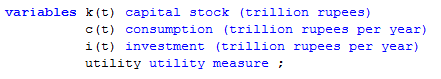
**Learning note**

**Chapter7 Variables**

**1.The syntax**

(1)example





var-type is the optional variable type;

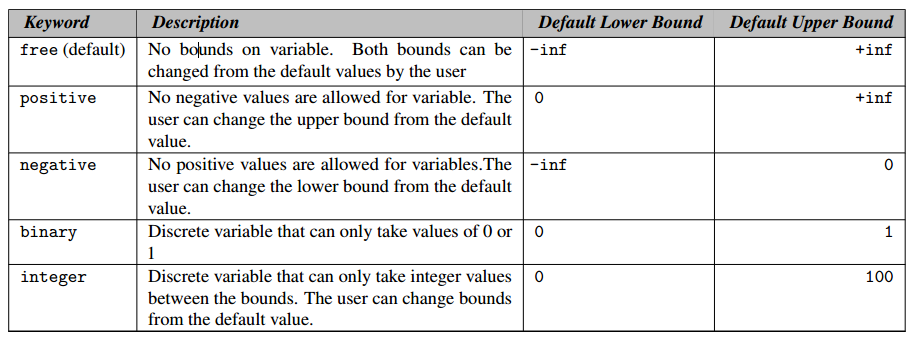
var\_name is the internal name of the variable, or identifier, start with a letter followed by more letters or digits.

text must be on the same line with the identifier. slash / is illegal in unquoted text.

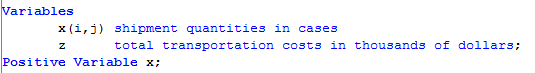
several variables can be declared in one statement.

attention ：two variables cannot appear on the same line.

**2.Variable types**



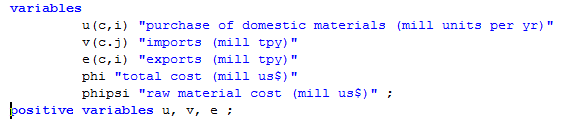
example



**3. Styles for variable declaration**

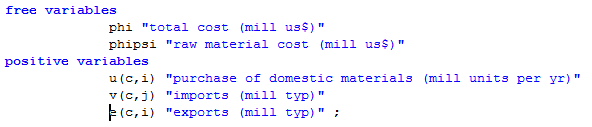
two styles are commonly used to declare variable types.

the first:



list all variables with domain specifications and explanatory text as a group, and later to group them separately as to type.

the second:



list them in groups by type.

attention: one identifier can be declared more than once, but does not contradict with each other, the second and subsequent should only add new information.

**4.Variable attributes**

variable has seven attributes, and they are as follows:

|  |  |
| --- | --- |
| attributes | description |
| .lo | The lower bound |
| .up | The upper bound |
| .fx | The fixed value |
| .l | The level value |
| .m | The marginal value |
| .scale | The scaling factor on the variable |
| .prior | The branching priority value. |

default bound can be changed using assignment statements.

for binary and integer variables, the consequences of the type declaration can note be completely undo.

fixing variables(.fx) is equivalent to the lower bound and upper bound being equal to the fixed value. fixed variables can subsequently be freed by changing the lower and upper bounds.

**5.assigning values to variable attributes**

examples:

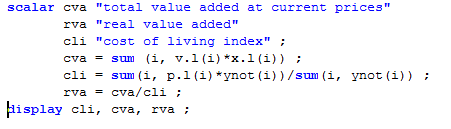
****

the order is important. the two pairs of statements below produce different values. the first lower bound is 0.01, the second is 1.

****

**6.variable attributes in assignments**

example



**7.Displaying variable attributes**

you must specify the variable attributes in displaying statements.

example:



**Chapter 8 Equations**

**1. equation declarations**

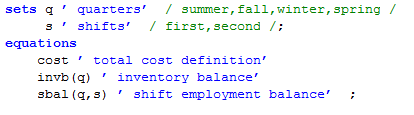
the syntax is as follows.



the equ\_name is an identifier has to start with letter followed by letters or digits.

there are no modifying keywords and no initializing data.

example:



the cost above is a scalar equation, which will produce at most one equation. the sbal is declared over the sets q(4 members) and s(2 members), which produce at most eight equations.

