقی درست نباشد، a مقدار جدید تخصیص یافته را نمی پذیرد. در غیر اینصورت به a مقدار جدید تخصیص می یابد.

```
A=number value$(b is true);
```

اگر عبارت منطقی درست نباشد، a مقدار صفر را می پذیرد. در غیر اینصورت به a مقدار جدید تخصیص می یابد.

مثال :

```
scalar a,b;
a=1; b=2;
a$(b>2)=3;
```

```
scalar a,b;
a=1; b=2;
a=3$(b>2);
```

عبارات شرطی

عملگر های رابطه ای :

رابطه	عملگر های گمز
مساوی	Eq or =
نا مساوی	Ne or <>
بزرگتر مساوی	Ge or >=
بزرگتر	Gt or >
کوچکتر مساوی	Le or <=
کوچکتر	Lt or <

عبارات شرطی

حالت هایی که عملگر $ استفاده می شود :

```
namedparameter$logical condition=term;
namedparameter(setelementdependency)$logical condition=term;
-------------------------------------
X$(qq gt 0)=3;
qq $(sum(I,q(i)) gt 0)=4;
qq $ (sum(i,abs(a(i))) gt 0)=1/sum(i,a(i));
a(i) $(qq gt 0) = q(i)+a(i);
a(i) $a(i) = q(i)/a(i);
```

```
scalar a,b;
a=1; b=2;
a$(b>2)=3;
```

```
scalar a,b;
a=1; b=2;
a=3$(b>2);
```

عبارات شرطی

حالت هایی که عملگر $ استفاده می شود :

```
Namedparameter = term1+term$logical condition    or
Namedequation.. term1+term$logical condition =L= other terms;
--------------------------------
qq     =  qq+1$(x gt 0);
qq     =  1$(x gt 0);
q(i)   =  a(i)+1$(a(i) gt 0);
q(i)   =  a(i)$(a(i) gt 0);
X      =  sum(I,q(i))$(qq gt 0)+4;
Eq4..     xvar+yvar$(qq gt 0)=e=3;
Eq5(i)..  ivar(i)$(a(i) gt 0)+yvar$(qq gt 0)=e=3;
```

عبارات شرطی

حالت هایی که عملگر $ استفاده می شود :

محدود کردن اندیس ها :

```
sum(namedset$set dependent logical condition ,term)
---------------------------------------
X=sum(I$(q(i) gt 0),a(i));
X=smin(I$(q(i) gt 0),q(i));
X=sum(I$(q(i) gt 0),1));
eq6(i)..  sum(j$(cost(i,j) gt 1),cost(i,j)*tran(i,j))=e=1;
```

عبارات شرطی

حالت هایی که عملگر $ استفاده می شود :

```
equation name$logical condition..  equation specification;
-----------------------------------
Eq1$(qq gt 0)..           xvar=e=3;
Eq2$(sum(I,q(i)) gt 0)..  yvar=l=4;
Eq3(i)$(a(i) gt 0)..      ivar(i)=g= -a(i);
Eq7(i)$(qq gt 0)..        sum(j,ijvar(I,j))=g= -a(i);
```

عبارات شرطی

حالت هایی که عملگر $ استفاده می شود :

```
display$condition listofitems;
if(condition, display listofitems);
```

```
scalar x /0/,y /10/;
display$(x+y le 0) "display when sum of x and y le 0",x,y;
x=2;
if(x gt 0,
display "X and y here if x is positive",x,y;);
display$(x > -1) "display with display$ at second place",x;
if((x+y > 2),
display "X and y at first place if x+y is greater than 2",x,y;);
if(x gt 0,display "X and y at first place if x is
positive",x,y;);
```

داده ورودی - جدول

در حالت کلی :

```
Table itemname (setone, settwo, ... ) descriptive text
                    set_2_element_1   set_2_element_2
set_1_element_1     value_11          value_12
set_1_element_2     value_21          value_22;
```

جدول چند بعدی ؟؟!

