معرفی اولیه نرم افزار:

نرم افزار*Lingo* جهت تحلیل مدلهای ریاضی خطی و غیر خطی طراحی شده است و قابل اجرا بر روی کامپیوترهای شخصی می باشد. نرم افزار *Lingo* ، امروزه به عنوان پرکاربردترین نرم افزار تحقیق در عملیات ، جایگاه خاصی را چه در محیط های علمی و چه در صنایع و شرکتهای بزرگ به خود اختصاص داده است. این نرم افزار نسبت به سایر نرم افزار های تحقیق در عملیات مانند *Win QSB, GAMS , DS , Storm , Lindo , Excel Solver* و ... دارای قابلیتهای بسیار بالایی می باشد که آن را به طور بارزی از این نرم افزارها متمایز می کند. قدرت برقراری ارتباط با نرم افزارهای صفحه گسترده و بانکهای اطلاعاتی برای تبادل داده ها ، توابع گسترده ریاضی ، سادگی استفاده و بهره گیری از روشهای حل (*Solver*) متنوع ، قدرت تحلیل حساسیت و توانایی حل مدلهای غیر خطی ، تنها تعداد معدودی از قابلیتهای بالای نرم افزار *Lingo* می باشد.

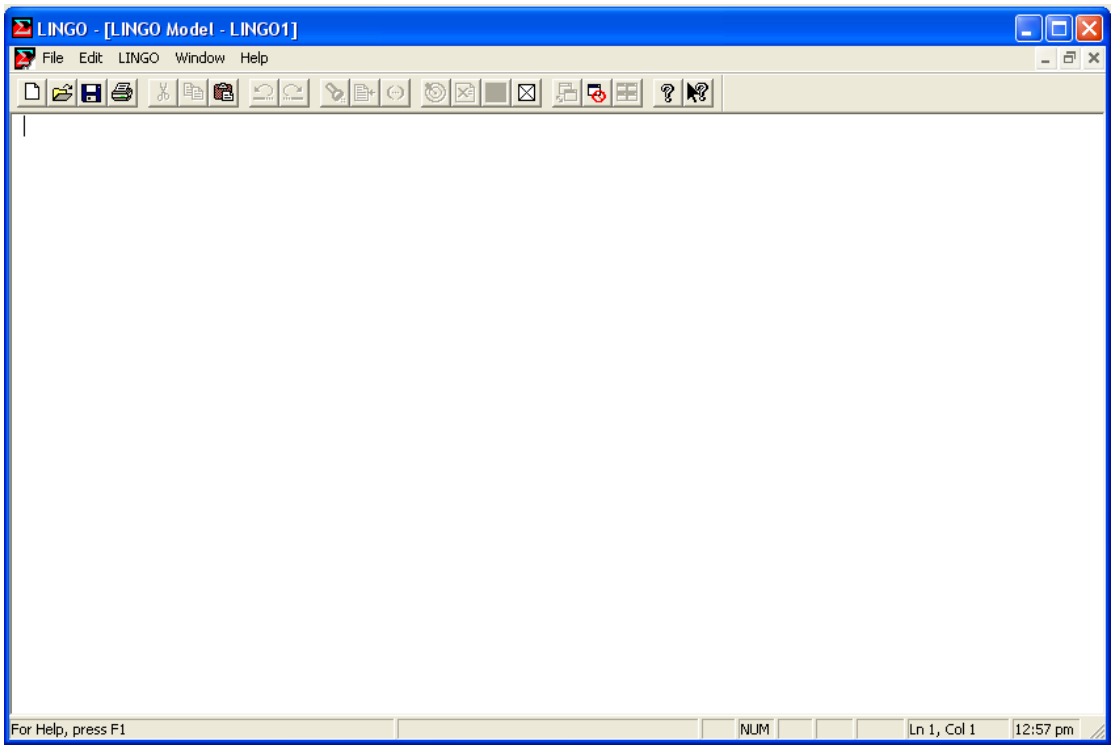
نرم افزار *Lingo* علی رغم سادگی یادگیری ، نرم افزار بسیار کاربردی و قابل استفاده حتی در سطوح بالای مدیریتی است و در عین حال دارای قابلیتهای *User Friendly* است که آن را به یکی از پرکاربردترین نرم افزارهای تحقیق در عملیات تبدیل کرده است.

در این مجموعه ، سعی شده است تا نحوه مدلسازی و حل مدلهای ریاضی در نرم افزار *Lingo8* تا حد امکان تشریح شده و به کابران در استفاده از این نرم افزار ، راهنمایی های لازم ارائه گردد.

صفحه اصلی نرم افزار:

شکل زیر صفحه اصلی نرم افزار را نشان می دهد. همانطور که در شکل مشاهده می شود، صفحه اصلی از چهار قسمت زیر تشکیل شده است :

- صفحه مدل که به رنگ سفید در شکل قابل مشاهده است و مدل مسأله مورد نظر باید در آن وارد شود.
- نوار فهرست (*Menu Bar*) که شامل کلیه فرمانهای نرم افزار است و در بالای صفحه قابل رؤیت می باشد.
- نوار ابزار (*Tools Bar*) هم مشابه کلیه نرم افزارهای تحت ویندوز، پایین تر از نوار فهرست قرار گرفته و شامل آیکون هایی است که فرمان های متداول نرم افزار را اجرا می کنند.



شکل۱ : صفحه اصلی نرم افزار *Lingo*

ساخت یک مدل جدید :

ساخت یک مدل جدید را باید از صفحه مدل آغاز کرد. در این صفحه ، کلمات کلیدی نرم افزار که از پیش برای آن تعریف شده اند ، به رنگ آبی ، توضیحاتی که توسط برنامه در نظر گرفته نمی شوند ، به رنگ سبز و سایر اجزای برنامه به رنگ سیاه نشان داده می شوند. در این صفحه ، پس از نوشتن هر خط برنامه ، باید از علامت " **;** " استفاده کرد. برای نامگذاری متغیرها می توان از حروف ، اعداد و کاراکتر " _ " استفاده کرد ، اما نام متغیر ، حتما باید با یک حرف آغاز شود. برای نوشتن توضیحاتی که می خواهیم توسط برنامه در نظر گرفته نشوند و هدف از آوردن آنها فقط قابل فهم تر شدن برنامه برای خواننده است ، از علامت " **!** " قبل از توضیحات استفاده کرده و توضیحات را با علامت " **;** " به پایان می بریم .

برای وارد کردن مدل در نرم افزار *Lingo* ، کافی است که دقیقا همان مدل استاندارد نوشته شده روی کاغذ را به همان شکل وارد مدل کنیم . این نرم افزار قابلیت آنالیز مدل به همان شکل استاندارد را دارد. برای مثال مدل زیر را روی کاغذ ساخته ایم . برای وارد کردن آن در *Lingo* کافی است آن را به شکل زیر در نرم افزار وارد کنیم :

$$Max(2x_1 + 7x_2)$$
$$4x_1 - 3x_2 + x_3 \le 10$$
$$8x_2 - 2x_3 \ge 6$$

مدل *Lingo* ⟵

```
Max = 2*X1+7*X2;
4*X1-3*X2+X3<=10;
8*X2-2*X3<=6;
```

• نوشتن تابع هدف :

همانطور که در مثال ساده بالا مشاهده شد ، تابع هدف بسته به اینکه به صورت حداکثرسازی یا حداقل سازی باشد ، به شکل " *Max = ...* " یا " *Min = ...* " نوشته می شود.

• چارچوب کلی مدل و دستورات مربوطه :

در آغاز نوشتن هر مدل ، می توان از دستور " *Model :* " استفاده کرد. پس از آن ، دستور " **;** عنوان مدل *Title :* " برای انتخاب یک نام دلخواه برای مدل به کار می رود که البته استفاده از این دستور ، اختیاری است. سپس کلیه ارکان مدل شامل تعریف

مجموعه ها ، ورود داده ها ، تابع هدف و محدودیتها ، نوشته خواهد شد و مدل با دستور " *End* " به پایان می رسد. دو دستور " *Model :* " و " *End* " ، نیازی به " **;** " ندارند.

مثال)

```
model:
title students;
max=2*x1+7*x2;
4*x1-3*x2+x3<=10;
8*x2-2*x3<=6;
end
```

خروجی پس از زدن دکمه Solve () :

```
Global optimal solution found at iteration:            0
  Objective value:                                91.00000

                     Variable           Value        Reduced Cost
                           X1        0.000000            26.00000
                           X2        13.00000            0.000000
                           X3        49.00000            0.000000

                          Row    Slack or Surplus      Dual Price
                            1        91.00000            1.000000
                            2        0.000000            7.000000
                            3        0.000000            3.500000
```

مثال)

```
MAX = 20 * A + 30 * C;
A < 60;
!note that < and <= are equivalent;
!in LINGO;
C < 30;
A + 2 * C < 120;
```

حل:

```
  Global optimal solution found at iteration:            2
Objective value:                                2100.000

                     Variable           Value        Reduced Cost
                            A        60.00000            0.000000
                            C        30.00000            0.000000

                          Row    Slack or Surplus      Dual Price
                            1        2100.000            1.000000
                            2        0.000000            20.00000
                            3        0.000000            30.00000
                            4        0.000000            0.000000
```

که reduced cost همان z_j-c_j ها هستند.

تذکر: بطور پیش‌فرض متغیرها بزرگتر-مساوی صفر در نظر گرفته می شوند و نیازی به نوشتن این محدودیت ها در انتهای مدل نیست.