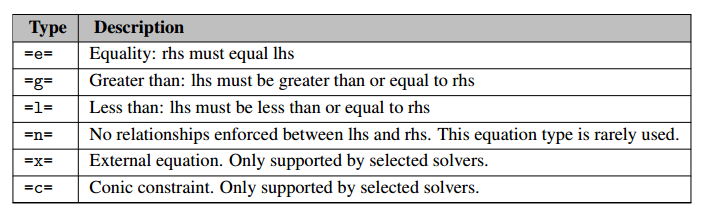
**2.equation definitions**

the syntax is as follows:  


eqn\_name is the name of the equation as in the equation declarations.

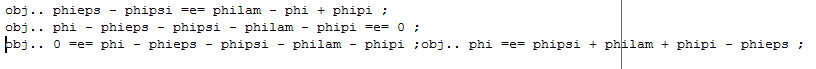
the two dots".." is always required.

equ\_type refers to the symbol between the two expressions, and can be the following types:



example:  


the obj equation is equivalent with any of the following forms.



**scalar equations** will produce at most one equation, which contains both scalar variables and indexed variables. consider the following example:  


indexed equations is as follows. Domain checking ensures that the domain over which an equation is defined must be the set or a subset of the set over which the equation is declared.



the extension to two or more index position is as follows:



using labels explicitly can be done by using quotes around the label. Consider the following example:



**3.arithmetic operators in equation definition**

example showing parentheses and exponentiation:  


**4.functions in equation definition**

**5.preventing undefined operations in equations**

certain operations can be undefined at particular values for the arguments. for example, the log-function is undefined when the argument is 0, and division by 0 is another example. one way to solve it is to add bounds to the variable. consider the following function reference:



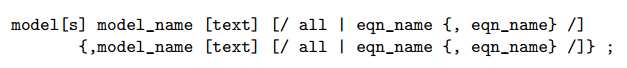
the bounding on c(t) away from 0 preventing the log-function from being undefined.

**5. data handling aspects of equations**

**chapter 9 model and solve statement**

**1. the model statement**

**1.1 the syntax**



model\_name is the internal name(also called an identifier).

the accompany text is used to describe the element or set immediately preceding it.

eqn\_name is the name of an equation that has been declared prior to the model statement.

the regulations for model\_name and text is the same as other identifier and its text.

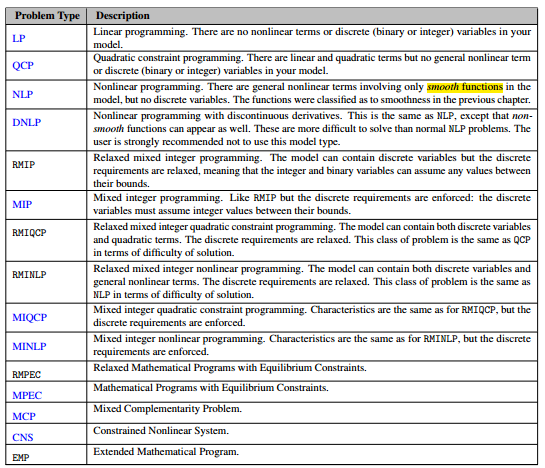
an example of a model definition is shown below:  
 

the model is called 'transport'. the keyword 'all' is a shorthand for all known(declared) equations.

several models can be declared in one statement as shown follows:



**1.2 classification of models**

the problem type and their identifiers are listed as follows:  


where，smooth function（光滑函数）是指在其定义域内无穷阶数连续可导的函数。

线性函数:一阶多项式函数

LP线性规划：目标函数和约束条件都是线性的，无非线性部分也没有离散变量（整数或二值变量）

NLP非线性规划：目标函数或约束条件中含有非线性的部分。这些非线性仅涉及光滑函数，没有离散变量。

QCP二次约束规划：包含线性和二次项，但是没有非线性部分（没有光滑函数）和离散变量。

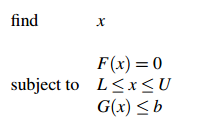
MIP混合整数规划：模型中包含离散变量

MINLP混合整数非线性规划：模型中包含离散变量和非线性部分。

①constrained nonlinear systems(CNS)

不懂这个是什么意思

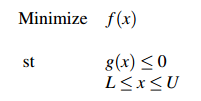
mathematically, a Constrained Nonlinear System(CNS) model looks like:



there is no objective and therefore no marginal values.

②nonlinear programming with discontinuous derivations(DNLP)

mathematically, the DNLP looks like:



x is a vector of variables that are continuous real number.

f(x) is the objective function.

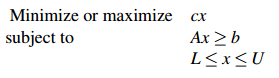
g(x) represents the set of constrains.