مثال :

مجموعه t, i قبلا تعریف شده اند.

```
table d(i,t) taghaza
    t1    t2    t3
i1  10    12    14
i2  11    15    7;
```

```
table d(i,t) taghaza
    t1    t2    t3
i1  10          14
i2        15    7;
```

متغیر

همانطور که گفته شد، متغیر ها درای انواع مختلف می باشند. در جدول زیر نوع متغیر و بازه ی متغیر آورده شده است :

نوع متغیر	بازه ی متغیر
Free	(-inf,+inf)
Posotive	(0,+inf)
Negative	(-inf, 0)
Binary	0 or 1
Integer	0,1,2,...,100

متغیر ها در هنگام بیان ،مقدار دهی اولیه نمی شوند. ولی کاربر می تواند با استفاده از پسوند های متغیر، مقدار اولیه و کران های متغیر را تعیین کند :

. l => تعیین مقدار آغازین برای متغیر
. up => تعیین مقدار برای زوج مکمل (دوگان)

. lo => تعیین کران پایین برای متغیر
. up => تعیین کران بالا برای متغیر
. fx => تعیین مقدار ثابت برای متغیر

متغیر

مثال :

```
set j/j1, j2/;
positive variable x(j)/j1.up 10, j2.lo 20, j2.up 30/;
variable z;
equation
func;
func.. z=e=x( 'j1')+x( 'j2');
model simple /all/;
solve simple using lp minimizing z;
```

قیود

بیان قید :

```
Firsteqname(setdependency)  optional explanatory text
/optional values for attributes/
secondeqname(setdependency) optional explanatory text
/optional values for attributes/
...
;
```

در بیان قید عبارت داخل پرانتز هم ارز « ∀ » در مدل ریاضی می باشد.

مشخصات قید : (به ازای هر قیدی که بیان کرده ایم ، مشخصات (رابطه ریاضی) وجود دارد)

```
equation name(setdependency)$optional logical condition ..
lhs_equation_terms   equation_type   rhs_equation_terms;
```

قیود


```
equation name(setdependency)$optional logical condition ..
lhs_equation_terms   equation_type   rhs_equation_terms;
```


دقت کنید که علامت = تنها در تخصیص مستقیم پارامتر استفاده می شود. و =e= تنها در تعریف معادله به کار می رود

.تعیین مستقیم مقدار مورد نظر را قبل از حل به پارامتر می دهد. اما در تعریف یک معادله تا قبل از حل شدن ارضا نمی

شود. به همین دلیل تعریف معادله شامل متغیر هاست اما تعیین مستقیم پارامتر ها نباید دارای متغیر شود.

Model - Solve

مدل :

```
Model modelname /firesteqname,secondequname,.../;
Model modelname /all/;
```

:Solve

```
Solve modelname using modeltype maximizing varname;
Solve modelname using modeltype minimizing varname;
```

Model - Solve

انواع مدل ها :

Lp	مدل بهینه سازی که نمی تواند عبارات غیر خطی یا متغیر های گسسته داشته باشد
Nlp	مسئله بهینه سازی که محتوای عبارات غیر خطی شامل توابع هموار باشد اما متغیر ها گسسته نباشند
Dnlp	مسئله nlp با این تفاوت که توابع نا هموار در این نوع ظاهر می شود
Mip	مسئله بهینه سازی شامل متغیر های گسسته اما عبارت خطی می باشند
Minlp	مسئله بهینه سازی شامل متغیر های گسسته و عبارات غیر خطی می باشند
Qcp	در این مدل عبارت خطی و درجه ی دوم موجود هستند اما متغیر ها پیوسته می باشند
Miqcp	در این مدل عبارت خطی و درجه دوم موجود هستند و متغیرها پیوسته و گسسته (عدد صحیح،باینری) می باشند
R() Miqcp Minlp Mip	مدل های روبرو که آمیخته با متغیر های گسسته می باشند ، به صورت آزاد شده حل می شوند یعنی فرض می شود متغیر ها همگی پیوسته اند ! (Relaxed) Rmiqcp –Rminlp-Rmip

Model - Solve

انواع مدل ها : مدل خطی

Lp	$\text{minimize or maximize } cx$ $\text{subject to } Ax \ \alpha \ b$ $x \geq L$ $x \leq u$

Model - Solve

انواع مدل ها : مدل غیر خطی

Nlp	$\text{minimize or maximize } f(x)$ $\text{subject to } g(x) \ \alpha \ 0$ $l \leq x \leq u$ F, g تابع غیر خطی با مشتقات پیوسته می باشند (تابع های دارای مشتقات نا پیوسته مانند : max،min،abs)

Model - Solve

انواع مدل ها : برنامه ریزی آمیخته با عدد گسسته

Mip	minimize or maximize $c_1 t + c_2 u + c_3 v$ subject to $t \geq 0$ $0 \leq u \leq L$ integer $v \in (0,1)$

Model - Solve

انواع مدل ها : برنامه ریزی غیر خطی میخته با اعداد گسسته

Minlp	minimize or maximize $f(x) + d(y)$ subject to $g(x)+h(y)\ \alpha\ 0$ $L \leq x \leq U$ integer $y \in \{0,1,2,\ldots\}$

display

برای بدست آوردن مقدار بهینه اولیه و یا دوگان متغیر ما می توانیم در خروجی حل کننده اطلاعات را بدست بیاوریم

یا می توانیم با استفاده از دستور Display با یک قالب مناسب تر به طور مجزا جدولی از خروجی متغیر ایجاد کنیم:

```
Display z.l, x.l, x.m;
```

که در آن z.l مقدار بهینه تابع هدف مساله اولیه و x.l مقدار بهینه x در مساله اولیه و x.m مقدار بهینه x در مساله دوگان می باشد.