می توان برای محدودیت ها نام‌گذاری کرد:

```
MODEL:
! Design a box at minimum cost that meets
 area, volume, marketing and aesthetic requirements;
 [COST] min = 2*( .05*(d*w + d*h) +.1*w*h);
 [SURFACE]  2*(h*d + h*w + d*w) >= 888;
 [VOLUME]   h*d*w >= 1512;
! These two enforce aesthetics;
 [NOTNARRO] h/w <= .718;
 [NOTHIGH]  h/w >= .518;
! Marketing requires a small footprint;
 [FOOTPRNT] d*w <= 252;
END
```

پس از حل:

```
Local optimal solution found at iteration:           181
  Objective value:                                50.96507


                     Variable           Value        Reduced Cost
                            D        23.03098            0.000000
                            W        9.562189            0.000000
                            H        6.865656            0.000000

                          Row    Slack or Surplus      Dual Price
                         COST        50.96507          -1.000000
                      SURFACE     -0.5989596E-04     -0.2342592E-01
                       VOLUME     -0.4623156E-03     -0.1329932E-01
                     NOTNARRO     -0.4761957E-06       2.298538
                      NOTHIGH       0.2000005          0.000000
                     FOOTPRNT        31.77348          0.000000
```

که یک مثال غیرخطی است و لذا تضمین بهینگی جواب وجود ندارد.

مدل سازی به زبان *Lingo*:

اما همه مدلهای ما مانند مثال بالا نیستند. بسیاری از مدلهای ریاضی در عمل شامل هزاران متغیر و محدودیت می باشند به طوری که عملا وارد کردن مدل به شکل فوق در نرم افزار ، کاری بسیار طاقت فرسا و در اغلب موارد حتی غیر ممکن است. در اصل ،

ابزار های نرم افزار *Lingo* شامل دستورات ، توابع ، عملگرها و ... ، ما را قادر می سازد تا این مدلهای بزرگ را فقط در چند خط ، وارد برنامه کرده و به سادگی ، جوابهای مورد نظر را از مدل دریافت کنیم. در ادامه سعی می شود تا این ابزارها تشریح شوند :

- **عبارات ریاضی :**

نوشتن عبارات ریاضی در *Lingo* ، بسیار شبیه آنچه است که در سایر برنامه ها دیده می شود. در نوشتن عبارات ریاضی ، از چند اپراتور بسیار ساده به شرح زیر استفاده می شود :

اپراتورهای ریاضی	نماد
جمع	+
تفریق	-
ضرب	*
تقسیم	/
توان	^

عبارات ریاضی می توانند شامل چهار عمل اصلی و توان باشند. برای نشان دادن هر یک از موارد مذکور ، از کاراکترهای متناظرشان استفاده می شود.

اولویت اپراتورهای مذکور به این شرح است که توان دارای اولویت اول است. اولویتهای بعدی از آن ضرب و تقسیم هستند و اولویت آخر هم با جمع و تفریق است.

اگر بخواهیم ترتیب خاصی برای اجرای اپراتورهای موجود در یک عبارت ریاضی اتخاذ کنیم ، از پرانتز در مکان مناسب استفاده می کنیم . برای مثال ، با توجه به مطالب مذکور داریم :

$$9 + 6 * 4 \wedge 2 / 3 = 41$$

$$((9+6) * 4) \wedge 2 / 3 = 1200$$

- **نامساوی ها (و مساوی ها) :**

نامساوی ها (و مساوی ها) یکی از پرکاربردترین ابزارها در مدلسازی می باشند. در اصل هر یک از محدودیتهای مسأله ، یک تساوی یا نامساوی هستند. برای وارد کردن تساوی ها و نامساوی ها در مدل ، از اپراتورهای رابطه ای استفاده می شود. اپراتورهای رابطه ای شامل موارد زیر هستند :

اپراتورهای رابطه ای	نماد
بزرگتر مساوی	>=
کوچکتر مساوی	<=
تساوی	=

نرم افزار *Lingo* ، بزرگتر اکید یا کوچکتر اکید را نمی پذیرد و کاراکترهای " < " و " > " را هم به ترتیب به عنوان بزرگتر مساوی و کوچکتر مساوی در نظر می گیرد. برای وارد کردن عبارات بزرگتر اکید مانند " $A > B$ " ، باید آن را به صورت " $A + e$ >=B " وارد کرده و بسته به آن که " A " چقدر از " B " بزرگتر باشد ، محدودیت " ... e >= " هم به مدل اضافه می شود.

- **عبا رات شرطی :**

عبارات شرطی در شرایطی به کار می رود که بخواهیم یک دستور را به یک سری متغیرهای خاص محدود کنیم . هر عبارت شرطی می تواند مقدار " درست " یا " غلط " بگیرد. اگر عبارت شرطی ، مقدار " درست " بگیرد ، دستور مطرح شده ، روی متغیرهای مورد نظر اجرا می شود و اگر مقدار " غلط " بگیرد ، دستور مذکور اجرا نمی شود. نرم افزار *Lingo* ، مقدار ۱ را برای عبارات شرطی " درست " و مقدار صفر را برای عبارات شرطی " غلط " اختصاص می دهد. برای نوشتن عبارات شرطی ، از اپراتورهای منطقی استفاده می شود. اپراتورهای منطقی عبارتند از :

اپراتور منطقی	نماد
نقیض	**#NOT#**
تساوی	**#EQ#**
مخالف	**#NE#**
بزرگتر	**#GT#**
بزرگتر مساوی	**#GE#**
کوچکتر	**#LT#**
کوچکتر مساوی	**#LE#**
و	**#AND#**
یا	**#OR#**

اولویت اجرای اپراتورهای منطقی به ترتیب زیر است که اولویت اول با " *#NOT#* " است. پس از آن عملگرهای " *#EQ#* " ، " *#NE#* " ، " *#GT#* " ، " *#GE#* " ، " *#LE#* " و " *#LT#* " در اولویتند. اولویت آخر هم با عملگرهای " *#AND#* " و " *#OR#* " است.

در زیر چند نمونه از عبارات شرطی نمایش داده شده است.

- x بزرگتر مساوی ۵ نباشد : *#NOT# (x #GE# 5)*
- x کوچکترمساوی ۳ و y بزرگتر از ۴ باشد : *(x #LE# 3) #AND# (y #GT# 4)*
- x مساوی ۳ باشد یا y مخالف ۲ باشد : *(x #EQ# 3) #OR# (y #NE# 2)*

- **مجموعه ها :**

مجموعه ها یکی از قوی ترین ابزارهای نرم افزار *Lingo* هستند که کار مدلسازی را بسیار تسهیل می کنند. در مدلهای عملی ، به دفعات مشاهده می شود که یک دسته از متغیرها یا پارامترها ، متناظر با یک مجموعه هستند ، مثلا مجموعه شهرها ، کارخانه ها ، محصولات ، افراد ، مکانها ، دوره های زمانی و ... .

همه اعضای مجموعه ، دارای یک سری خصوصیات مشابه هستند که بین همه آنها مشترک است ؛ لذا به جای تعریف کردن هر عضو به عنوان یک متغیر یا پارامتر جداگانه ، می توان یک مجموعه تعریف کرد و کلیه مشخصه های مشترک را روی همه اعضای مجموعه تعمیم داد . در صورتی که تعدادی از اعضای مجموعه ، دارای برخی از مشخصه ها نباشند ، به کمک عبارات شرطی می توان آنها را از سایر اعضای مجموعه متمایز کرد. همچنین تعریف مجموعه ها ، ما را در نوشتن محدودیتها به صورت جمعی کمک می کند به طوری که به جای نوشتن چندین محدودیت مشابه در مدل استاندارد روی کاغذ ، می توان فقط از یک محدودیت مشترک برای اعضای مجموعه استفاده کرد.

مجموعه ها می توانند یک یا چند بعدی باشند . مجموعه یک بعدی ، مجموعه ای است که با اعضای آن تعریف می شود . مجموعه های چند بعدی ، مجموعه هایی هستند که روی دو یا چند مجموعه یک بعدی تعریف می شوند.

برای تعریف مجموعه ها از دستور " *Sets :* " استفاده می شود. پایان تعریف مجموعه ها در برنامه با دستور " *Endsets* " مشخص می شود.

Sets :

...

...

...

Endsets

تعریف مجموعه ها

برای تعریف یک مجموعه یک بعدی ، به شکل زیر عمل می کنیم :

; نام مشخصه یا مشخصه های اعضای مجموعه : / نام اعضای مجموعه / نام مجموعه

هر مجموعه می تواند از صفر تا چندین مشخصه داشته باشد. منظور از مشخصه ، ویژگی هایی است که بین تمام اعضای مجموعه مشترک است. برای مثال ، در مجموعه محصولات ، مواردی چون رنگ ، اندازه ، قیمت ، جنس ، طول عمر و ... همه می توانند به عنوان مشخصه مجموعه محصولات تعریف شوند ؛ یا در مجموعه شهرها ، مواردی چون جمعیت ، مساحت ، متوسط بارندگی ، درجه حرارت متوسط ، تعداد واحدهای صنعتی و ... ، می توانند به عنوان مشخصه های مجموعه شهرها تعریف شوند. انتخاب مشخصه ها برای مدل ، بستگی به مسأله مورد بررسی دارد. مشخصه های هر مجموعه هم می توانند پارامترهای مشخص و معلوم باشند و هم می توانند متغیرها و مجهولاتی باشند که هدف ما ، یافتن مقادیر آنهاست. مثلا برای تعریف مجموعه شهرها با دو مشخصه جمعیت و مساحت ، به این صورت عمل می کنیم :

cities / London , Paris , New York , Tehran / : population , area ;

اگر یک مجموعه یک بعدی ، زیر مجموعه ای از یک مجموعه یک بعدی دیگر باشد ، باید نام اعضایی که در مجموعه کوچکتر هستند را مانند مثال زیر مشخص کنیم :

; نام مشخصه های مجموعه کوچکتر *:* / نام اعضای مجموعه کوچکتر / (نام مجموعه مرجع) نام مجموعه کوچکتر

subcities (cities) / Paris , Tehran / : towns ;

دستور بالا ، یک زیرمجموعه دو عضوی از مجموعه شهرها تعریف می کند و یک مشخصه هم که تعداد شهرهای تابعه آن شهر است ، به آن اختصاص می دهد.

در تعریف مجموعه ها ، می توان به جای نوشتن اعضای مجموعه ، از اعداد (اندیس ها) استفاده کرد. مثلا برای تعریف یک مجموعه ۱۰ عضوی که ۱۰ نوع محصول یک کارخانه را نشان می دهد و مشخصه های قیمت و موجودی انبار برای هر محصول را تعریف می کند ، داریم :

products / 1..10 / : price , inventory ;

> در حالتی که نام اعضای مجموعه را می نویسیم ، خود نرم افزار ، اندیس ها را به ترتیب ، به اعضا اختصاص می دهد. برای مثال در مورد فوق ، اندیس اول به شهر " *London* " اختصاص می یابد. در حالت کلی برای دستیابی به اعضای مجموعه ، از اندیس استفاده می شود. برای مثال ، " *population (1)* " ، به معنی جمعیت شهر " *London* " است و " *price (5)* " ، به معنی قیمت محصول پنجم است.

مجموعه های چندبعدی مجموعه هایی هستند که روی اعضای مجموعه های یک بعدی تعریف می شوند. برای مثال ، برای بیان تعداد هر نوع محصول که به هر شهر ارسال شده است ، ما هم به مجموعه محصولات و هم به مجموعه شهرها نیاز داریم. بنابراین مجموعه کالاهای حمل شده ، روی اعضای دو مجموعه مذکور تعریف می شود. برای تعریف مجموعه های چندبعدی به صورت زیر عمل می کنیم :

; نام مشخصه های مجموعه چندبعدی : (نام مجموعه های یک بعدی) نام مجموعه چندبعدی

transportation (products , cities) : numbers ;

دستور بالا ، مجموعه کالاهای حمل شده از هر نوع به هر شهر را با یک مشخصه تعریف می کند. آن یک مشخصه ، مربوط به تعداد کالای حمل شده است. همانطور که در بالا اشاره شد ، مشخصه های مجموعه می توانند پارامترهای ثابت و معلوم یا مقادیر متغیرهای مجهول مسأله باشند.

اگر همه اعضای حاصل از ضرب دکارتی مجموعه های یک بعدی ، در مجموعه چند بعدی نباشند ، کافی است که اعضای مجموعه چند بعدی را به طور مستقیم نشان دهیم. (این کار را با به کار بردن عبارات شرطی هم می توان انجام داد.) در مثال فوق ، اگر بخواهیم فقط کالاهای حمل شده به شهرها را در مورد دو نوع محصول اول تعریف کنیم ، باید نوشت :

transportation (products , cities) / 1,London 1,Paris 1,Newyork 1,Tehran 2,London 2,Paris 2,Newyork 2,Tehran / : numbers ;

لازم به ذکر است که در پایان تعریف هر مجموعه ، از علامت " **;** " استفاده می شود ، ولی بعد از دستور " *Endsets* " نیازی به استفاده از " **;** " نیست.

- **داده های ورودی به مدل :**

اگر بخواهیم داده هایی را به مدل بدهیم ، از دستور " *Data:* " به شکل زیر استفاده می کنیم. ورود داده ها به مدل با دستور " Enddata " پایان می یابد.

Data :

... ... ورود داده ها

...

Enddata

برای مثال اگر بخواهیم مقدار های مربوطه را برای جمعیت شهرها در مثال بالا بر حسب میلیون نفر به مدل بدهیم ، به این شکل عمل می کنیم :

population = 6 5.5 10 9.3 ;

عبارت فوق مثلا نشان می دهد که برای مثال ، جمعیت شهر " *Paris* " برابر ۵.۵ میلیون نفر است. ابعاد داده ها دقیقا باید با ابعاد مجموعه ها یکسان باشد ، برای مثال در مورد فوق چون چهار شهر داریم ، پس باید دقیقا چهار داده وارد کنیم. برای جدا کردن داده ها از هم می توان از فاصله خالی (مانند مثال فوق) یا از علامت " , " استفاده کرد.

برای ورود داده ها در مورد مجموعه های چندبعدی ، بهتر است که داده ها را به شکل سطری و ستونی وارد کنیم تا تشخیص آنها هم ساده تر باشد. مثلا اگر بخواهیم به مشخصه " *numbers* " که به مجموعه دو بعدی " *transportation* " متعلق است ، مقدار دهیم ، داریم : (داده ها بر حسب هزار تن در سال هستند.)

```
numbers = 10  14   20  13
          3   4    6   5
          20  25   18  24
          32  12   40  22
          4   3    5   2
          13  12   15   6
          54  43   68   71
          31  21   27  11
          7   4    8   10
          11  14   17  21 ;
```

همانطور که مشاهده می شود ، به تعداد اعضای مجموعه اول (محصولات) ، سطر و به تعداد اعضای مجموعه دوم (شهرها) ، ستون داریم. لازم به ذکر است که در پایان تعریف هر داده ، از علامت " **;** " استفاده می شود ، ولی بعد از دستور " *Enddata* " نیازی به استفاده از " **;** " نیست.

- **توابع :**

توابع *Lingo* ، دامنه کاربرد نرم افزار را دوچندان می کنند. استفاده از توابع از پیش تعیین شده *Lingo* ، امکانات قابل توجهی در اختیار ما قرار می دهد و ساخت انواع متنوع مدل را ممکن می سازد. کلیه توابع *Lingo* ، با علامت " *@* " آغاز می شوند و انواع زیر را شامل می شود : (در کلیه توابع زیر ، هر کجا که از علامت " *[]* " استفاده شده است به آن معنا است که آوردن عبارت درون " *[]* " برای تابع ، اختیاری است و می تواند در مواردی که لازم نیست ، حذف شود.)

- **توابع مالی (برای مطالعه):**

توابع مالی ، توابعی هستند که ارزش معادل جریان نقدی را در دوره های زمانی مختلف تعیین می کنند و در اقتصاد مهندسی کاربرد دارند و عبارتند از :

@FPA(I , N) : این تابع ، دو مقدار نرخ بهره (I) و تعداد پریودها (N) را به عنوان ورودی دریافت می کند و ارزش فعلی معادل جریان نقدی سالیانه یک واحد پول در هر سال به مدت N سال و با نرخ بهره I را به عنوان خروجی منعکس می کند. در صورتی که جریان نقدی سالیانه ، برابر یک واحد پول نباشد ، کافی است که خروجی تابع فوق را در مقدار مورد نظر ضرب کنیم.

@FPL(I , N) : این تابع ، دو مقدار نرخ بهره (I) و تعداد پریودها (N) را به عنوان ورودی دریافت می کند و ارزش فعلی معادل جریان نقدی یک واحد پول در N سال بعد را با نرخ بهره I ، به عنوان خروجی منعکس می کند.

- **توابع احتمالی (برای مطالعه):**

توابع احتمالی ، توابعی هستند که توزیع های مختلف احتمال را شامل می شوند و شامل انواع زیر هستند :

@PBN(P , N , X) : این تابع ، مقدار تابع تجمعی توزیع دوجمله ای را منعکس می کند. در واقع ، خروجی این تابع ، برابر احتمال آن است که در یک جامعه با جمعیت N که در آن نرخ خرابی ، برابر P است ، تعداد X یا کمتر ، قطعه معیوب وجود داشته باشد.

@PCX(N , X) : این تابع ، مقدار تابع تجمعی توزیع مربع کای را نشان می دهد. در واقع ، خروجی این تابع ، برابر احتمال آن است که یک مشاهده از توزیع مربع کای با N درجه آزادی ، مقداری کمتر یا مساوی X بگیرد.

@PEB(A , X) : این تابع ، مقدار تابع توزیع " *Erlang's Busy* " در یک سیستم با X خدمت دهنده و با بار ورودی A را نشان می دهد. در واقع مقدار A ، برابر تعداد متوسط مشتری ورودی به مدل در واحد زمان ، ضرب در زمان متوسط لازم برای خدمت دهی به هر مشتری است.

@PEL(A , X) : این تابع ، مقدار تابع توزیع " *Erlang's Loss* " در یک سیستم با X خدمت دهنده و با بار ورودی A را نشان می دهد. در واقع مقدار A ، برابر تعداد متوسط مشتری ورودی به مدل در واحد زمان ، ضرب در زمان متوسط لازم برای خدمت دهی به هر مشتری است.

@PFD(N , D ,X) : این تابع ، مقدار تابع تجمعی توزیع " F " را نشان می دهد. در واقع ، خروجی این تابع ، برابر احتمال آن است که یک مشاهده از توزیع " F " با درجات آزادی N و D ، مقداری کمتر یا مساوی X بگیرد.

@PFS(A , X , C) : این تابع ، تعداد متوسط مشتری های در انتظار تعمیر یا در حال تعمیر در یک سیستم خدمت دهی پواسون با X خدمت دهنده موازی ، C مشتری و بار محدود کننده A است. مقدار A ، در واقع برابر تعداد مشتری ها ضرب در متوسط زمان خدمت دهی تقسیم بر متوسط زمان تعمیر است.

@PHG(POP , G , N , X) : این تابع ، مقدار تابع تجمعی توزیع فوق هندسی را نشان می دهد. خروجی این تابع ، در واقع برابر احتمال آن است که در یک جامعه با جمعیت POP که G عضو آن سالم هستند ، تعداد X عضو یا کمتر از یک نمونه بدون جایگذاری با N عضو ، سالم باشند.

@PPL(A , X) : این تابع ، تابع فقدان خطی برای توزیع پواسون است. خروجی این تابع ، در واقع مقدار متوسط *Max (0 , Z-X)* را نشان می دهد که در آن Z ، دارای توزیع پواسون با پارامتر A می باشد.

@PPS(A , X) : این تابع ، مقدار تابع تجمعی توزیع پواسون را نشان می دهد. در واقع ، خروجی این تابع ، برابر احتمال آن است که یک مشاهده از توزیع پواسون با پارامتر A ، مقداری کمتر یا مساوی X بگیرد.

@PSL(X) : این تابع ، تابع فقدان خطی برای توزیع نرمال است. خروجی این تابع ، در واقع مقدار متوسط *Max (0 , Z-X)* را نشان می دهد که در آن Z ، دارای توزیع نرمال استاندارد است.

@PSN(X) : این تابع ، مقدار تابع تجمعی توزیع نرمال استاندارد را نشان می دهد. در واقع ، خروجی این تابع ، برابر احتمال آن است که یک مشاهده از توزیع نرمال استاندارد ، مقداری کمتر یا مساوی X بگیرد.

@PTD(N , X) : این تابع ، مقدار تابع تجمعی توزیع " t " را نشان می دهد. در واقع ، خروجی این تابع ، برابر احتمال آن است که یک مشاهده از توزیع " t " با N درجه آزادی ، مقداری کمتر یا مساوی X بگیرد.

@QRAND(SEED) : این تابع ، یک دنباله از اعداد شبه تصادفی (*Quasi Random*) از توزیع یکنواخت *(0,1)* را تولید می کند. اگر در این تابع ، ما مقدار *SEED* را مشخص نکنیم ، خود نرم افزار به طور خودکار ، از ساعت سیستم به عنوان *SEED* استفاده می کند.

@RAND(SEED) : این تابع ، یک مقدار شبه تصادفی (*Pseudo Random*) بین صفر و یک تولید می کند.

- **توابع مجموعه :**

توابع مجموعه ، در واقع مهمترین دسته توابع *Lingo* هستند و توابعی را شامل می شوند که روی مجموعه ها تعریف می شوند . این توابع عبارتند از :

@INDEX([set_name,] primitive_set_element) : این تابع ، اندیس متناظر با عضو داده شده به آن (*primitive_set_element*) را در مجموعه ای که نام آن را از ما دریافت می کند (*set_name*) ، نشان می دهد. دادن نام مجموعه به این تابع ، اختیاری است. اگر نام مجموعه را به تابع ندهیم ، تابع به طور خودکار ، اندیس اولین مجموعه ای که عضو یادشده در آن صدق می کند ، به عنوان خروجی به ما می دهد. برای مثال در مجموعه دستورات زیر ، مقدار M برابر ۱ و مقدار N برابر ۳ خواهد بود.

```
Sets :
S1 / A B C / : ;
S2 / X Y Z / : ;
Endsets
```

M = @INDEX (S1,A) ;

N = @INDEX (S2,Z) ;

@IN(set_name , primitive_index_1[,primitive_index_2...]) : این تابع ، در صورتی که اندیس متغیر یا متغیرهای داده شده به آن (*primitive_index*) ، در مجموعه نام برده شده (*set_name*) باشند ، مقدار " درست " و در غیر اینصورت مقدار " غلط " را نشان می دهد. در مجموعه دستورات زیر ، مقدار P برابر ۱ یا همان " درست " و مقدار Q برابر صفر یا " غلط " خواهد بود. چون عضو B,Y در مجموعه $S3$ وجود دارد ، ولی عضو C,Y ، در این مجموعه وجود ندارد.

Sets :

S1 / A B C / : ;

S2 / X Y Z / : ;

S3 (S1 , S2) / A,X A,Z B,Y C,X / : ;

Endsets

P = @IN (S3 , @INDEX(S1,B) , @INDEX(S2,Y)) ;

Q = @IN (S3 , @INDEX(S1,C) , @INDEX(S2,Y)) ;

@SIZE(set_name) : این تابع ، تعداد اعضای مجموعه با نام داده شده را نشان می دهد. استفاده از این تابع در مدل ، انعطاف پذیری مدل را در برابر تغییر اندازه مجموعه ها و کم یا زیاد کردن اعضای آنها ، بالا می برد.

@FOR(set_name[(set_index_list) [| cond_qualifier]]: exp_list) : این تابع ، دستورات موجود در *exp_list* را برای کلیه اعضای مجموعه *set_name* که در شرط *cond_qualifier* صدق می کنند ، اجرا می کند. *set_index_list* ، یک اندیس دلخواه برای اعضای مجموعه است که توسط خود ما انتخاب می شود. مجموعه دستورات زیر را در نظر می گیریم :

Sets :

final / 1..10 / : batch ;

Endsets

@FOR (final(i) | i#LE#5 : batch(i)=1000) ;

@FOR (final(i) | i#GT#5 : batch(i)=2000) ;

مطابق این دستورات ، مشخصه " *batch* " برای ۵ عضو اول مجموعه "*final*" ، برابر ۱۰۰۰ و برای ۵ عضو دیگر ، برابر ۲۰۰ خواهد بود.

@MAX(set_name[(set_index_list) [| cond_qualifier]]: expression) : این تابع ، حداکثر مقدار *expression* را روی مجموعه *set_name* که در شرط *cond_qualifier* صدق می کنند ، منعکس می کند. *set_index_list* ، یک اندیس دلخواه برای اعضای مجموعه است که توسط خود ما انتخاب می شود. این مطلب ، در مثال زیر تشریح شده است :

```
Sets :
vendors / 1..5 / : demand ;
Endsets
max_demand = @MAX (vendors (i) | i#LE#3 : demand(i)) ;
```

این دستور ، بزرگترین مقدار مشخصه " *demand* " بین ۳ عضو اول مجموعه " *vendors* " را در متغیر " *max_demand* " ذخیره می کند.

@MIN(set_name[(set_index_list) [| cond_qualifier]]: expression) : این تابع ، حداقل مقدار *expression* را روی مجموعه *set_name* که در شرط *cond_qualifier* صدق می کنند ، منعکس می کند. *set_index_list* ، یک اندیس دلخواه برای اعضای مجموعه است که توسط خود ما انتخاب می شود. . این مطلب ، در مثال زیر تشریح شده است :

```
Sets :
vendors / 1..5 / : demand ;
Endsets
min_demand = @MIN (vendors (i) | i#LE#3 : demand(i)) ;
```

این دستور ، کوچکترین مقدار مشخصه " *demand* " بین ۳ عضو اول مجموعه " *vendors* " را در متغیر " *min_demand* " ذخیره می کند.

@SUM(set_name[(set_index_list) [| cond_qualifier]]:expression) : این تابع ، مجموع مقادیر *expression* را روی مجموعه *set_name* که در شرط *cond_qualifier* صدق می کنند ، منعکس می کند. *set_index_list* ، یک اندیس دلخواه برای اعضای مجموعه است که توسط خود ما انتخاب می شود. در ادامه مثال فوق ، دستور زیر ، مجموع مشخصه " *demand* " برای ۳ عضو آخر مجموعه " *vendors* " را در متغیر *K* ذخیره می کند.

```
K = @SUM (vendors(i) | i#GT#2 : demand(i)) ;
```

- **توابع ریاضی (برای مطالعه):**

توابع ریاضی در واقع یک یا چند ورودی را دریافت کرده و یک خروجی را تعیین می کنند و انواع زیر را شامل می شوند :

@ABS(X) : این تابع ، مقدار قدر مطلق X را نشان می دهد.

@COS(X) : این تابع ، مقدار کسینوس X را نشان می دهد که در آن ، X یک زاویه بر حسب رادیان است.

@EXP(X) : این تابع ، عدد نپر ($e=2.718281\ldots$) را به توان X رسانده و مقدار آن را نشان می دهد.

@FLOOR(X) : این تابع ، مقدار جزء صحیح عدد X را نشان می دهد.

@LGM(X) : این تابع ، مقدار لگاریتم طبیعی تابع گامای X را نشان می دهد.

@LOG(X) : این تابع ، مقدار لگاریتم طبیعی عدد X را نشان می دهد.

@SIGN(X) : این تابع ، علامت عدد X را نشان می دهد. اگر X عددی مثبت باشد ، تابع مقدار یک و اگر X عددی منفی باشد ، تابع مقدار منفی یک را منعکس می کند. اگر X برابر صفر باشد ، مقدار تابع هم برابر صفر خواهد شد.

@SIN(X) : این تابع ، مقدار سینوس X را نشان می دهد که در آن ، X یک زاویه بر حسب رادیان است.

@SMAX($X_1,X_2,\ldots,X_N$) : این تابع ، بزرگترین عدد از بین اعداد داده شده به آن را نشان می دهد.

@SMIN($X_1,X_2,\ldots,X_N$) : این تابع ، کوچکترین عدد از بین اعداد داده شده به آن را نشان می دهد.

@TAN(X) : این تابع ، مقدار تانژانت X را نشان می دهد که در آن ، X یک زاویه بر حسب رادیان است.

- **توابع قلمرو متغیر :**

توابع قلمرو متغیر ، نوع متغیرها را مشخص می کنند. متغیرها در *Lingo* ، به صورت پیش فرض ، پیوسته و بزرگتر مساوی صفر تعریف می شوند. در صورتی که متغیری ، شرایط دیگری داشته باشد ، از یکی از چهار تابع زیر استفاده می کنیم :

@BIN(X) : این تابع ، متغیر X را به یک متغیر صفر و یک تبدیل می کند و در مدلهای برنامه ریزی عدد صحیح کاربرد دارد.

@BND(lower_bound , X , upper_bound) : این تابع ، مقدار متغیر X را به مقادیر بزرگتر از *lower_bound* و کوچکتر از *upper_bound* محدود می کند و در واقع حد بالا و حد پایین برای متغیرها تعیین می کند.

@FREE(X) : این تابع ، به متغیر X اجازه می دهد که مقادیر منفی هم اختیار کند و در واقع ، متغیر را به یک متغیر آزاد در علامت تبدیل می کند.

@GIN(X) : این تابع ، به متغیر X فقط اجازه می دهد که مقادیر صحیح بگیرد و در مدلهای برنامه ریزی عدد صحیح کاربرد دارد.

➢ **توابع خارجی (برای مطالعه):**

توابع خارجی در واقع ، امکان برقراری ارتباط *Lingo* را با سایر نرم افزارها و فایلها از قبیل فایلهای متنی ، صفحه گسترده ها ، بانکهای اطلاعاتی و ... فراهم نموده و یک سری امکانات اضافی را به شرح زیر در اختیار ما قرار می دهند :

@DUAL(variable_or_row_name) : این تابع ، مقادیر دوگان متناظر با متغیر یا سطر داده شده را به عنوان خروجی به ما می دهد.

@FILE('filename') : این تابع ، به ما اجازه می دهد که داده ها را در هر جای مدل که بخواهیم ، از فایلهای متنی فراخوانی کنیم . *Lingo* به خواندن داده از فایل مورد نظر تا زمان رسیدن به انتهای فایل یا برخورد به علامت " ~ " ادامه می دهد.

@ODBC(['data_source'[,'table_name'[,'col_1'[,'col_2'...]]]]) : این تابع ، برای گرفتن اعضای مجموعه یا مقادیر داده ها از بانکهای اطلاعاتی کاربرد دارد. *data_source* نام فایل مورد نظر ، *table_name* نام جدول شامل داده ها در بانک اطلاعاتی و *col_1* ، نام ستون (یا ستونهایی) است که داده های مورد نظر ، در آن قرار دارند.

@OLE('spreadsheet_file'[, range_name_list]) : این تابع ، برای تبادل اطلاعات بین *Lingo* و نرم افزارهای صفحه گسترده مانند *Excel* به کار می رود. *spreadsheet_file* ، نام فایل مورد نظر و *range_name_list* ، نام دامنه یا محدوده ای از داده ها در آن صفحه است که قرار است مورد استفاده قرار گیرد.

@POINTER(N) : این تابع ، برای استفاده از کتابخانه پویای *Lingo* ، (*Dynamic Link Library*) است و امکان استفاده مستقیم از این اطلاعات را فراهم می کند.

@RANGED(variable_or_row_name) : این تابع ، مقدار کاهش مجاز در ضریب یک متغیر در تابع هدف یا مقدار کاهش مجاز در پارامتر سمت راست یک سطر را نشان می دهد.

@RANGEU(variable_or_row_name) : این تابع ، مقدار افزایش مجاز در ضریب یک متغیر در تابع هدف یا مقدار افزایش مجاز در پارامتر سمت راست یک سطر را نشان می دهد.

@STATUS() : این تابع ، وضعیت نهایی فرآیند حل یک مدل را به صورت کدهای زیر نشان می دهد :

کد تابع *@STATUS()*	تفسیر
0	بهینه سراسری : حل بهینه مدل یافته شده است.
1	ناموجه : هیچ نقطه شدنی برای مدل با توجه به محدودیت ها وجود ندارد.

2	نامحدود : تابع هدف می تواند بدون هیچ محدودیتی اضافه شود.
3	نامشخص : فرآیند حل ، بی نتیجه مانده است.
4	----------
5	ناموجه یا نامحدود : پیش پردازنده ، مدل را ناموجه یا نامحدود تشخیص داده است.
6	بهینه محلی : جواب بهینه محلی یافته شده است ، اما ممکن است جواب بهتریموجود باشد.
7	ناموجه محلی : اگرچه ممکن است نقطه شدنی موجود باشد ، اما *Lingo* قادر به یافتن آن نیست.
8	توقف : دستیابی به سطح مورد نظر برای تابع هدف ، انجام شده است.
9	خطای عددی : فرآیند حل ، به خاطر وجود یک عبارت تعریف نشده در یکی ازمحدودیتها ، متوقف شده است.

@TEXT([‘filename’]) : این تابع ، برای انتقال نتایج حل یک مدل به فایلهای متنی به کار می رود. *filename*، نام فایلی است که می خواهیم نتایج در آن نوشته شود.

- **سایر توابع (فقط تابع If جزء امتحان است):**

@WRAP(INDEX , LIMIT) : این تابع ، اندیس متغیر را طوری تغییر می دهد که در محدوده مجاز و تعریف شده برای مجموعه قرار گیرد. در مدلهای برنامه ریزی چند دوره ای ، مواقعی اتفاق می افتد که ما می خواهیم به کمک توابع مجموعه ها ، یک محدودیت را برای یک سری متغیر تعمیم دهیم. اما با توجه به ویژگی سیکلی این مسائل ، این موضوع ممکن است باعث شود که اندیس متغیرها از محدوده تعریف شده در ابتدا یا انتهای مجموعه خارج شود. تابع فوق ، به مقدار *INDEX* ، مضرب صحیحی از *LIMIT* را طوری اضافه می کند که عدد حاصل ، در محدوده بین صفر و *LIMIT* قرار گیرد.

@IF(logical_condition , true_result , false_result) : این تابع ، ابتدا شرط داده شده (*logical_condition*) را ارزیابی می کند. اگر عبارت شرطی ، درست بود ، مقدار *true_result* و اگر غلط بود ، مقدار

false_result را به عنوان خروجی می دهد. برای مثال ، محدودیت زیر ، در صورتی که " *Y* " بزرگتر از ۰ باشد، مقدار " *setup_cost* " را برابر ۵۰۰ و در غیر اینصورت برابر صفر قرار می دهد.

```
setup_cost = @IF( Y#GT#0 , 500 , 0) ;
```

@WARN('text',logical_condition) : این تابع ، در صورت برقرار بودن شرط مطرح شده (*logical_condition*) ، متن داده شده به آن را به صورت یک پیغام هشدار ، نمایش می دهد.

@USER(user_determined_arguments) : این تابع هم ، برای استفاده از کتابخانه پویای *Lingo* ، (*Dynamic Link Library*) است و امکان استفاده مستقیم از این اطلاعات را فراهم می کند.

@WKX('input_worksheet' , ['output_worksheet',] 'range') : این تابع هم ، برای برقراری ارتباط *Lingo* با نرم افزارهای صفحه گسترده به کار می رود.

مثال: کوله پشتی (Knapsack problem)

```
MODEL:

SETS:
   ITEMS / ANT_REPEL, BEER, BLANKET,
    BRATWURST, BROWNIES, FRISBEE, SALAD,
     WATERMELON/:
      INCLUDE, WEIGHT, RATING;
ENDSETS

DATA:
   WEIGHT= 1 3 4 3 3 1 5 10;
   RATING =2 9 3 8 10 6 4 10;
   KNAPSACK_CAPACITY = 15;
ENDDATA

MAX = @SUM( ITEMS: RATING * INCLUDE);

@SUM( ITEMS: WEIGHT * INCLUDE) <=
 KNAPSACK_CAPACITY;

@FOR( ITEMS: @BIN( INCLUDE));

END
```

