

ADDGOALS

ADDGOALS forces all arguments to be computed prior to the first argument in the function

Syntax

1. ADDGOALS([GoalValExp](#), [SubGoal](#), [[SubGoal](#)!])
2. ADDGOALS([GoalObject](#)(...), [SubGoal](#), [[SubGoal](#)!])

Arguments

Method 1

- [GoalValExp](#) (a [ValExp](#)) is the parameter or expression that should be the result of the function;
- [SubGoal](#) (an [InpVar](#)) is a parameter or object that should be computed prior to the [GoalValExp](#).

Method 2

- [GoalObject](#)(...) is the Object that should be the result of the function;
- [SubGoal](#) (an [InpVar](#)) is a parameter or object that should be computed prior to the [GoalObject](#)(...).

Remarks

1. When complex object structures are created in [Quaestor](#), the knowledge engineer might require the possibility to force the computation of parameters in order to manage the evaluation order. By adding the ADDGOALS function around an value, expression, [TeLiTab](#) or Object, in combination with the forced goals ([SubGoal](#)), the value, expression, [TeLiTab](#) or Object is still received as result but after the other goals are carried out.
2. In method 2, Object(..) means that in the object itself additional [TopGoals](#) etc. can be defined. See also the documentation on the use of objects in [Quaestor](#).

Examples

A=ADDGOALS(B,C)

When [A](#) is asked as top goal, both [C](#) and [B](#) are added to the goal list of [Quaestor](#). Thus both both [C](#) and [B](#) should be determined.

In the [Workbase](#) this is shown by the fact that (for the example) for both [B](#) and [C](#) values are request to the user while they do not depend on each other. When the following values are given:

B=5

C=7

Parameter [A](#) will have 5 as a result (being determined by [B](#)). Moreover, both [B](#) and [C](#) are part of the solution.

Quick links: [Functions overview](#) | [Attribute overview](#) | [Constants overview](#) | [Dimensions overview](#)