مهمترین پسوند ها در متغیرها یا قیود .up ,.lo ,.m ,.l می باشند.

برای نشان دادن داده های ورودی بدون وابستگی های متقابل به صورت زیر عمل می کنیم :

```
set i/1*10/, j(i)/1*5, 8/;
parameter
c(i)/1 10, 5 15, 7 11/, d(j)/5 4/;
scalar a, b;
a=1;b=2;a=2$(b>2);
display a, b, d, c, i, j;
```

انواع option ها :

برای تغییر نمایش تعداد قید گسترش یافته به ازای دسته قیود در فهرست قیود پنجره خروجی	Option limrow= number
برای تغییر نمایش تعداد متغیرهای گسترش یافته به ازای هر دسته متغیر در فهرست ستون پنجره خروجی	Option limcol=number
برای تغییر زمان اجرای حل کننده برای رسیدن به جواب در گزارش وضعیت پنجره خروجی	Option reslim=number
برای تغییربارست های لازم برای رسیدن به جواب بهینه در گزارش وضعیت پنجره خروجی	Option iterlim=number
به صورت پیش فرض مقدار gap برابر 0.10 است اما می توانیم مقدار gap را تغییر دهیم :	Option optcr=number Number=[0,1]

وضعیت حل کننده :

1	Normal compeletion	به این معناست که حل کننده جواب را در یک روش نرمال بدست آورده است
2	Iteration interrupt	حل کننده به دلیل رسیدن به کران بالای تعداد بارست ها متوقف شده است
3	Resource interrupt	حل کننده به دلیل رسیدن به کران بالای زمان اجرا متوقف شده است
4	Terminated by solver	حل کننده به علت ناتوانی در ادامه ی مراحل اجرا متوقف شده است.پیغام هایی بعد از این وضعیت مشاهده می شود
5	Evaluation error limit	به این معناسات که در یک عبارت غیر خطی از مقداری تعریف نشده استفاده می شود. مثلا وجود صفر در مخرج یک کسر
6	Unknown error Preprocessor Error setup failure Error solver failure Error internal solver Error system failure	همه این پیغام ها برخی ویژگی های پیش بینی نشده Gams ، حل کننده، یا بین این دو را اعلام می کند. (باید پنجره خروجی را برای یافتن خطا به طور کامل بررسی کرد)

وضعیت های مدل:

1	optimal	به معنای رسیدن به جواب بهینه است. در مساله های lp و Rmip
2	localy optimal	به معنای وجود بهینه موضعی در مسائل برنامه ریزی غیر خطی
3	unbounded	به معنای وجود جواب بیکران در مسئله
4	Infeasible	مسئله نشدنی است .وضعیت معمولا به دلیل وجود خطا در منطق مدل یا داده ها رخ می دهد
5	Locally infeasible	به این معناست که هیچ جواب شدنی با شروع از جواب شدنی اولیه برای مسئله برنامه ریزی غیر خطی حاصل نشده است.این مطلب لزوما به معنای نبود هیچ جواب شدنی نیست
6	intermediate infeasible	به این معنا که جواب جاری نشدنی است و حل کننده به علت محدودیت در زمان اجرا ی برنامه یا تعداد بارست ها یا به علت برخی اشکالات متوقف شده است
7	Intermediate nonoptimal	به معنای عدم وجود جواب بهینه است
8	Integer solution	به معنای وجود جواب با اعداد صحیح برای مسئله mip است
9	Intermadiate noninteger	به معنای نرسیدن به یک جواب بهینه با اعداد صحیح برای مسئله mip است.
10	integer infeasible	به معنای نبود جواب با اعداد صحیح برای مسئله آمیخته با اعداد صحیح است
11	Error unknown Error no solution	به معنای اینکه جوابی در هیچ یک از این وضعیت ها وجود ندارد . در صورت مواجه با این وضعیت ، بررسی دوباره ساختار مدل توصیه می شود