پس از حل:

```
Global optimal solution found at iteration:           0
  Objective value:                                38.00000


                     Variable           Value        Reduced Cost
            KNAPSACK_CAPACITY        15.00000            0.000000
           INCLUDE( ANT_REPEL)        1.000000           -2.000000
                INCLUDE( BEER)        1.000000           -9.000000
             INCLUDE( BLANKET)        1.000000           -3.000000
```

```
            INCLUDE( BRATWURST)        1.000000           -8.000000
             INCLUDE( BROWNIES)        1.000000           -10.00000
              INCLUDE( FRISBEE)        1.000000           -6.000000
                INCLUDE( SALAD)        0.000000           -4.000000
           INCLUDE( WATERMELON)        0.000000           -10.00000
             WEIGHT( ANT_REPEL)        1.000000            0.000000
                  WEIGHT( BEER)        3.000000            0.000000
               WEIGHT( BLANKET)        4.000000            0.000000
             WEIGHT( BRATWURST)        3.000000            0.000000
              WEIGHT( BROWNIES)        3.000000            0.000000
               WEIGHT( FRISBEE)        1.000000            0.000000
                 WEIGHT( SALAD)        5.000000            0.000000
            WEIGHT( WATERMELON)        10.00000            0.000000
             RATING( ANT_REPEL)        2.000000            0.000000
                  RATING( BEER)        9.000000            0.000000
               RATING( BLANKET)        3.000000            0.000000
             RATING( BRATWURST)        8.000000            0.000000
              RATING( BROWNIES)        10.00000            0.000000
               RATING( FRISBEE)        6.000000            0.000000
                 RATING( SALAD)        4.000000            0.000000
            RATING( WATERMELON)        10.00000            0.000000

                          Row    Slack or Surplus      Dual Price
                            1        38.00000            1.000000
                            2        0.000000            0.000000
```

مثال) مسئله مکان‌یابی محل احداث کارخانه ها و نحوه پوشش تقاضای مشتریان

```
MODEL:
 ! Capacitated Plant Location Problem;
 SETS:
    PLANTS / P1, P2, P3/: FCOST, CAP, OPEN;
    CUSTOMERS / C1, C2, C3, C4/ : DEM;
    ARCS( PLANTS, CUSTOMERS)  : COST, VOL;
 ENDSETS

 DATA:
  ! Fixed cost of opening at each origin;
       FCOST = 91, 70, 24;
  ! Capacities at each origin;
       CAP =   39, 35, 31;
  ! Demands at each destination;
       DEM =   15, 17, 22, 12;
  ! The cost/unit shipment matrix;
       COST =   6,  2,  6,  7,
                4,  9,  5,  3,
                8,  8,  1,  5;
 ENDDATA

 ! The objective;
    [TTL_COST] MIN = @SUM( ARCS: COST * VOL) +
     @SUM( PLANTS: FCOST * OPEN);

 ! The demand constraints;
    @FOR( CUSTOMERS( J): [DEMAND]
```

```
     @SUM( PLANTS( I): VOL( I, J)) >= DEM( J)
    );

 ! The supply constraints;
    @FOR( PLANTS( I): [SUPPLY]
     @SUM( CUSTOMERS( J): VOL( I, J)) <=
      CAP( I) * OPEN( I)
    );

 ! Make OPEN binary(0/1);
    @FOR( PLANTS: @BIN( OPEN));
 END
```

پس از حل:

```
Global optimal solution found at iteration:           24
  Objective value:                                327.0000


                     Variable           Value        Reduced Cost
                   FCOST( P1)        91.00000            0.000000
                   FCOST( P2)        70.00000            0.000000
                   FCOST( P3)        24.00000            0.000000
                     CAP( P1)        39.00000            0.000000
                     CAP( P2)        35.00000            0.000000
                     CAP( P3)        31.00000            0.000000
                    OPEN( P1)        1.000000            91.00000
                    OPEN( P2)        0.000000           -70.00000
                    OPEN( P3)        1.000000           -38.00000
                     DEM( C1)        15.00000            0.000000
                     DEM( C2)        17.00000            0.000000
                     DEM( C3)        22.00000            0.000000
                     DEM( C4)        12.00000            0.000000
                COST( P1, C1)        6.000000            0.000000
                COST( P1, C2)        2.000000            0.000000
                COST( P1, C3)        6.000000            0.000000
                COST( P1, C4)        7.000000            0.000000
                COST( P2, C1)        4.000000            0.000000
                COST( P2, C2)        9.000000            0.000000
                COST( P2, C3)        5.000000            0.000000
                COST( P2, C4)        3.000000            0.000000
                COST( P3, C1)        8.000000            0.000000
                COST( P3, C2)        8.000000            0.000000
                COST( P3, C3)        1.000000            0.000000
                COST( P3, C4)        5.000000            0.000000
                 VOL( P1, C1)        15.00000            0.000000
                 VOL( P1, C2)        17.00000            0.000000
                 VOL( P1, C3)        0.000000            3.000000
                 VOL( P1, C4)        3.000000            0.000000
                 VOL( P2, C1)        0.000000            2.000000
                 VOL( P2, C2)        0.000000            11.00000
                 VOL( P2, C3)        0.000000            6.000000
                 VOL( P2, C4)        0.000000            0.000000
                 VOL( P3, C1)        0.000000            4.000000
                 VOL( P3, C2)        0.000000            8.000000
```

```
     VOL( P3, C3)        22.00000          0.000000
     VOL( P3, C4)        9.000000          0.000000

              Row    Slack or Surplus      Dual Price
        TTL_COST        327.0000         -1.000000
      DEMAND( C1)       0.000000         -6.000000
      DEMAND( C2)       0.000000         -2.000000
      DEMAND( C3)       0.000000         -3.000000
      DEMAND( C4)       0.000000         -7.000000
      SUPPLY( P1)       4.000000          0.000000
      SUPPLY( P2)       0.000000          4.000000
      SUPPLY( P3)       0.000000          2.000000
```

سایر ابزارهای نرم افزار *Lingo* :

- **چک کردن و تصحیح پرانتزها :**

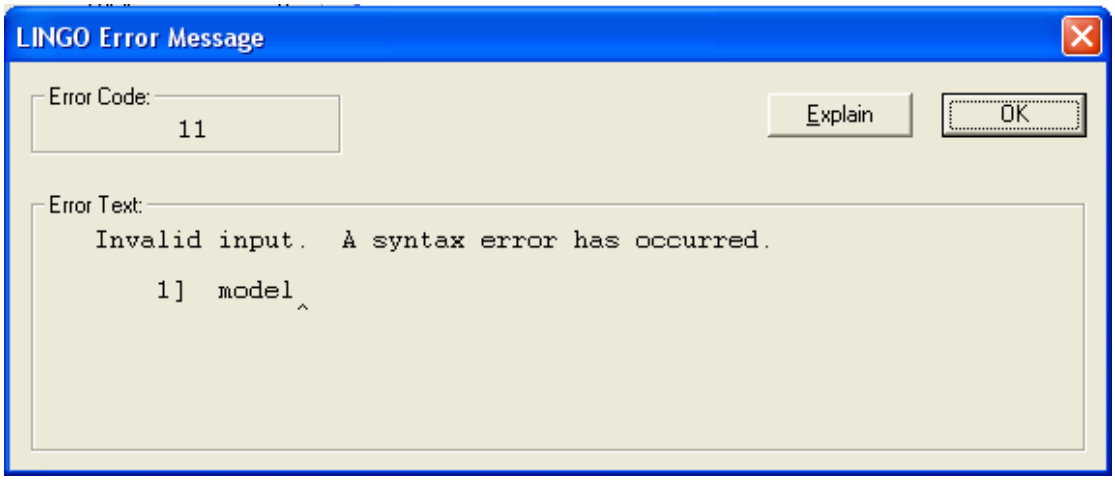
گاهی در عبارتهای طولانی و پیچیده ریاضی ، زیادی تعداد پرانتزها و عملگرهای محاسباتی ، ممکن است تا حدودی گیج کننده باشد به طوری تشخیص اینکه هر پرانتز دقیقا در چه محلی بسته یا باز می شود ، کار مشکلی به نظر می رسد. در صورتی که بخواهیم تعیین کنیم که یک پرانتز ، دقیقا در چه محلی بسته یا باز می شود ، کافی است که آن پرانتز را انتخاب کرده و سپس از منوی " *Edit* " ، گزینه " *Match Parenthesis* " یا از نوار ابزار ، آیکن مربوطه را انتخاب کنیم. در این حالت ، نرم افزار ، به طور خودکار ، پرانتز متناظر با پرانتز انتخاب شده را نشان می دهد.

- **حل کردن مدل و دریافت نتایج :**

پس از نوشتن مدل ، کافی است که از منوی " *LINGO* " ، گزینه " *Solve* " یا از نوار ابزار ، آیکن " *Solve* " را انتخاب کنیم. در این حالت ، در صورتی که مدل نوشته شده ، خطایی نداشته باشد ، حل مدل آغاز شده و جواب نهایی نمایش داده می شود.

- **دریافت خطا :**

زمانی که مدل را حل می کنیم ، ممکن است که مدل دارای خطا باشد. در این حالت ، پیغام خطای مربوطه که در آن شماره خطا و توضیح مختصری درباره آن نوشته شده است ، نمایش داده می شود. در صورتی که بخواهیم در مورد خطا ، توضیحات بیشتری ببینیم ، گزینه " *Explain* " را از این پیغام انتخاب می کنیم تا توضیحات بیشتر ظاهر شود.



شکل ۲ : پیغام خطا

- **پنجره وضعیت حل :**

پس از رفع خطاهای موجود و انتخاب گزینه " *Solve* " ، پنجره ای ظاهر می شود که وضعیت حل مدل در آن درحال نمایش است و اطلاعات زیر در این پنجره مطرح شده است :

نوع مدل از نظر برنامه ریزی خطی ، غیر خطی ، عدد صحیح و ....

وضعیت نهایی از نظر جواب بهینه سراسری ، بهینه موضعی و .....

مقدار تابع هدف

ناموجه بودن (در صورتی که جواب ، در همه محدودیتها صدق کند ، عدد صفر نمایش داده می شود.)

تعداد تکرارها تا رسیدن به جواب

اطلاعاتی در مورد نوع *Solver* به کار گرفته شده توسط *Lingo* و سایر مطالب در مورد آن

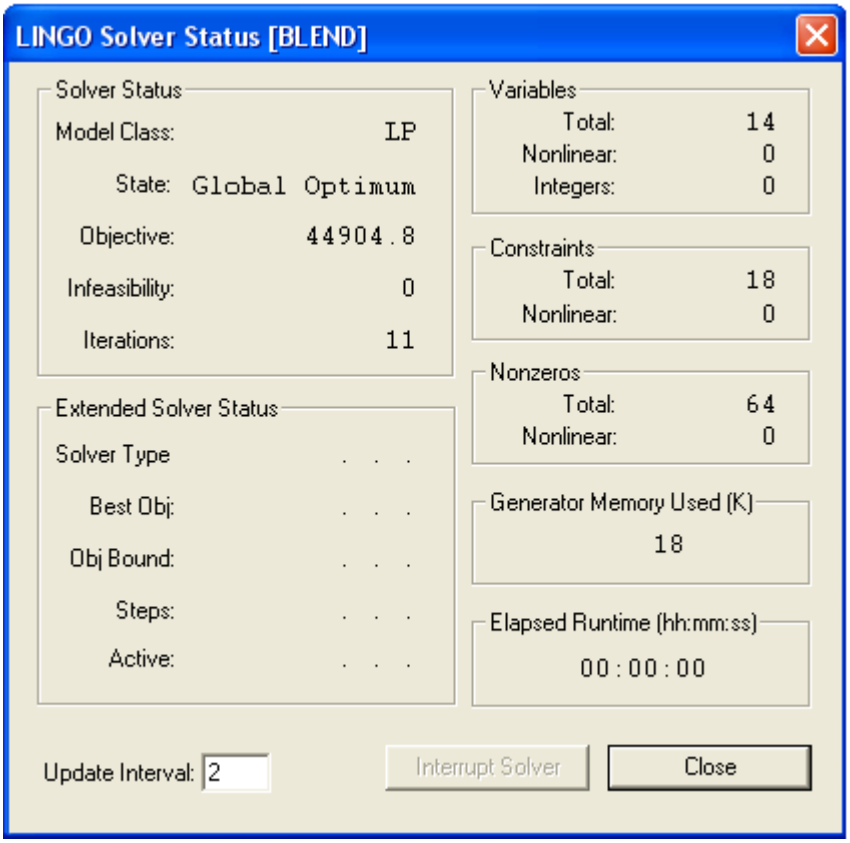
تعداد کل متغیرها ، متغیرهای عدد صحیح و متغیرهای غیر خطی

تعداد کل محدودیتها و محدودیتهای غیر خطی

تعداد ضرایب غیرصفر و ضرایب متناظر با محدودیتهای غیر خطی

کل حافظه استفاده شده بر حسب کیلوبایت

زمانی که طول کشیده تا به جواب رسیده ایم.



شکل ۳ : پنجره وضعیت حل

- **پنجره نهایی حل :**

علاوه بر پنجره وضعیت حل ، پس از حل کامل مدل ، پنجره نهایی حل که شامل کلیه جوابهاست ، ظاهر می شود. در سطر اول این پنجره ، تعداد تکرارهای لازم تا یافتن جواب نهایی و در سطر دوم ، مقدار نهایی تابع هدف نشان داده شده است. در ادامه این صفحه ، کلیه مقادیر اعم از پارامترهای معلوم و مقادیر متغیرهای مجهول به همراه نرخ کاهش آنها ، آورده شده است. منظور از نرخ کاهش هر متغیر ، جریمه ای است که ما برای ورود هر یک واحد از آن متغیر به درون جواب باید بپردازیم. در ادامه صفحه حل نهایی مدل هم ، نام هر یک از سطرها به همراه مقدار متغیر کمکی متناظر با آن سطر و متغیر دوگان متناظر با آن سطر ، آورده شده است. با انتخاب گزینه "*Save As ...*" از منوی "*File*" می توان این خروجی را به صورت یک فایل جداگانه ذخیره کرد.

- **ذخیره کردن ، باز کردن و پرینت گرفتن مدل :**

مشابه سایر نرم افزارهای تحت ویندوز ، در منوی "*File*" نرم افزار *Lingo* ، گزینه هایی برای ساخت یک مدل جدید(*New*) ، باز کردن یک مدل (*Open*) ، ذخیره (*Save*) و بستن مدل (*Close*) وجود دارد.

مدل های ساخته شده در *Lingo* ، در فایلهایی با پسوند " *lg4* " ، نتایج و گزارشهای حل در فایلهایی با پسوند " *lgr* " ، فایلهای متنی مدل های *Lingo* ، در فایلهایی با پسوند " *lng* " و داده های *Lingo* ، در فایلهایی باپسوند " *ldt* " ذخیره می شوند.

- **حداکثر ابعاد مدل و *Solver* های مورد استفاده :**

هریک از نسخه های نرم افزار *Lingo* ، دارای حداکثرهایی از نظر تعداد محدودیتها و تعداد انواع متغیرهای مدل هستند. این حداکثرها ، در جدول زیر آورده شده است :

نسخه ***Lingo***	کل متغیرها	متغیرهای عدد صحیح	متغیرهای غیرخطی	محدودیتها
Demo/Web	*300*	*30*	*30*	*150*
Solver Suite	*500*	*50*	*50*	*250*
Super	*2,000*	*200*	*200*	*1,000*
Hyper	*8,000*	*800*	*800*	*4,000*
Industrial	*32,000*	*3,200*	*3,200*	*16,000*
Extended	نامحدود	نامحدود	نامحدود	نامحدود

برای پی بردن به این محدودیتها در مورد نسخه ای که در دست داریم، کافی است که از منوی"*Help* "، گزینه " *About Lingo* " را انتخاب کنیم تا این موارد را به تفصیل ببینیم. در انتهای صفحه ظاهر شده ، لیست *Solver* هایی که *Lingo* از آنها استفاده می کند ، آورده شده است. هر *Solver* ، دارای قابلیتهای خاصی است و برای حل رده های خاصی از مسائل مفید است. تعدادی از این *Solver* ها عبارتند از :

Primal Simplex
Dual Simplex
Branch & Bound
Nonlinear
Barrier
Quadratic
Global

Multistart

شکل ۴ : صفحه خصوصیات نرم افزار

سایر گزینه های منوی " *Help* " هم ، توضیحات کاملی در مورد کلیه سؤالاتی که ممکن است برای کاربر ایجاد شود ، ارائه می کند.

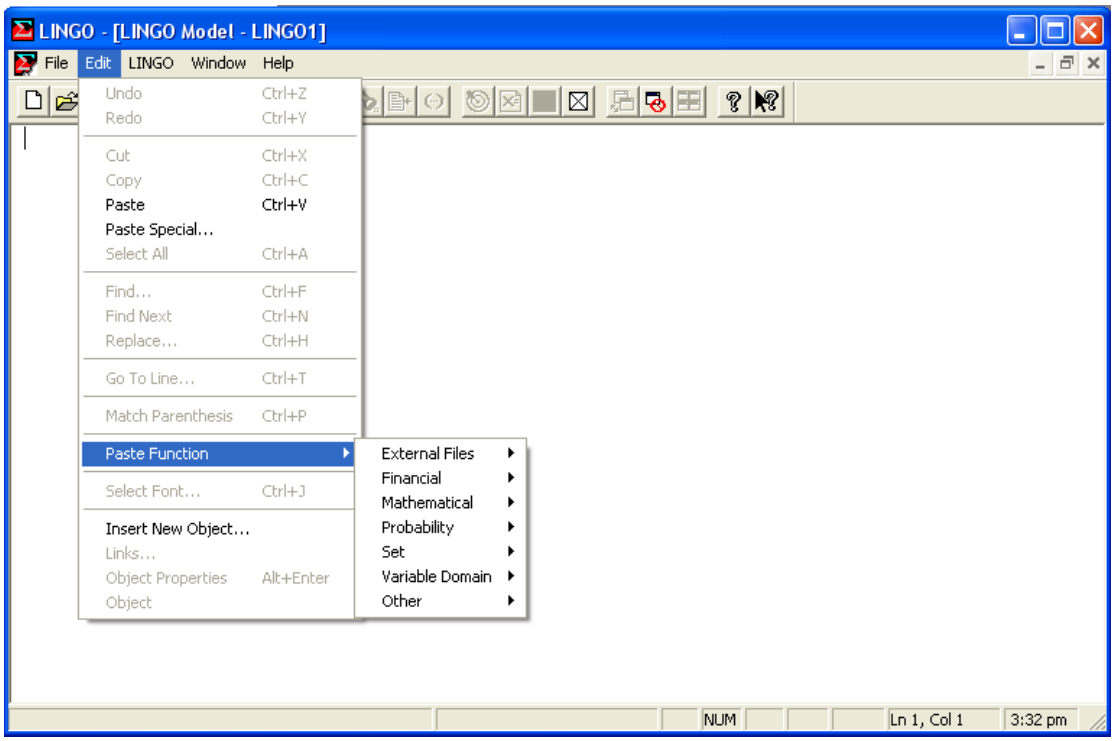
- **آدرس اینترنتی :**

نرم افزار *Lingo* ، ساخته شرکت *Lindo System* می باشد. آدرس اینترنتی این شرکت ، www.lindo.com است که نسخه " *Demo* " نرم افزار به صورت رایگان از این سایت ، قابل دریافت است. سایر نسخه های نرم افزار و راهنمای کار با نرم افزار هم از طریق این سایت ، قابل خریداری است.

تا آخر جزوه برای مطالعه

- **نحوه استفاده از توابع :**

گزینه " *Paste Function* " از منوی" *Edit* " ، کلیه توابع نرم افزار را در اختیار ما قرار می دهد. لذا با کمک این گزینه ، می توان به جای نوشتن نام تابع ، آن را از این طریق انتخاب کرد تا خود نرم افزار ، تابع را در آن محل قرا دهد.



شکل ۵ : نحوه استفاده از توابع

- **تحلیل حساسیت :**

پس از حل مدل ، با انتخاب گزینه " *Range* " از منوی " *Lingo* " می توان تحلیل حساسیت را روی مدل انجام داد. در صفحه ای که ظاهر خواهد شد ، تحلیل حساسیت روی کلیه ضرایب تابع هدف و کلیه مقادیر سمت راست انجام شده است ، به این

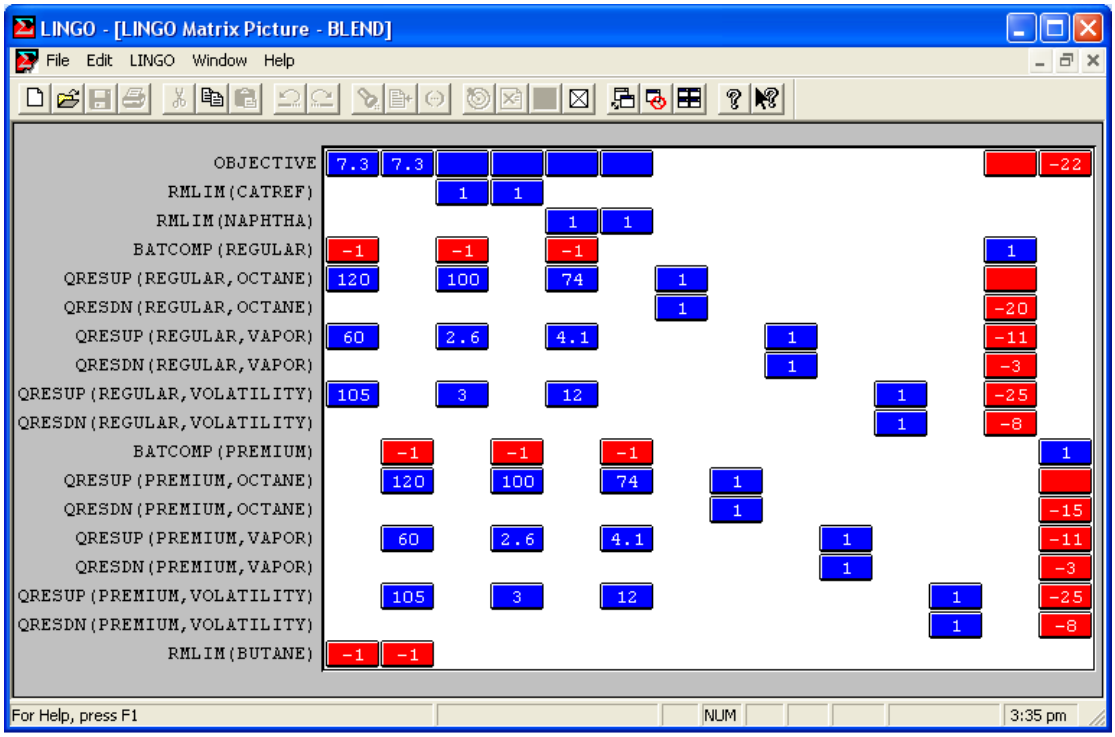
ترتیب که میزان مجاز افزایش و کاهش در مقدار ضرایب تابع هدف و مقادیر سمت راست طوری که جواب بهینه فعلی ، همچنان بهینه باقی بماند ، نمایش داده می شود.

- **نمایش حالت استاندارد مدل :**

برای هر مدل نوشته شده به وسیله *Lingo* ، می توان مدل استاندارد معادل آن را تولید کرد. کاربرد این ابزار زمانی است که فهم مدل نوشته شده با نرم افزار ، برای کاربر مشکل است. در این حالت ، می توان با مشاهده مدل استاندارد معادل ، به راحتی ابهامات موجود را رفع و مدل را تحلیل کرد. برای این کار ، ابتدا گزینه " *Generate* " را از منوی " *Lingo* " برگزیده و " *Display Model* " را انتخاب می کنیم.

- **نمایش فرم ماتریسی مدل :**

برای هر مدل نوشته شده به وسیله *Lingo* ، می توان مدل ماتریسی معادل آن را هم تولید کرد. کاربرد این ابزار زمانی است که فهم مدل نوشته شده با نرم افزار ، برای کاربر مشکل است. در این حالت ، می توان با مشاهده مدل ماتریسی معادل ، به راحتی ابهامات موجود را رفع و مدل را تحلیل کرد. برای این کار ، گزینه " *Picture* " را از منوی " *Lingo* " انتخاب می کنیم. در شکل ظاهر شده ، با کلیک راست ماوس ، می توان نام متغیرها و نام محدودیتها را هم دید و به کمک " *Zoom In* " و " *Zoom Out* " ، روی ضرایب مورد نظر تمرکز کرد. با دابل کلیک روی یک ضریب ، می توان نام متغیر متناظر با آن را در بالای صفحه و سطر (محدودیت) متناظر با آن را در سمت چپ صفحه دید.



شکل ۶ : فرم ماتریسی مدل

- **نمایش همزمان پنجره ها و حرکت بین پنجره ها :**

منوی " *Window* " ، هم امکاناتی از قبیل نمایش همزمان چند پنجره باز و امکان حرکت بین پنجره ها را برای ما فراهم می کند. این ابزار خصوصا زمانی که چندین مدل را به صورت همزمان باز کرده ایم ، کابرد زیادی دارد.

- **گرفتن خروجی دلخواه از مدل :**

Lingo ، این قابلیت را دارد که انواع مختلفی گزارش با توجه به انتخاب کاربر ، در اختیار او قرار دهد. با انتخاب گزینه " *Solution...* " از منوی " *Lingo* " ، محدودیت یا مشخصه ای که می خواهیم در مورد آن گزارش بگیریم را در سمت چپ صفحه انتخاب کرده و در سمت راست هم نوع گزارش که متن یا نمودار باشد ، را تعیین می کنیم. در پایین صفحه هم نوع نمودار ، حدبالا و حدپایین آن و اینکه مقادیر اولیه باشد یا دوگان ، قابل تعیین است. به این ترتیب ، می توان گزارش لازم را دریافت کرد.

- **مشاهده یک قسمت خاص از مدل :**

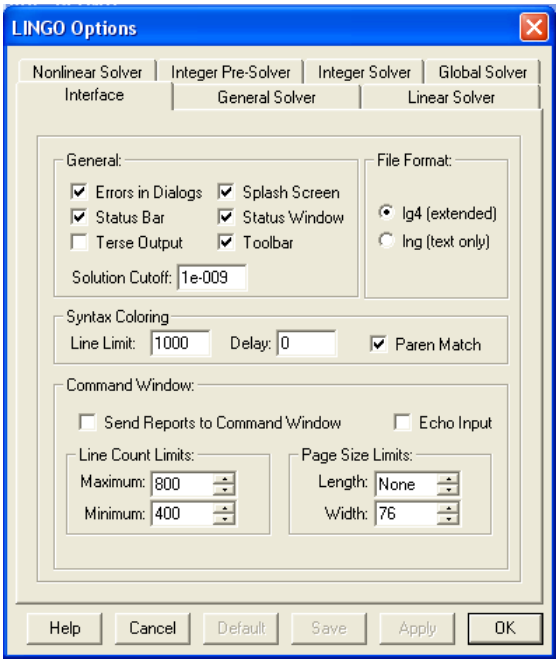
در مدل های پیچیده و طولانی ، گاهی بهتر است که همه مدل نمایش داده نشود ، بلکه فقط قسمتی از آن که در حال حاضر ما با آن کار می کنیم ، در پنجره ظاهر شود. برای این کار ، کافی است از منوی " *Lingo* " گزینه " *Look...* " را انتخاب کرده و سطرهایی که می خواهیم دیده شوند ، در آنجا تعیین کنیم.

- **آمار مدل :**

با انتخاب "*Model Statistics* " از منوی "*Lingo* " ، آمار کلی مدل شامل تعداد انواع متغیرها ، تعداد انواع محدودیتها ، تعداد ضرایب غیرصفر ، کوچکترین و بزرگترین عدد در مدل و نوع تابع هدف از نظر حداکثر یا حداقل سازی نشان داده می شود.

- **سایر تنظیمات :**

با انتخاب گزینه " *Options...* " از منوی " *LINGO* " ، می توان به سایر تنظیمات نرم افزار دست یافت. این گزینه مطابق شکل ، شامل ۷ برگه *(Tab)* در بالای صفحه می باشد که هریک شامل تنظیمات و امکاناتی مرتبط با برگه انتخاب شده هستند.



شکل ۷ : صفحه تنظیمات نرم افزار