L and U are vectors of lower and upper bounds on the variables.

this is as NLP, except that non-smooth functions(abs, min, max) can appear in f(x) and g(x)

③linear programming (LP)

mathematically, the LP looks like:



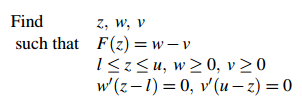
x is a vector of variables that are continuous real number.

cx is the objective function.

Ax represents the set of constrains.

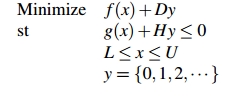
L and U are vectors of lower and upper bounds on the variables.

④ mixed complementarity problem(MCP)

mathematically, the MCP looks like   


④mixed integer nonlinear programming (MINLP)

mathematically, the MINLP looks like:



x is a vector of variables that are continuous real number.

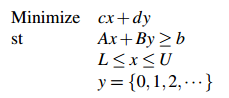
f(x)+Dy is the objective function.

g(x)+Hy represents the set of constrains.

L and U are vectors of lower and upper bounds on the variables.

⑤ mixed integer programming (MILP)

mathematically, the MINLP looks like:



x is a vector of variables that are continuous real number.

y is a vector in variables that can only take integer values.

cx+dy is the objective function.

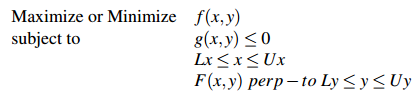
Ax+By>=b represents the set of constrains.

L and U are vectors of lower and upper bounds on the continuous variables.

y = {0,1,2,..}is the integrality requirement on the integer variables y.

⑥mathematical program with equilibrium constrains(MPEC)

mathematically, the MPEC looks like:



x and y are vectors of continuous real variables. the variables x are often called control or upper-level variables. in contrast, the variables y are often called state or lower-level variables.

f(x,y) is the objective function.

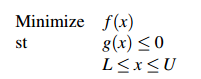
g(x,y) represents the traditional constrains.

the functions F(x,y) and bounds Ly and Uy define the equilibrium constrains. if x is fixed, the F(x,y) and bouds Ly and Uy define the MCP.

the perp-to indicates such a complementary relationship holds.

⑦nonlinear programming(NLP)

mathematically, the NLP looks like:



x is a vector of variables that are continuous real number.

f(x) is the objective function.

g(x) represents the set of constrains.

L and U are vectors of lower and upper bounds on the variables.

⑧quadratically constrained programs (QCP)

QCP is a special case of the NLP in which all nonlinearities are required to be quadratic.

⑨ mixed integer quadratically constrained programs (MIQCP)

MIQCP is a special case of MINLP in which all nonlinearities are required to be quadratic.

1.3 model attributes

model attributes can be accessed through:



some of the attributes are mainly used before the solve statement to provide information to GAMS or the solver link. others are set by GAMS or solver link and hence are used after solve statement.

moreover, some of the input attributes can also be set globally via an option statement or the command line, e.g.



**2. the solve statement**

**2.1 the syntax**

the syntax in GAMS for a model declaration is:



model\_name is the name of a model as defined by a model statement.

var\_name is the name of objective variable that is being optimized.

model \_type is one of model types.

an example is shown below:

  
the objective variable must be scalar or of type free.

**2.2 requirements for valid solve statement**

when the GAMS encounters a solve statement, symbolic equations, objective variable, the equation fits into the model type ,all sets and parameters will be verified.

**2.3 actions triggered by the solve statement.**

**3.programs with several solve statements**

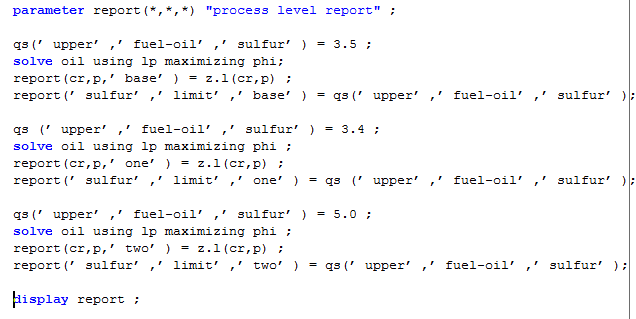
several solve statement can be processed in the same program.

**3.1 several models**

example:  


when there is more than one solve statement, GAMS uses as much information as possible to provide a starting point in the search for the next solution.

**3.2 sensitivity or scenario analysis**

the following example shows not only how simply sensitivity analysis can be done, but also how the associated multi-case reporting can be handled:  


**3.3 iterative implementation of non-standard algorithms**

Another use of multiple solve statements is to permit iterative solution of different blocks of equations, solution values from the first are used as data in the next. for exmaple:

