

UNIVERSIDADE ESTADUAL DE CAMPINAS  
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA  
**UNICAMP / IMECC**

## **INTRODUÇÃO A LINGUAGEM AIMMS E DICAS DE USO**

Campinas, São Paulo - Brasil  
Junho/2015

## **INTRODUÇÃO A LINGUAGEM AIMMS E DICAS DE USO :**

Esse trabalho foi feito com o intuito de transmitir conhecimento que foi adquirido através do estágio na empresa UniSoma. Aqui mostrarei um pouco do AIMMS, que é usado como plataforma para os projetos na UniSoma, referências para aprender mais sobre ele e dicas.

Orientadora: Profa. Kelly Poldi  
Aluna : Camila Fernandes Salles RA: 119195

Campinas, São Paulo - Brasil  
Junho/2015

## Resumo

Esse é um manual para um breve conhecimento do software AIMMS, será descrito os principais identificadores, procedimento, como usá-los, como fazer um otimização, como fazer uma interface gráfica para o usuário, instalar o software, salvar um caso e um exemplo resumindo tudo o que foi aprendido.

### Palavras – chave :

AIMMS, Manual AIMMS, tutorial iniciantes AIMMS, otimização AIMMS , identificadores AIMMS, interface gráfica AIMMS, principais funções AIMMS , índice AIMMS .

Campinas, São Paulo - Brasil  
Junho/2015

## Abstract


This is a guide to a brief knowledge of AIMMS software, will be described the main identifiers, procedures, how to use them, how to make an optimization, how to make a graphical user interface, install the software, save a case and one example summarizing all what has been shown.

## Keywords :

AIMMS, guide AIMMS, beginners tutorial AIMMS, optimization AIMMS , identifiers AIMMS, graphical interface AIMMS, main functions AIMMS , index AIMMS .

Campinas, São Paulo - Brasil  
June/2015

## Sumário

1	Introdução .....	1
1.1	Como instalar .....	1
1.2	Lógica AIMMS.....	7
2	Entendendo o software .....	8
2.1	Começando um novo projeto.....	8
2.2	Identificadores.....	9
2.2.1	Conjunto.....	10
2.2.1.1	<i>Subset of</i> .....	10
2.2.1.2	<i>Index</i> .....	10
2.2.1.3	<i>Order by</i> .....	10
2.2.1.4	<i>Definition</i> .....	11
2.2.1.5	<i>Initial Data</i> .....	11
2.2.1.6	O símbolo  .....	11
2.2.2	<i>Parameter</i> .....	11
2.2.2.1	<i>Index Domain</i> .....	12
2.2.2.2	<i>Range</i> .....	12
2.2.2.3	A unidade .....	12
2.2.2.4	<i>Definition</i> .....	12
2.2.3	<i>Variable</i> .....	12
2.2.3.1	<i>Index Domain</i> .....	13
2.2.3.2	<i>Range</i> .....	13
2.2.4	<i>Constraint</i> .....	13
2.2.4.1	<i>Index Domain</i> .....	14
2.2.4.2	Unidade da restrição .....	14
2.2.4.3	<i>Definition</i> .....	14
2.2.5	<i>Element Parameter</i> .....	14
2.2.5.1	<i>Range</i> .....	15
2.2.5.2	<i>Index Domain</i> .....	15

2.2.6 <i>String Parameter</i> .....	17
2.2.6.1 <i>Index Parameter</i> .....	17
2.2.6.2 <i>Definition</i> .....	17
2.2.7 <i>Mathematical Program</i> .....	17
2.2.7.1 <i>Objective</i> .....	18
2.2.7.2 <i>Direction</i> .....	19
2.2.7.3 <i>Constraints</i> .....	19
2.2.7.4 <i>Variables</i> .....	21
3 <i>Procedimento (Procedure)</i> .....	21
3.1 <i>Body</i> .....	21
3.2 <i>Identificadores locais</i> .....	22
4 <i>Realizando a otimização</i> .....	22
5 <i>Função (Function)</i> .....	23
5.1 <i>Arguments</i> .....	24
5.2 <i>Index Domain</i> .....	24
5.3 <i>Range</i> .....	24
5.4 <i>Unit</i> .....	24
5.5 <i>Body</i> .....	24
6 <i>Declaration</i> .....	25
7 <i>Section</i> .....	25
8 <i>Principais funções usadas</i> .....	25
8.1 <i>StringToElement</i> .....	26
8.2 <i>Element</i> .....	26
8.3 <i>StringToUpper</i> .....	26
8.4 <i>Val</i> .....	26
8.6 <i>Return</i> .....	27
8.7 <i>For</i> .....	27
8.8 <i>While</i> .....	27
8.9 <i>If</i> .....	27
8.10 <i>First / Last</i> .....	27
8.11 <i>SetElementAdd</i> .....	27
8.12 <i>Somatória</i> .....	27

9 Dicas .....	28
9.1 Completando argumentos .....	28
9.2 Compilar Modelo .....	28
10 Debugger .....	28
11 Problema da mochila .....	32
12 Interface Gráfica .....	35
12.1 Criando nova página .....	37
12.2 Inserindo texto .....	37
12.3 Inserindo tabela .....	39
12.4 Inserindo botão .....	45
14 Salvar Data .....	53

## 1 Introdução

Pesquisa Operacional é um campo da matemática aplicada que estuda métodos para tomada de decisões, surgiu na segunda guerra mundial e hoje é muito usada por empresas, principalmente indústrias, para as áreas de planejamento. A UniSoma, uma empresa de consultoria localizada em Campinas, desenvolve projetos para a tomada de decisões de empresas, todos seus projetos são feitos a medida da empresa que contrata, sendo primeiro feito um estudo do problema e depois o desenvolvimento. O objetivo do trabalho é ajudar a quem gostaria de aprender a usar a linguagem da plataforma para o desenvolvimento do projeto. O software AIMMS além de implementar um problema de programação linear ou não linear ainda fornece a construção da interface a ser apresentada ao usuário.

### 1.1 Como instalar

Como o AIMMS é um software pago, então uma opção para os estudantes ou para quem queira usar não com muita complexidade, limitado a 200 identificadores é a licença instantânea que esta na primeira página do site, <http://www.aimms.com/licensing/free-licenses/free-student-license/> , o numero de licença e o código de ativação :

#### Instructions

To use the free student license you can [download](#) and install AIMMS\* and use the information below to activate the license when you open AIMMS for the first time:

License Number : **080.080.080.015**

Activation Code : **XyGRvc-BvcGuX-TMrXNu-tpALYP-LTJCzL**

Please copy the activation code to prevent typos. If you [register](#) you can stay up to date by receiving AIMMS related news. Select to keep informed on the Academic license and you will be kept up to date about changes in the academic licenses, such as addition of new solvers, and the renewal of the free academic licenses. (\*) The use of the free student license allows the usage of all Production Releases and Feature Release that are available for download. For unrestricted versions of the software, we recommend you to consider the [Free AIMMS Academic License](#).

Figura 1

Para ter a licença para trabalhar com mais variáveis precisamos fazer um cadastro e ser aprovado , para isso primeiramente clique em "*free AIMMS academic license*" e



registre-se, entre no seu email registrado e confirme o cadastro, você receberá outro email dizendo que, em alguns dias, eles lhe enviaram uma resposta:

## Registration

If you are eligible for a Free AIMMS Academic License, you can request your license through the registration form below. In case you do not wish to register, you can still use the [free limited student license](#) that requires no registration. *\*required fields*

First name*	<input type="text"/>
Last name*	<input type="text"/>
E-mail address*	<input type="text"/>
Academic Institution*	<input type="text"/>
University Website (e.g. asu.edu, nus.edu.sg, utwente.nl)*	<input type="text"/>
Country*	<input type="text" value="Netherlands"/>
Profile*	<input type="text" value="Student"/>
Intended use*	<input type="text"/>
Remark	<input type="text"/>

Figura 2

Após ter conseguido a licença, vamos baixar o Aimms. Para isso vá em <http://www.aimms.com/downloads/aimms/download-aimms/>. Nessa página temos a opção rápida ou a personalizada. A primeira somente funciona para Windows 64 bits, caso não seja seu sistema operacional, então deve escolher a opção personalizada (*AIMMS 4.6.4 Instalation-free*):

## Quick download AIMMS 4.6

(Windows 64 bit, incl. prerequisites)

**Please note:** You should have a valid AIMMS license to use AIMMS. If you don't have one yet, you can request a [free license](#). Also note that AIMMS 4.6 is a version of AIMMS that doesn't need an installation. For an overview of the changes in this version, please see the [release notes](#). To use it most effectively, you can download and install the [AIMMS Launcher](#).

Select View					
Application	Operating System	Architecture	Flavor	Major Release	Software Update
any	Windows	64-bit	UTF8	4.6	4
<b>Current view</b>					
AIMMS 4.6.4 Installation-free ** (Windows 64-bit UTF8)					166.5 MB
** This version requires <b>no installation</b> .					
<a href="#">Aimms-4.6.4.277-x64.exe</a>					
AIMMS PRO Package 4.6.4 (Windows 64-bit UTF8)					88.6 MB
<a href="#">AimmsPROPackage-4.6.4.277-x64-x64.7z</a>					
<b>Third party software (prerequisites)</b>					
VC Redist SP1 8.0.50727.4053 <a href="#">vcredist_x86.exe</a> visit company <a href="#">website</a> New AIMMS versions (3.9.4/3.10 FR3 and higher) require the VC Redistributable SP1 8.0.50727.4053 to be installed on your system.					2.6 MB
VC Redist SP1 (x64) 8.0.50727.4053 <a href="#">vcredist_x64.exe</a> visit company <a href="#">website</a> New 64-bit AIMMS versions (3.9.4/3.10 FR3 and higher) require the VC Redistributable SP1 8.0.50727.4053 (x64) to be installed on your system.					3.1 MB
VC Redist SP1 8.0.50727.762 (old AIMMS versions!) <a href="#">vcredist_x86.exe</a> visit company <a href="#">website</a> Old AIMMS versions (before 3.9.4/3.10 FR3) require the VC Redistributable SP1 8.0.50727.762 to be installed on your system.					2.6 MB
VC Redist SP1 (x64) 8.0.50727.762 (old AIMMS versions!) <a href="#">vcredist_x64.exe</a> visit company <a href="#">website</a> Old 64-bit AIMMS versions (before 3.9.4/3.10 FR3) require the VC Redistributable SP1 8.0.50727.762 (x64) to be installed on your system.					3 MB
Microsoft .NET 2.0 (x64) <a href="#">NetFx64.exe</a> visit company <a href="#">website</a> Microsoft .NET 2.0 (x64)					47.7 MB

Figura 3

Sempre lembrando de verificar todos os pre requisitos especificados acima e instalar o necessário em “*Third party software*”.

Após baixar o AIMMS, como inserir a licença?

Abra o programa, vá em *tools, license, license configuration*.



Figura 4

Após isso irá abrir uma janela, clique em *install license*, irá abrir outra janela:

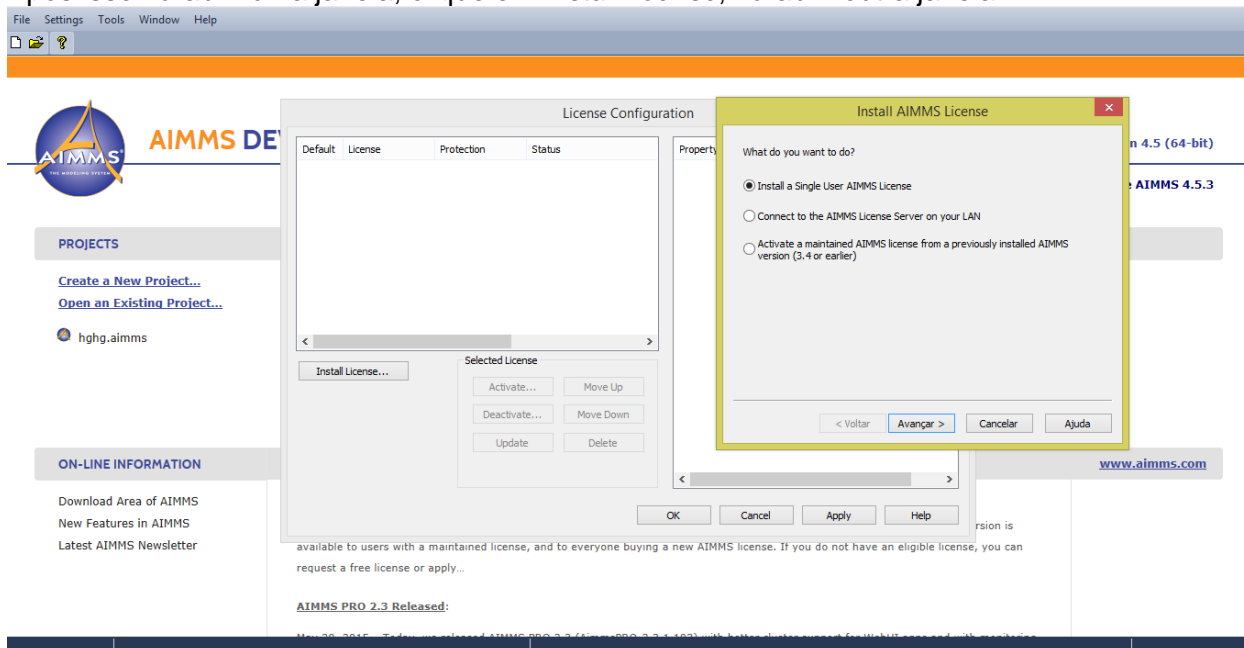


Figura 5

Nessa janela vamos instalar nossa licença. Selecione *Install a Single user* como na figura acima, de *next* e então coloque o número da sua licença:

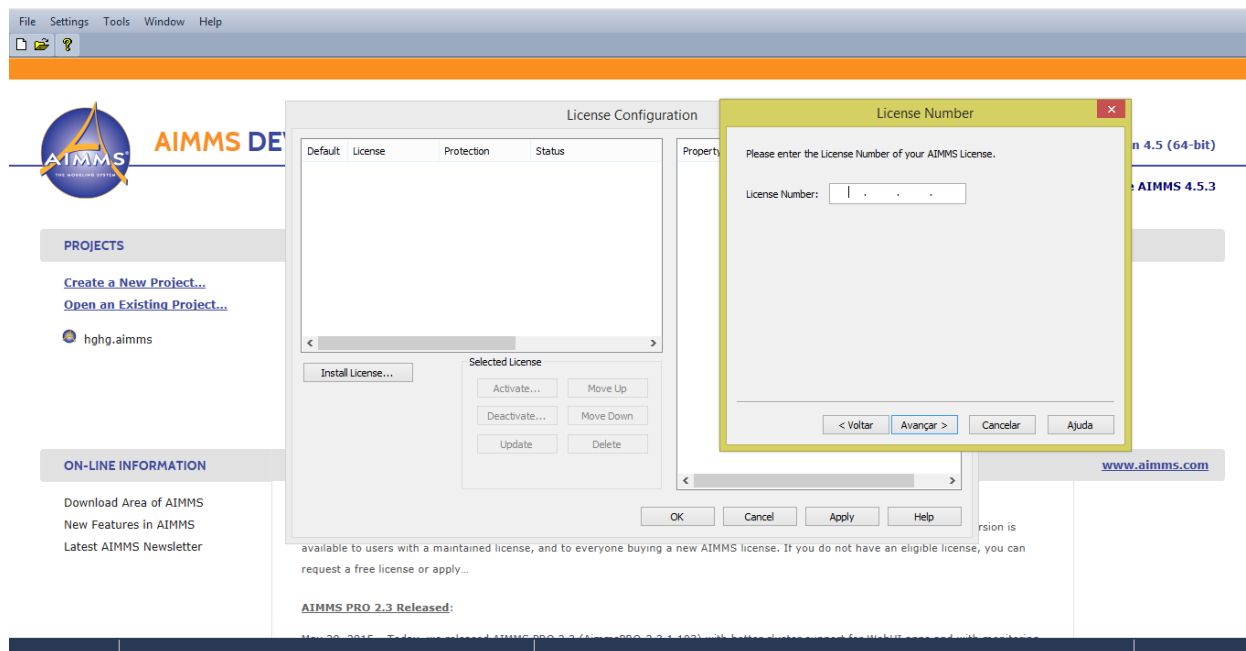


Figura 6

Apertando *Next*.

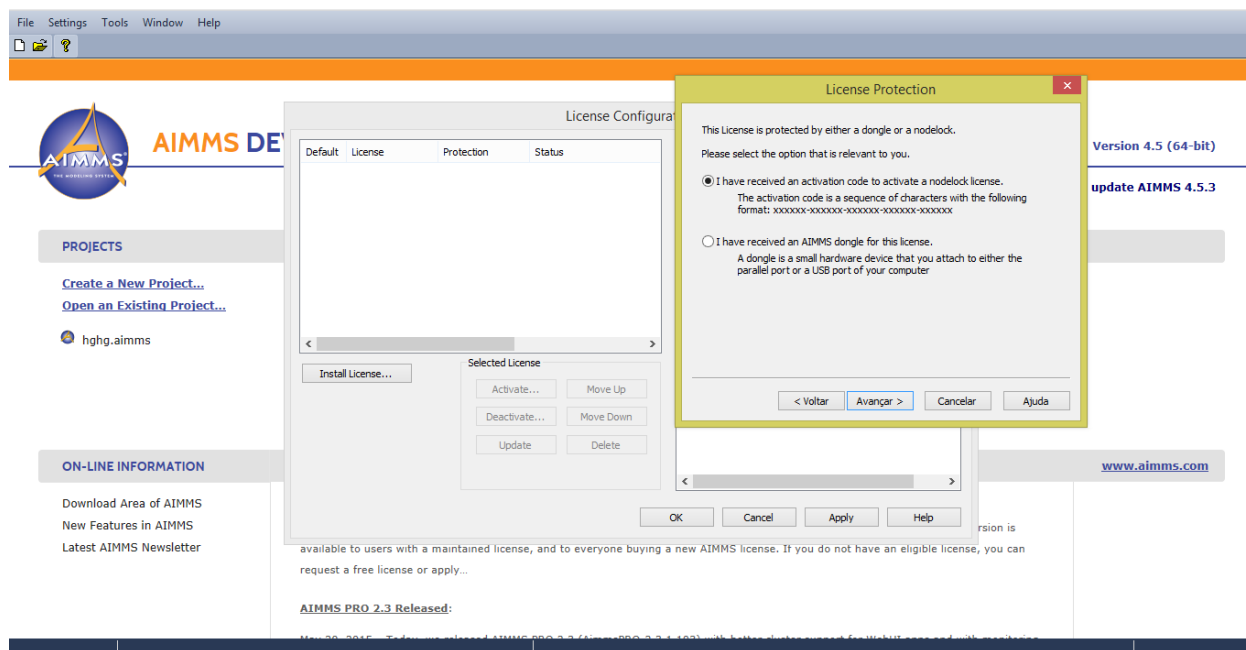


Figura 7

Insira a opção como da figura e *Next*

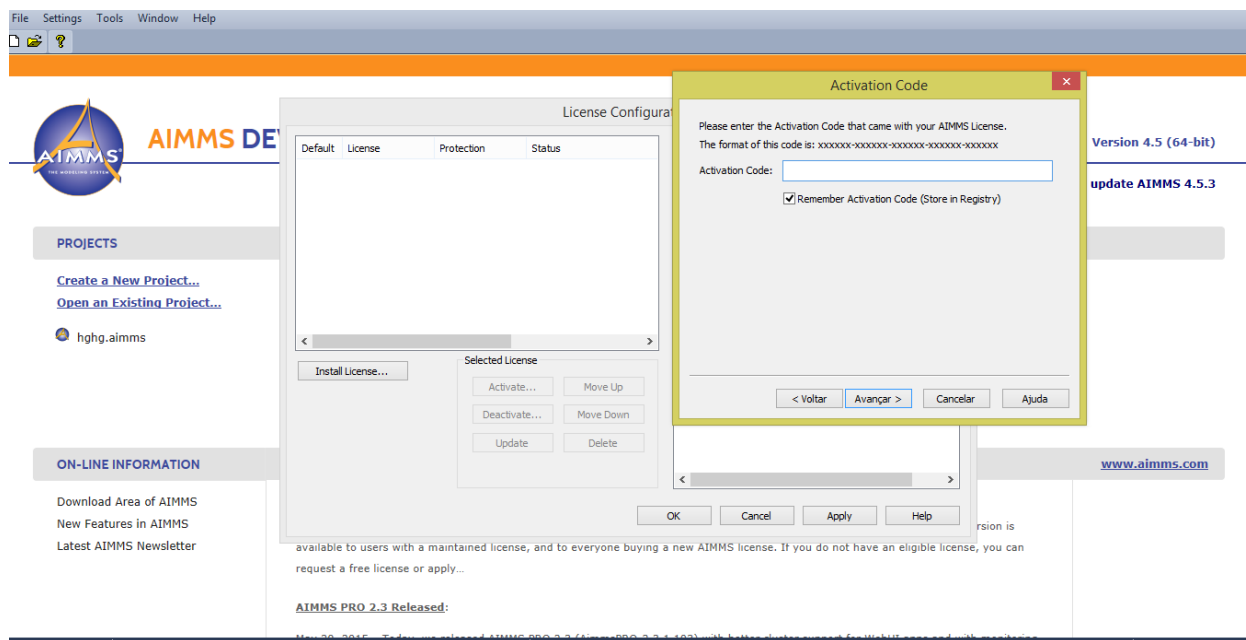


Figura 8

Coloque o código de ativação recebido

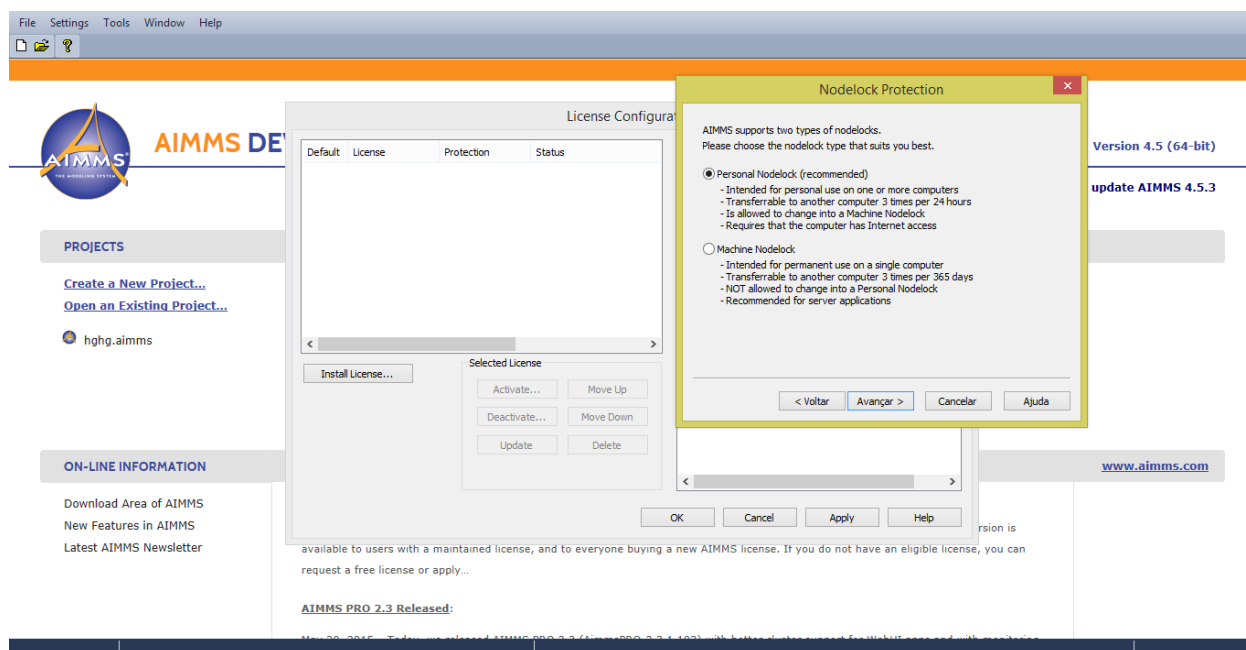


Figura 9

Selecione a primeira opção, avance

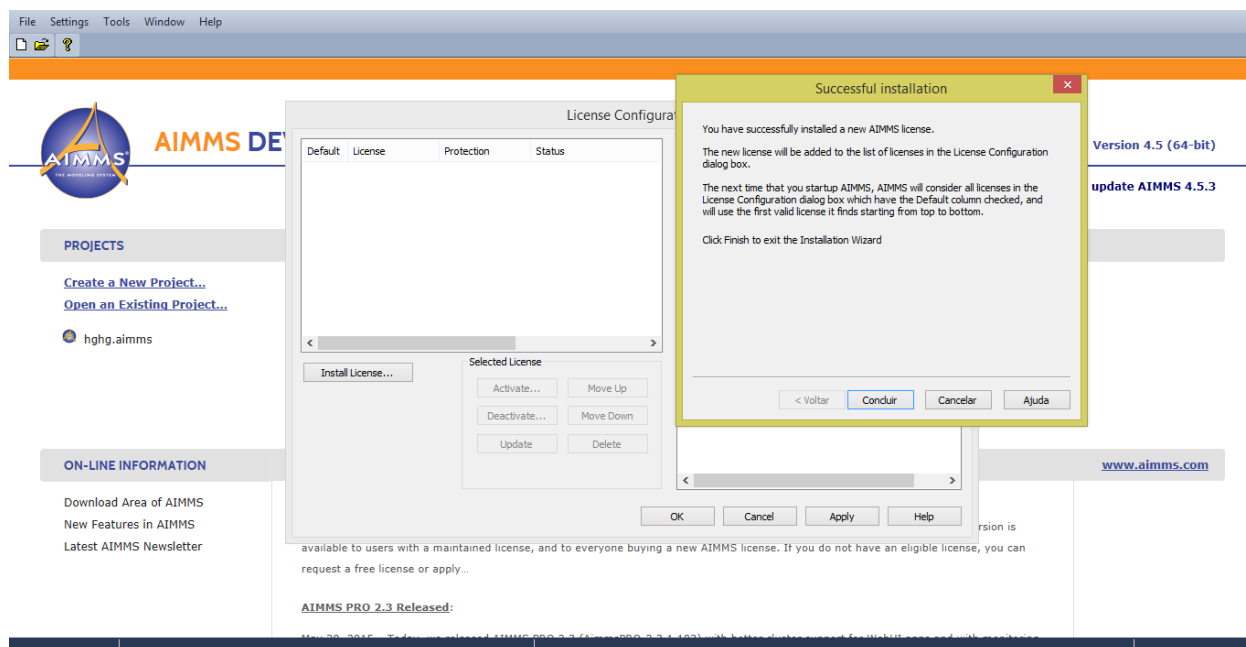


Figura 10

E Finalmente concluímos a ativação da licença.

Para se aprofundar melhor e outros lugares para entender o AIMMS são:

- Fazer o G-AIMMS (<http://www.aimms.com/community/g-aimms/witch-apprentice/>)
- Tutorial for Beginners (<http://www.aimms.com/downloads/tutorials/tutorial-for-beginners/>)
- Tutorial for Professionals (<http://www.aimms.com/downloads/tutorials/tutorial-for-professionals/>)

## 1.2 Lógica AIMMS

É importante sempre ter em mente que AIMMS trabalha com conjuntos (Set's), então sempre primeiramente definimos o conjunto a ser trabalhado, associamos a um índice, para poder percorrer ou associar a esse conjunto em outros lugares. Existem os parâmetros, que podem ser de números, letras ou só apontadores a outros parâmetros. Um parâmetro pode ser em função de um conjunto, chamamos isso de indexação, se temos o peso como parâmetro, podemos fazer o peso em função dos produtos, isso quer dizer que teremos um peso para cada produto. Por exemplo, temos sacolas1, sacolas2 que são elementos do conjunto "Sacolas" e produto1, produto2 que são elementos do conjunto "Produtos" e queremos associar quantos produtos por sacola, isso quer dizer que nosso parâmetro será quantidade e os índices serão sacola x produto, assim vamos indexar os conjuntos: Seja Prod o index do conjunto "Produtos" e Sac o index do conjunto "Sacolas" então basta associar os índices dos conjuntos ao parâmetro que teremos a tabela espaço para ser preenchido o valor associado a cada produto X sacola. Exemplo : Quantidade (Produtos,Sacola).

Podemos ter procedimento e funções. Em procedimento podemos fazer cálculos e definir valor de parâmetros, preencher dados, fazer importação e exportação, critérios de decisão ( IF,WHILE,FOR,.. ), etc..

Importatíssimo para otimização, temos as restrições (*Constrains*), funções objetivo (*Mathematical Program*) e variáveis a ser otimizadas (*Variable*) .

## 2 Entendendo o software

### 2.1 Começando um novo projeto

Vamos começar aprender a usar o Aimms com um novo projeto, assim primeiro apresentarei a base teórica para depois ir ao desenvolvimento :

Clique no ícone do aimms, irá aparecer a página inicial (Figura1), para começarmos um novo projeto clique na pagina em branco e de um nome a ele.

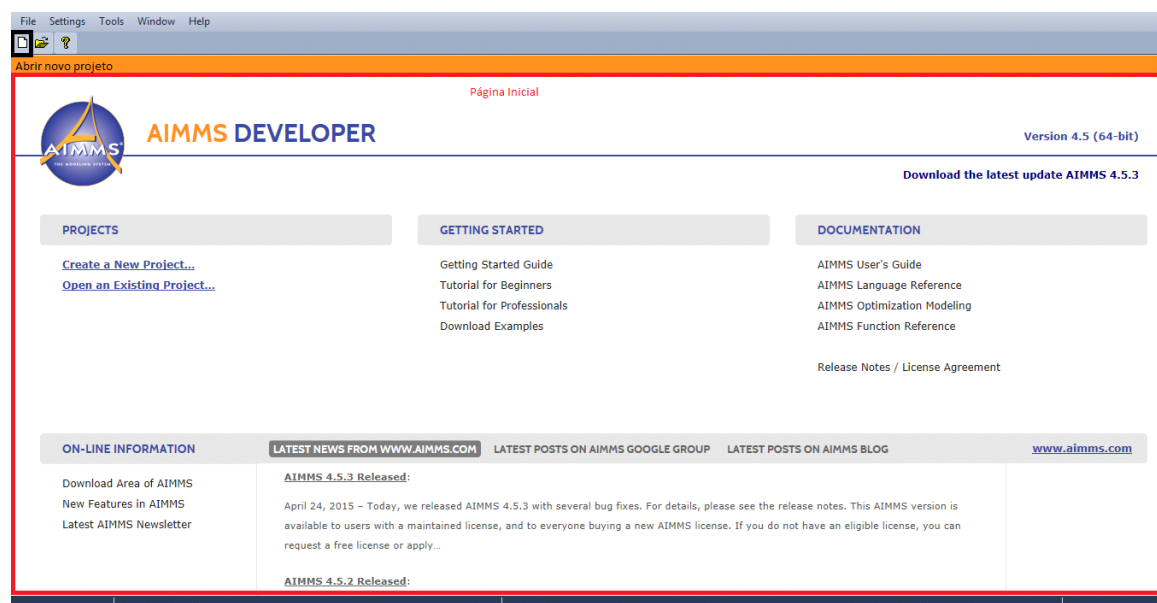


Figura 11

Aqui darei o nome de NomeDoProjeto. O novo projeto abre somente com o *Model Explorer* e a Barra de Ferramentas, que é o que precisaremos inicialmente (Figura 2). Note que existem 6 itens previamente dentro do *Model Explorer*.

- Main “nome do projeto” : Aqui é biblioteca principal em que declararemos nosso problema, mas podemos adicionar mais bibliotecas, como veremos mais para frente.

- *Declaration* é uma section inicialmente aberta, aqui que faremos as declarações do parâmetros e conjuntos. Note que parâmetros, conjuntos, restrições e outros identificadores que estão juntos na barra de ferramentas só podem ser adicionados dentro de uma *section*, já para funções e procedimentos é ao contrário, eles só podem ser colocados fora da *section*.

- *MainInitialization* é um procedimento previamente definido, tudo o que for colocado dentro dele será executado na inicialização do programa.

- *MainExecution* é o mesmo do “*MainInitialization*”, mas será executado na otimização do programa.

- *MainTermination* é o mesmo do “*MainInitialization*”, mas será executado antes do fechamento do programa.

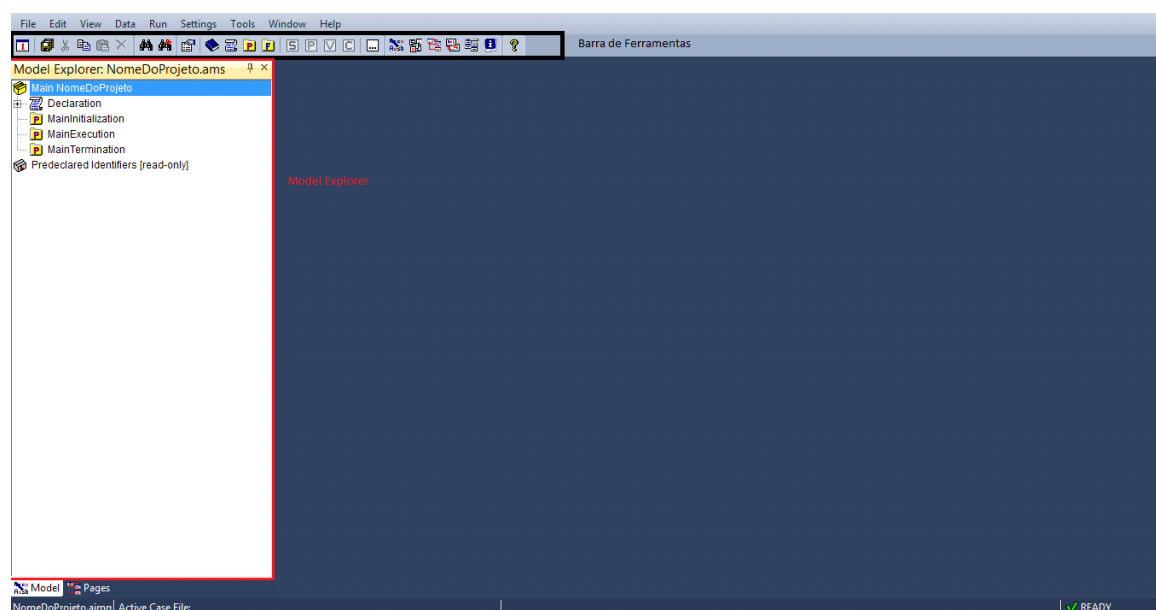






Figura 12

## 2.2 Identificadores

São “funções” previamente definidas pelo software, aqui falaremos sobre algumas.

Sempre que ocorrer modificações em algum identificador, para sair excluindo as modificações clique . Se quiser sair salvando internamente (não o programa, para isso clique sempre em salvar ) clique  e se não quiser sair, mas só salvar internamente, então clique em . O símbolo  é sempre para visualizar os dados do identificador.

Para selecionar algo .



## 2.2.1 Conjunto ( Set )

Conjunto a ser declarado. Basta dar 1 clique em *Declaration* ir na barra de ferramentas, clicar Set e dar um nome. Depois do processo abra ele dando dois cliques na set nomeada.

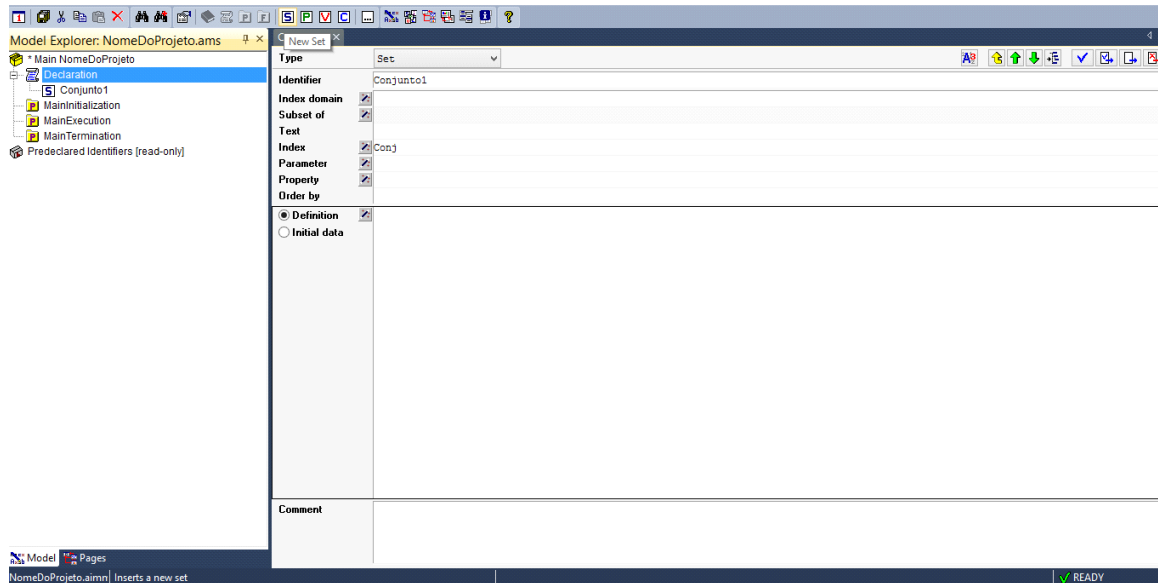


Figura 13

### 2.2.1.1 Subset of

é o campo que preenchemos se esse conjunto é subconjunto de outro conjunto. Basta colocar o nome do conjunto pai.

### 2.2.1.2 Index

é aqui que associamos um índice ao conjunto. Esse índice depois pode ser passado a parâmetros para percorrer elementos do conjunto.

### 2.2.1.3 Order by

é onde ordenamos o conjunto, é importante na hora que usar o índice, pois as vezes queremos percorrer um conjunto ordenado. Podemos ordenar passando o índice por exemplo.

#### 2.2.1.4 Definition

é o campo que podemos definir o conjunto, mas se for preenchido por aqui, ele não poderá ser alterado depois.

#### 2.2.1.5 Initial Data

é onde podemos definir o conjunto inicial, poderá ser alterado depois.

#### 2.2.1.6 O símbolo

é usado para preencher o conjunto, clicando nele abrimos

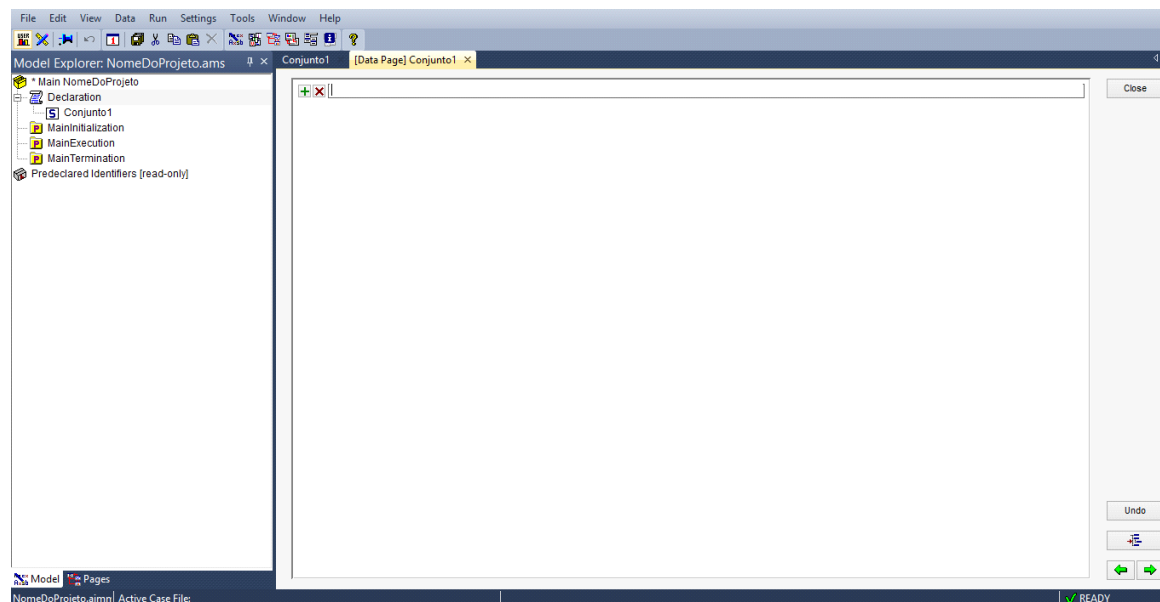


Figura 14

Que é onde podemos preencher o conjunto, ou alterar, ou visualizar depois de alguma modificação. Aqui o preenchimento poderá ser alterado.

#### 2.2.2 Parameter

É onde armazenaremos os valores relacionados aos conjuntos. Pode não ser relacionado a nenhum conjunto, se for relacionado, então devemos indexar o parâmetro.

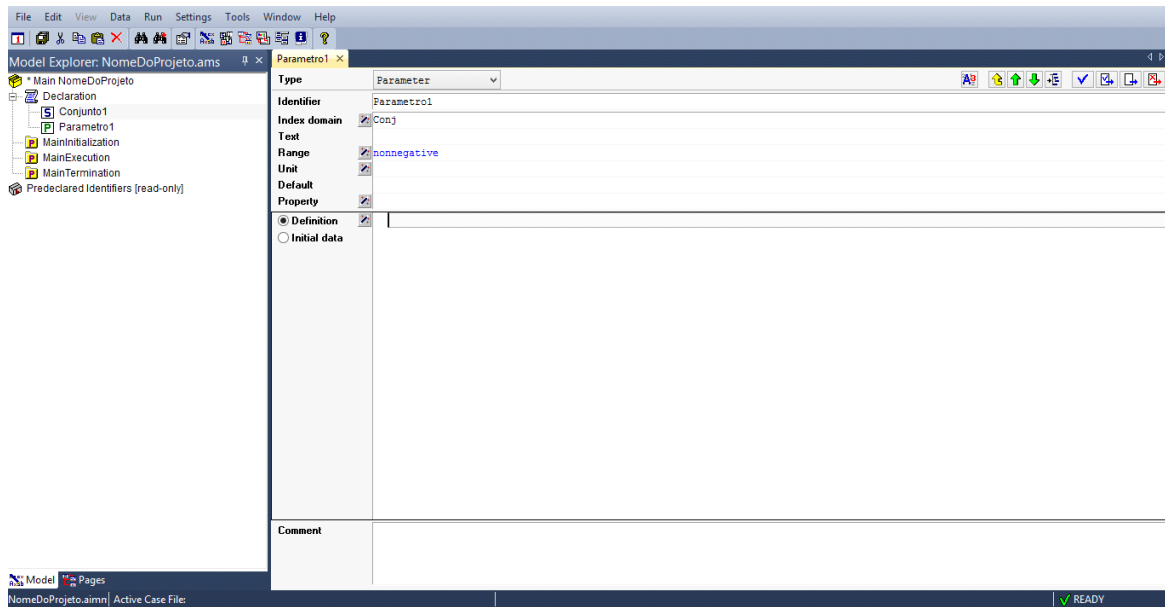



Figura 15

### 2.2.2.1 Index Domain

aqui que declaremos o índice com relação ao conjunto, para declarar mais de um índice basta colocar como (exemplo1,exemplo2,exemplo3,.....).

### 2.2.2.2 Range

é qual o domínio que estaremos trabalhando, se é binário, 0-1, 0- infinito, - infinito a infinito. Basta clicar em  e escolher.

### 2.2.2.3 A unidade

é a unidade do parâmetro, exemplo : se for o peso de algo, então deve ser declarado uma unidade de medida de peso; kg, g etc.

### 2.2.2.4 Definition

é onde podemos definir os dados, mas aqui eles não poderam ser modificados posteriormente.

## 2.2.3 Variable

É onde iremos declarar a variável a ser Max/Min.

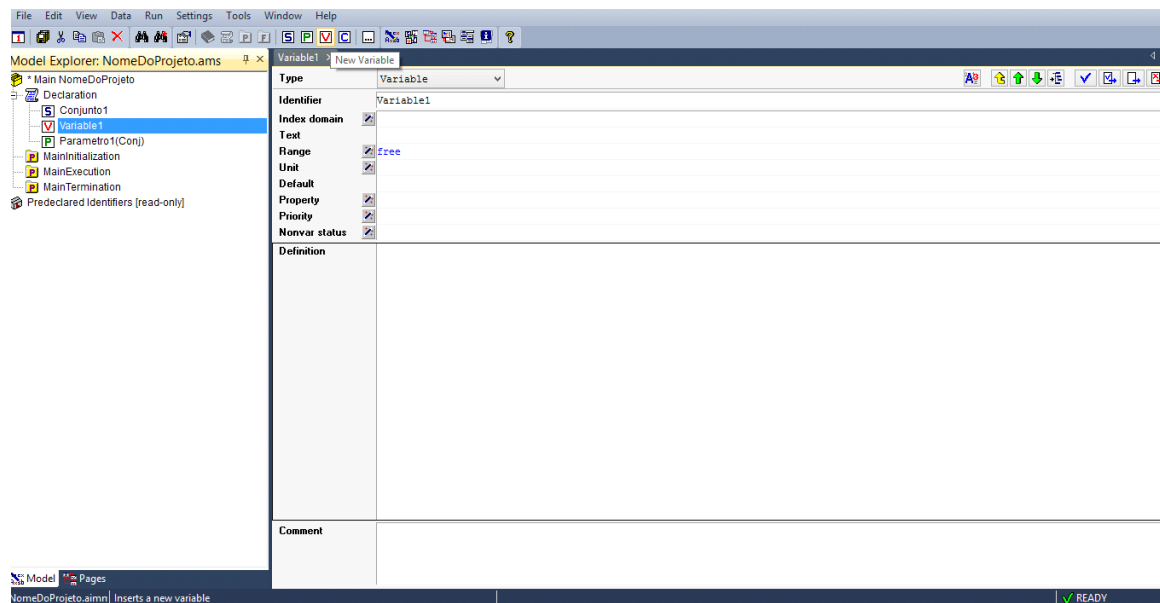


Figura 16

### 2.2.3.1 Index Domain

é similar ao caso do parâmetro, mas aqui lembre-se que a variável é para ser otimizada. Então teremos uma variável por índice por exemplo.

### 2.2.3.2 Range

é o domínio da variável.

### 2.2.4 Constraint

É onde colocaremos as restrições.

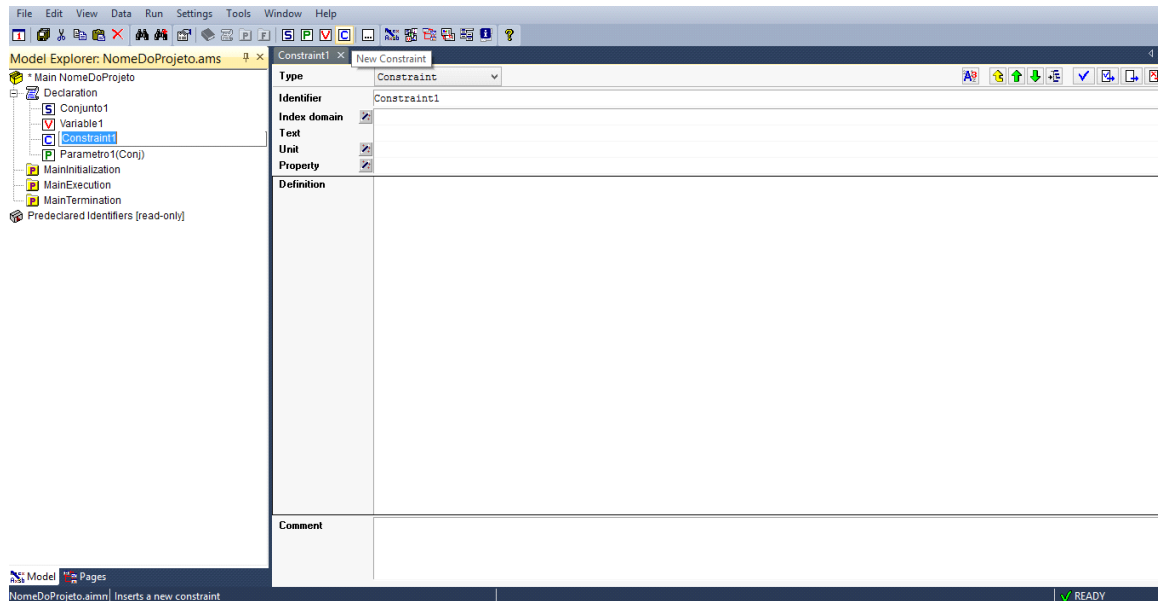


Figura 17

### 2.2.4.1 Index Domain

é o campo que será declarado o índice, se declarmos, então temos uma restrição por índice.

### 2.2.4.2 Unidade da restrição

por exemplo, o peso  $\leq 5$  ( kg) . Então Kg é a unidade.

### 2.2.4.3 Definition

é onde declararemos a restrição. Por exemplo

$Variable1 \leq 6$ . Se a *Constraint1* retornar 1 é por que a restrição foi satisfeita, se 0 então não foi satisfteita.

### 2.2.5 Element Parameter

É um apontador. Ele aponta para um elemento de um conjunto e guarda ele até que aponte para outro elemento. O *element parameter* pode ser declarado diretamente pelo campo *Parameter* da Set.

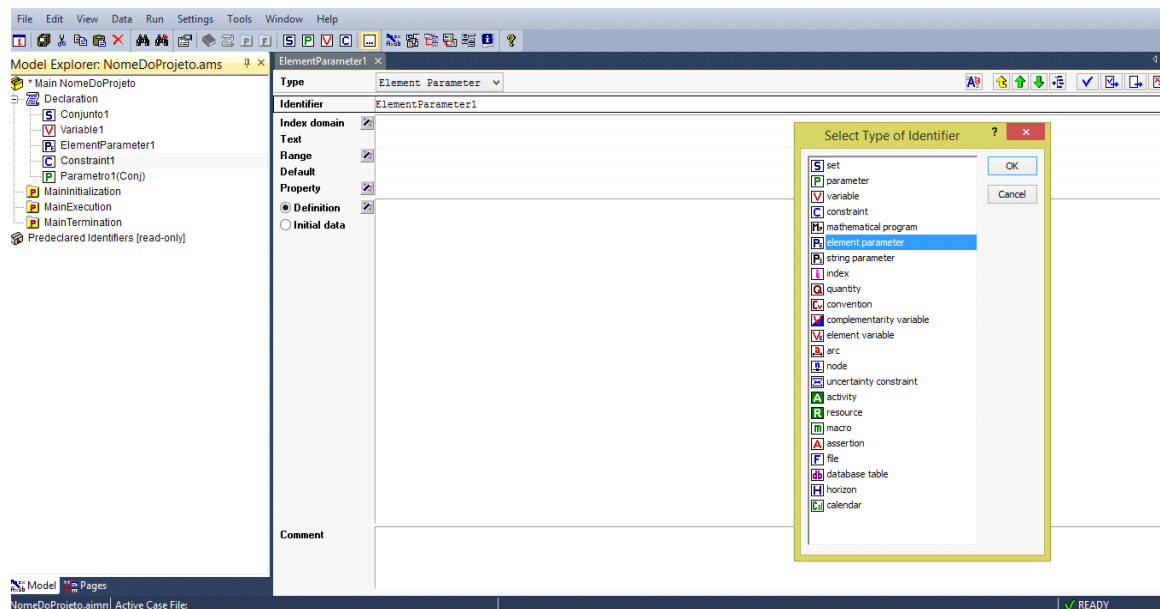



Figura 18

Para adicionar ele basta clicar em *Declaration*, aparecerá a barra de ferramentas, nos três pontos e então *Element Parameter*.

### 2.2.5.1 Range

é um campo obrigatório, aqui declararemos o conjunto para qual o *element parameter* irá apontar.

### 2.2.5.2 Index Domain

é onde pode ser declarado o índice, com isso teremos um vetor de *element parameter* para 1 índice e assim por diante. Depois de preenchido o set de elementos podemos ver o vetor de elemento *parameter* em .

Exemplo:

Preenchendo o conjunto

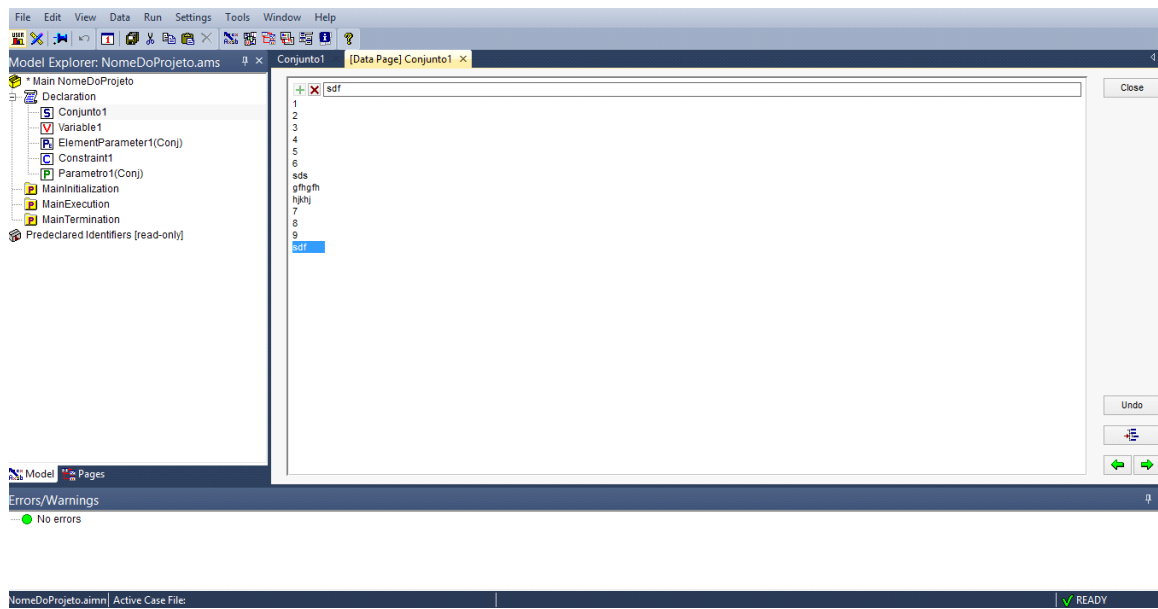



Figura 19

Coloque Conj no *Index Domain* do seu elemento *parameter*. Como é um exemplo muito simples, vamos colocar o mesmo conjunto no range (Conjunto1) e após clique em  (data).

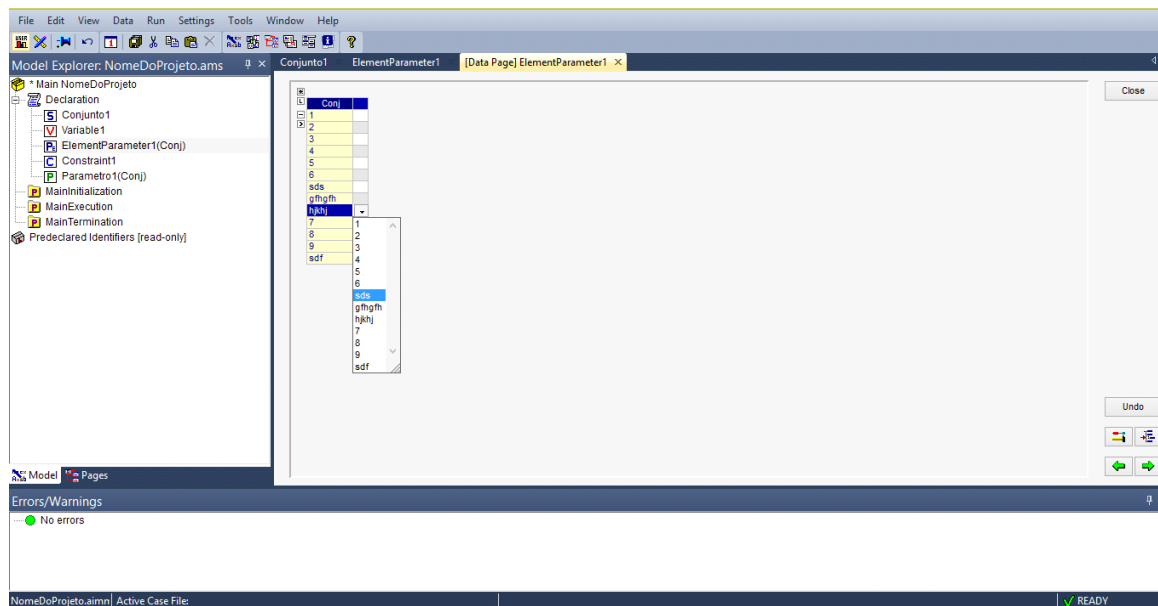


Figura 20

Podemos ver que para cada elemento do conjunto temos um elemento *parameter*, isso se deve ao *Index Domain*. E vemos também que para cada elemento *parameter* podemos atribuir um outro elemento do conjunto, isso se deve ao *Range*. Então se declararmos conjuntos diferentes nesses dois campos, vamos trabalhar com elementos de conjuntos diferentes no *data*.

## 2.2.6 String Parameter

*String Parameter* é um parâmetro para palavras/frases.

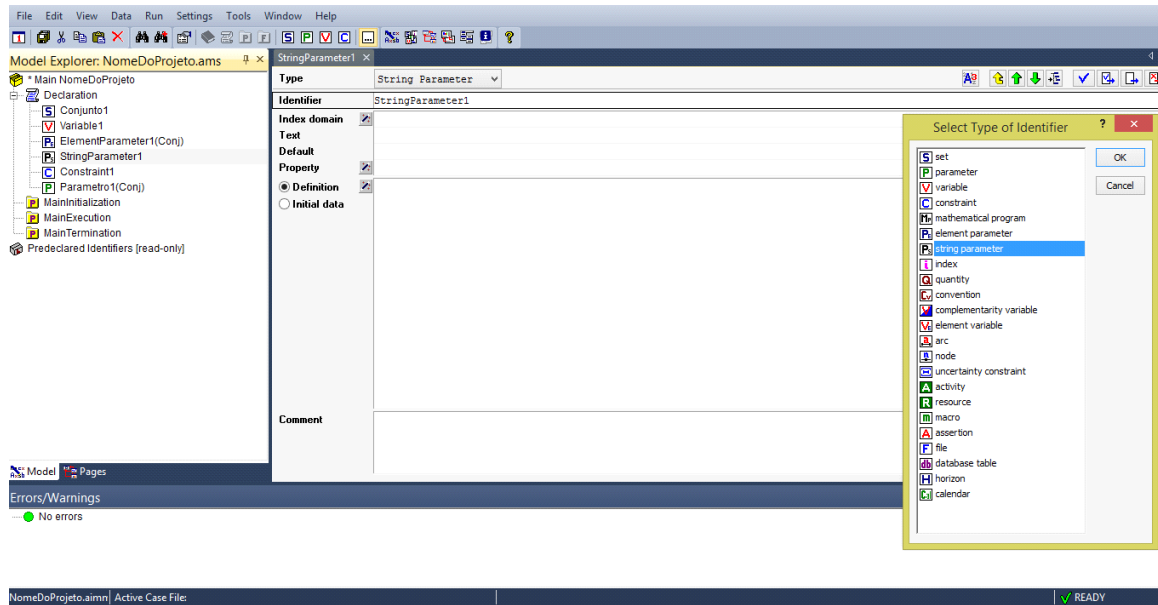


Figura 21

### 2.2.6.1 Index Parameter

é o mesmo visto anteriormente.

### 2.2.6.2 Definition

é onde podemos dar a palavra / frase/ número ao parâmetro, mas lembrando que se for declarado aqui não poderá ser modificado depois, em procedimento por exemplo.

## 2.2.7 Mathematical Program

é onde declaramos a função objetivo.



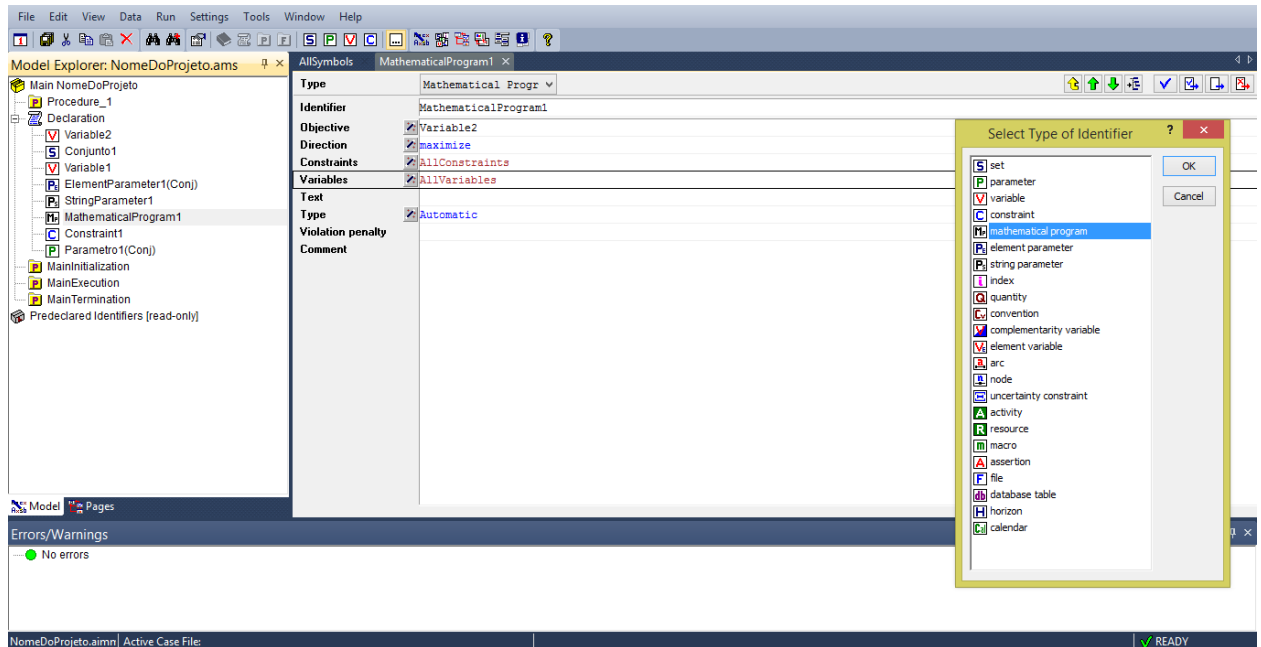


Figura 22

### 2.2.7.1 Objective

Declaramos neste campo a função objetivo, mas aqui só podemos colocar variáveis, logo vamos declarar um outra variável2 definida com a função objetivo, como por exemplo:

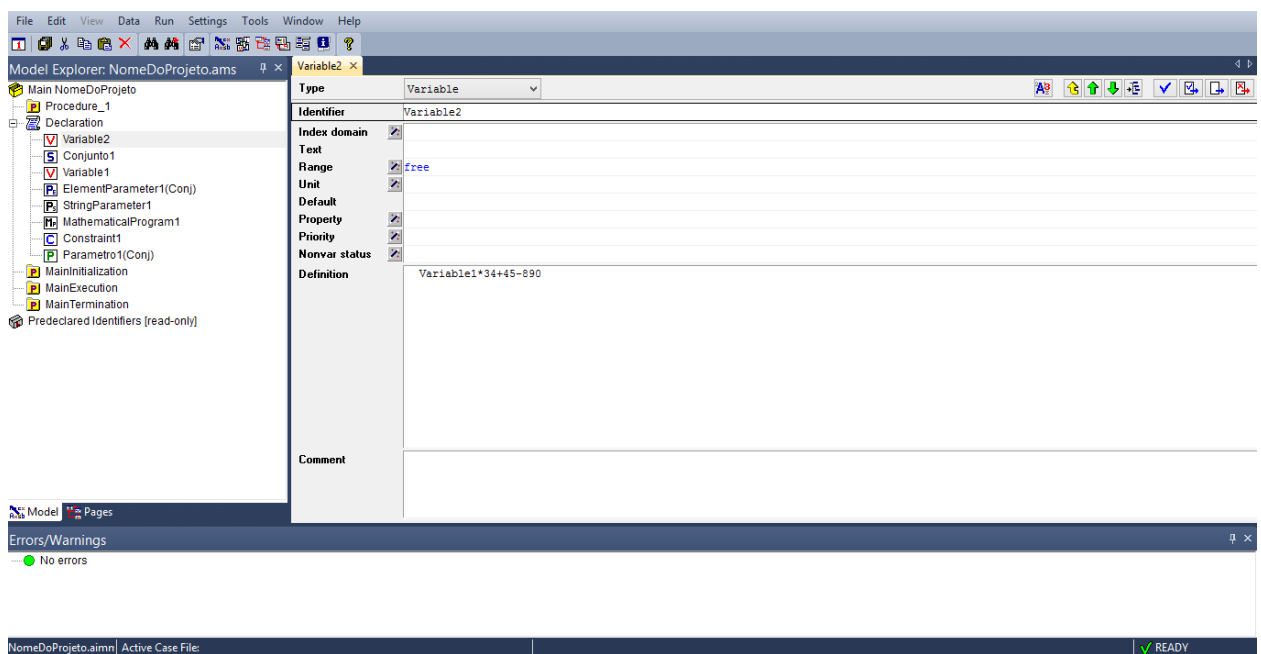


Figura 23

E então colocar no campo a variável2.

### 2.2.7.2 Direction

é se vamos maximizar ou minimizar.

### 2.2.7.3 Constraints

é onde vamos colocar as restrições que serão levadas em conta na otimização. Aqui como serão todas então basta colocar *AllConstraints*. Se tiver mais de 1 e quiser selecionar quais colocar, então temos que criar um conjunto como *subset* de *AllIdentifiers* e selecionar quais queremos, por exemplo:

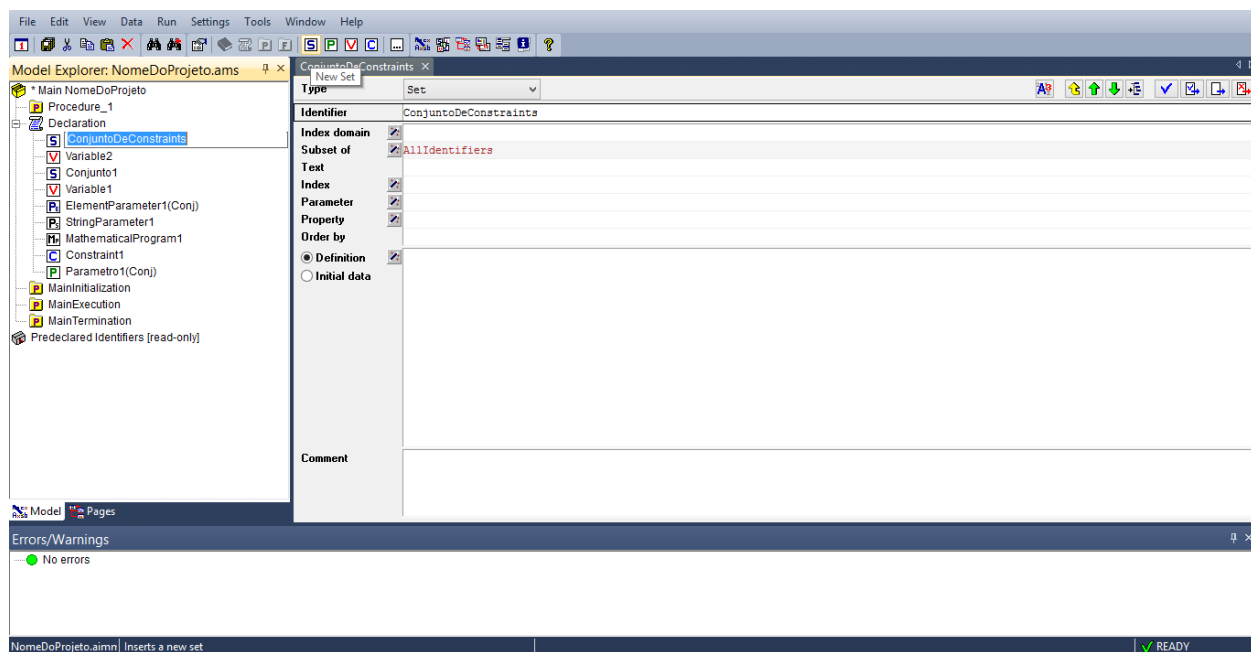



Figura 24

Clicar em  para escolher, exemplo:

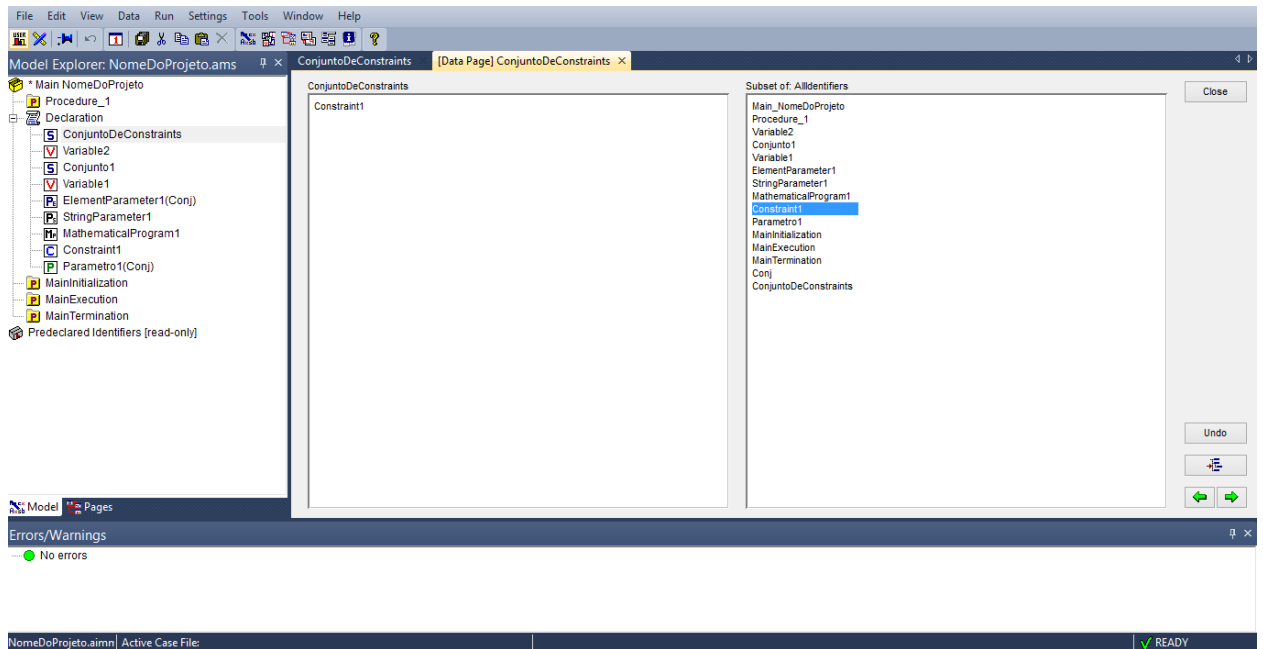


Figura 25

Ao fazer isso:

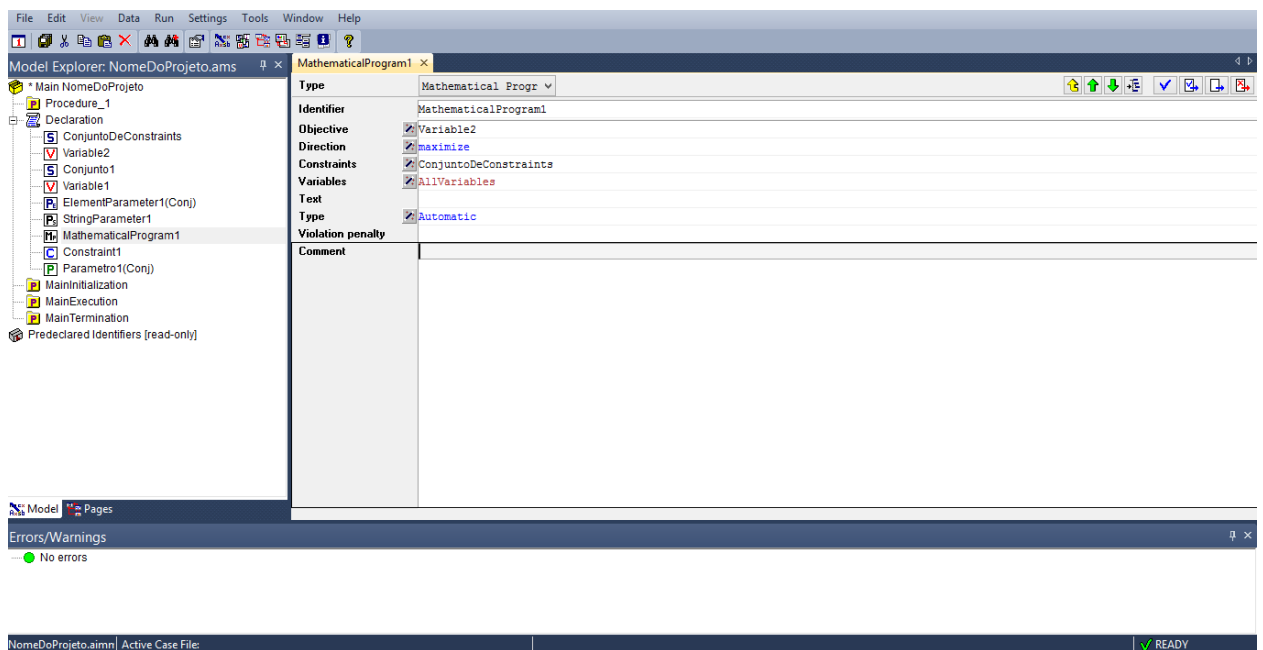


Figura 26

### 2.2.7.4 Variables

como *constraints*, é onde vamos definir quais variáveis irão ser consideradas na otimização, como no nosso caso é todas, então *AllVariables*, caso contrário basta fazermos como em *constraints* mudando o nome do conjunto e selecionando as variáveis ao invés das restrições

## 3 Procedimento (*Procedure*)

É onde vamos escrever linha de código atribuindo valores a parâmetros, alterando conjunto, fazendo cálculos e etc.

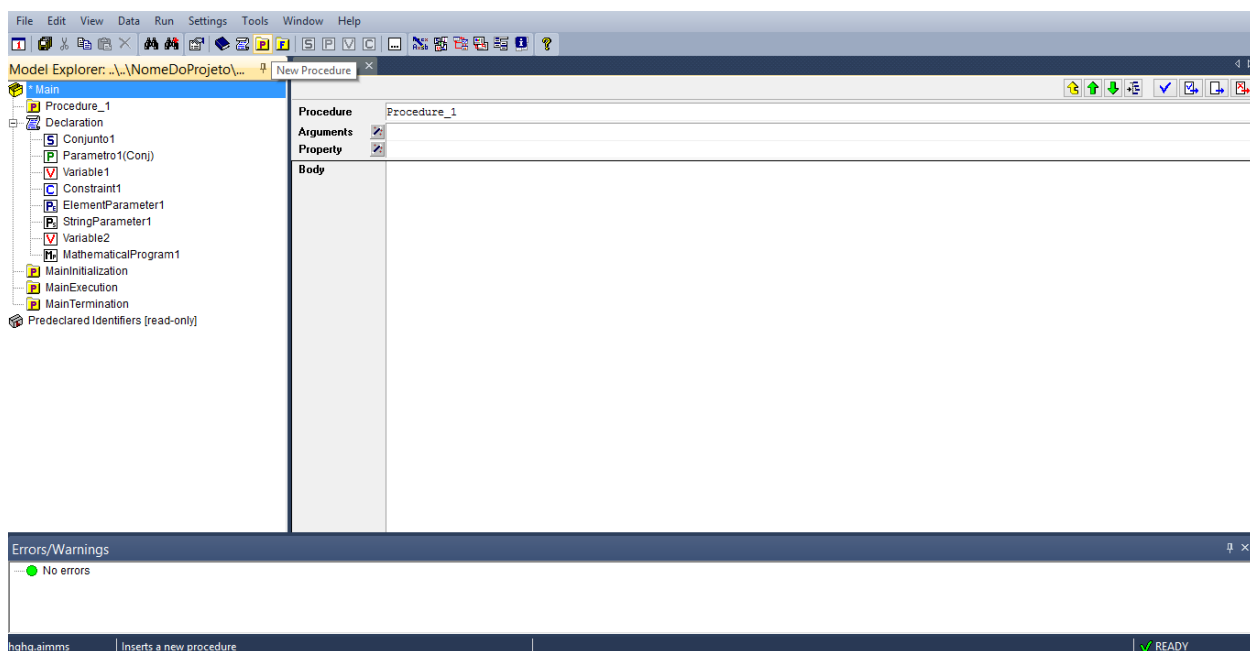


Figura 27

Sempre lembrando que para adicionar um procedimento, devemos clicar na biblioteca, no caso o *Main*, fora do *declaration*.

### 3.1 Body

é o campo em que será escrito o código. Aqui podemos usar as funções como *for*, *while*, entre outras que serão vistas mais a baixo.

Por exemplo:

For( Conj1) Do

```
Parameter1(Conj1):=5*Variable2+45;
```

```
Endfor;
```

Nesta pequena parte atribuímos valores ao parâmetro1. Esse procedimento deverá ser rodado após a otimização já que estamos utilizando a *variable1*, mas poderíamos substituir por outro valor e ser rodado antes.

### 3.2 Identificadores locais

são identificadores que só assumirão seus papéis (valores, elementos,..) enquanto o procedimento estiver sendo executado. Ou seja, eles não tem memória. Para criar um identificador local, basta clicar duas vezes em cima da figura da pasta da procedure no *model explorer*, dentro dele você precisa criar um *declaration* e dentro do *declaration* adicione os identificadores. Lembrando que esses identificadores só poderão ser usados dentro do *procedure*. Para que fique melhor a visualização, na hora de escrever o nome do identificador, coloque o "LOC\_", assim será mais fácil reconhecer o que é local ou não.

### 4 Realizando a otimização

Após ver procedimento e função matemática, vamos rodar a otimização, para fazer isso vamos declarar um procedimento e dentro desse procedimento “Solve “nome da função matemática””, no nosso caso, “Solve *MathematicalProgram1*”. Desse modo:

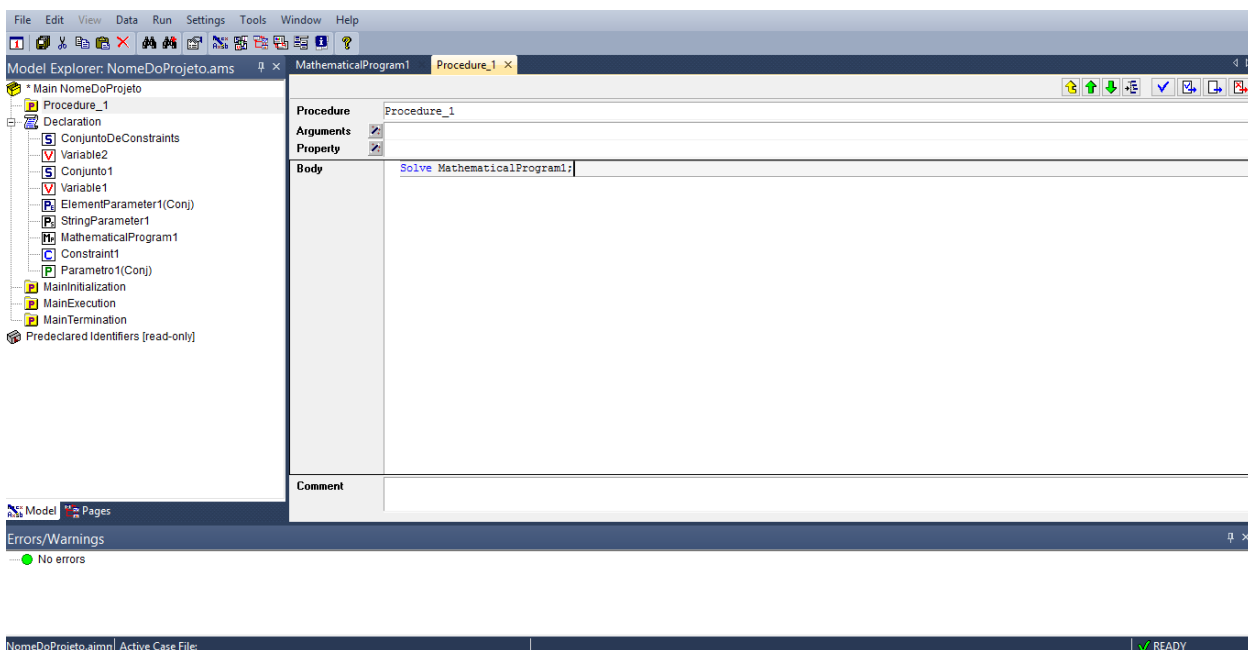


Figura 28

De *check*  e para rodar vamos clicar com o botão direito em *Procedure\_1* e escolher *run procedure*.

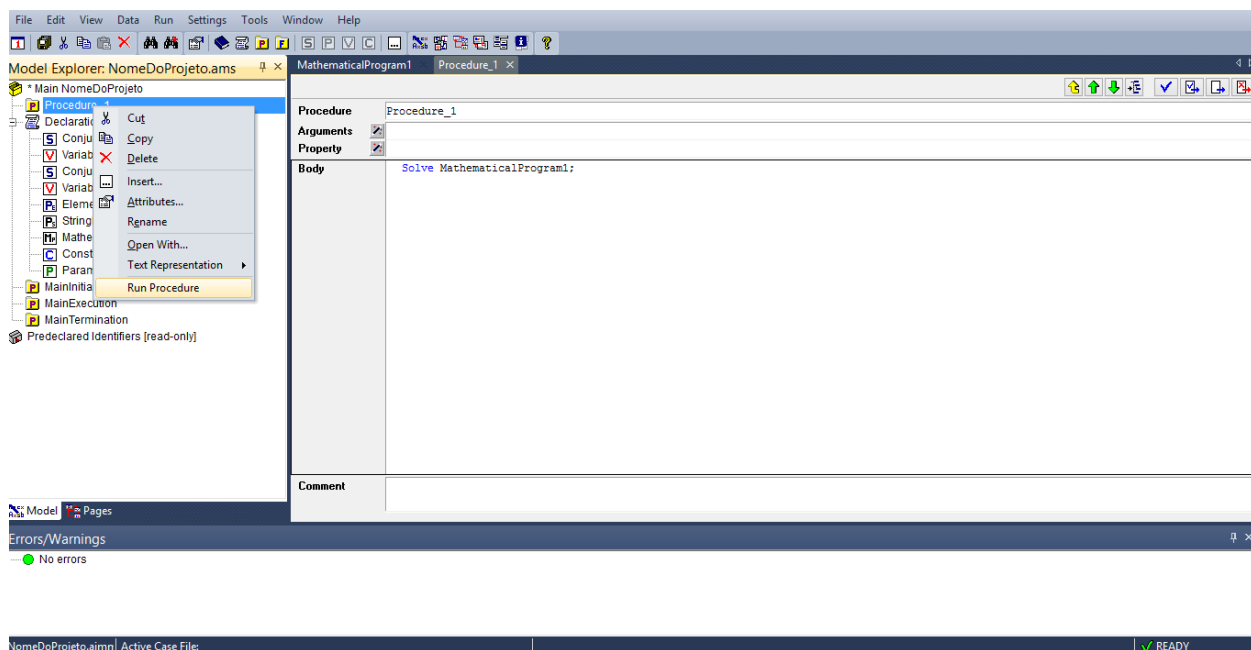


Figura 29

E assim estará otimizado, agora basta ver o valor da *variable1*, que foi otimizada.

## 5 Função (*Function*)

Aqui podemos criar uma função que poderá ser usada dentro de um procedimento.

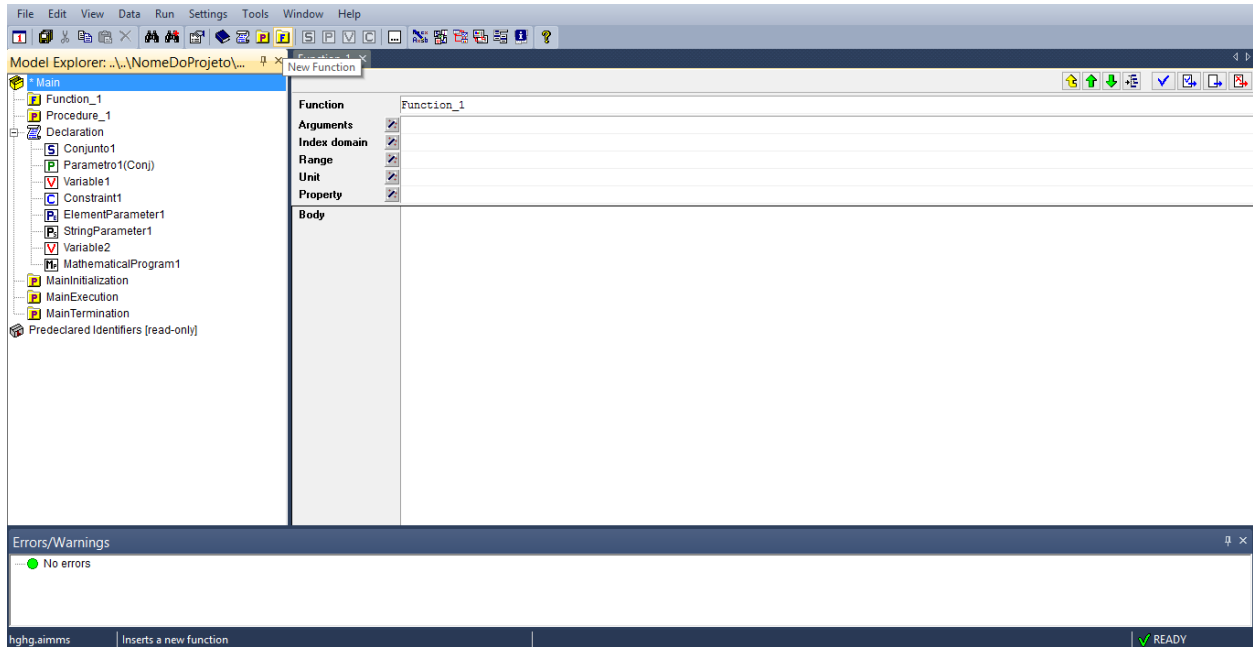


Figura 30

### 5.1 Arguments

são os argumentos que deverão ser passados para a função quando ela for chamada, por exemplo : (tes1,tes2).

### 5.2 Index Domain

e o campo que podemos colocar um ou mais índices e assim definir uma função para cada índice.

### 5.3 Range

é o domínio do valor que a função pode retornar, podendo ser  $\{0, \text{inf}\}$  ou  $\{-\text{inf}, \text{inf}\}$  e etc.

### 5.4 Unit

é a unidade do retorno da função, pode ser kg ou M, etc.

### 5.5 Body

é o campo em que vamos escrever a função.

Por exemplo :

For(Tes1,Tes2)

If(Tes1>=Tes2) Then

```
        Return((Tes1)
Else
        Return(Tes2)
Endif;
Endfor;
```

Essa função, dada a entrada de argumentos tes1 e tes2, ela retorna o maior número. Na chamada da função basta atribuir ela a um parametro2 por exemplo : Parametro2:=Function1(8,9); Isso irá retornar 9.

## 6 Declaration

Sempre que adicionamos uma *section* nova, precisamos adicionar pelo menos um *declaration*, pois só dentro dele podemos adicionar os identificadores. Já o procedimento e a função só podem ser adicionados fora dele. É muito usado para organização e separação de identificadores por grupos.

## 7 Section

É usada para organização, as *declarations* só podem ser adicionadas dentro dele. Preferencialmente usamos varias *sections* para separar grupo de funcionalidades dentro do modulo (programa).

## 8 Principais funções usadas

Lembrando que todas essas funções podem ser encontradas detalhadas no help:



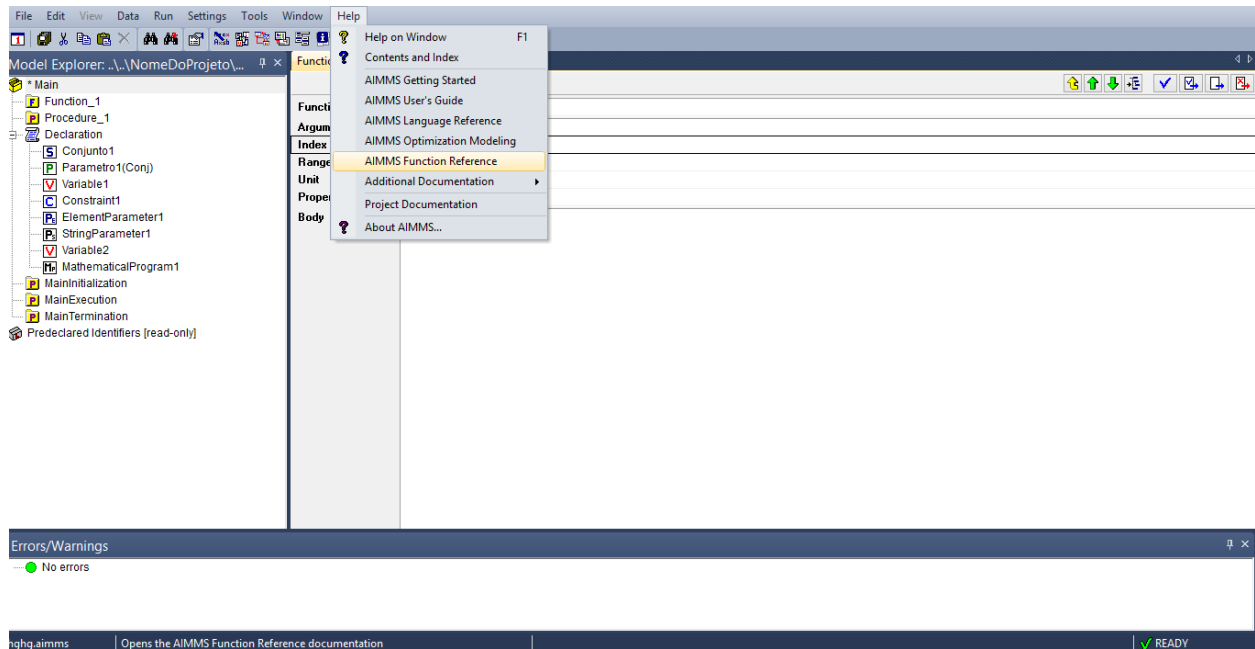


Figura 31

### 8.1 *StringToElement*

*StringToElement* (Conjunto, nome, criar?) serve sempre que temos o nome de um elemento de um conjunto mas não temos o apontador para ele, então basta colocar o nome do conjunto referenciado, o nome do elemento e se o elemento não existir no conjunto se queremos adicioná-lo ao conjunto ( 0 ou 1). Essa função retornará um *Element Parameter* do conjunto, logo sempre que chamada tem ser atribuída a um elemento *parameter*.

### 8.2 *Element*

*Element*(Conjunto,n) retorna o n-ésimo elemento do conjunto

### 8.3 *StringToUpper*

*StringToUpper*(Texto) retorna o texto todo em maiúsculo.

### 8.4 *Val*

*Val*(*ElementParameter1*) é uma função que dada algum *element parameter* que está atribuído em um valor, ela retorna o valor dele, logo quando chamada precisa estar atribuída a um parâmetro de valor.

### 8.6 Return

*Return*( algo) é usada para função ou mesmo para procedimento, se executada ela sai da função ou procedimento e retorna o que estiver dentro do parênteses .

### 8.7 For

*For*(Conj) *Do* (.....) *Endfor*; é usado sempre que queremos fazer um loop de algo percorrendo os elementos de um Conjunto com o índice conj, “Para tal elemento faça”. O que irá repetir é o que estará dentro do (..)

### 8.8 While

*While* ( Algo ) *Do* (..) *Endwhile*; é usado para um loop também, mas não referente a um conjunto passando o índice como elemento como no for. Aqui é “Enquanto algo existir repita”, então dentro do *while* precisa ter algo atualizando esse “Algo” senão entraremos em um loop infinito. O “Algo ” pode ser um parâmetro, um conjunto etc..

### 8.9 If

*If*(Algo) *Then* (..) *Endif*; quer dizer “Se “Algo” é verdade então ” é usado para filtrar, ou seja, se “ Algo “ é verdade então ele irá realizar os comandos que estarão entre o *then* e o *endif*; Senão acontecer então ele não realiza. Podemos complementar a função com *Else* ou *IfElse*, que ficaria *If*(“Algo”) *Then* (..) *Else* (..) *Endif*;

*Else* é usado para quando não satisfizer o *if*, isso quer dizer que quando “Algo” for falso então realizará os comandos do *else*.

### 8.10 First / Last

*First*(Conjunto)/*Last*(Conjunto) retorna um *element parameter* o primeiro ou ultimo elemento do conjunto.

### 8.11 SetElementAdd

*SetElementAdd* (Conjunto, apontador do conjunto, elemento a ser adicionado) aqui adicionaremos um elemento ao conjunto, então precisamos passar para a função o conjunto que será adicionado, um elemento *parameter* que retornará o elemento adicionado ao conjunto e o elemento que queremos adicionar em numero ou letras. Retornará 1 se sucesso 0 caso contrário.

### 8.12 Somatória

*Sum*((Conj),(Expressao(conj))); é a função da soma, vamos somar em Conj uma expressão dependente de conj.

Por exemplo somamos *sum*(elemento,*Peso*(elemento)); então ele soma para cada elemento o peso. Assim , por exemplo, *Peso*(caixa)+*Peso*(balde)+..

## 9 Dicas

### 9.1 Completando argumentos

Para isso basta escrever o nome da função e clicar ao mesmo tempo **Ctrl+Shift+Espaço** duas vezes seguidas. Isso completará com os argumentos *Default*, depois vc deverá colocar os apropriados por você.

### 9.2 Compilar Modelo

Sempre que puder compile o modelo clicando em **F5**, pois as vezes salvamos, mas ele nem sempre atualizou certo internamente.

## 10 Debugger

Sempre que queremos rodar um procedimento por partes para encontrar algum erro ou observar o que está acontecendo, principalmente quando temos identificadores locais, usamos o debugger. Então seja o procedimento exemplo:

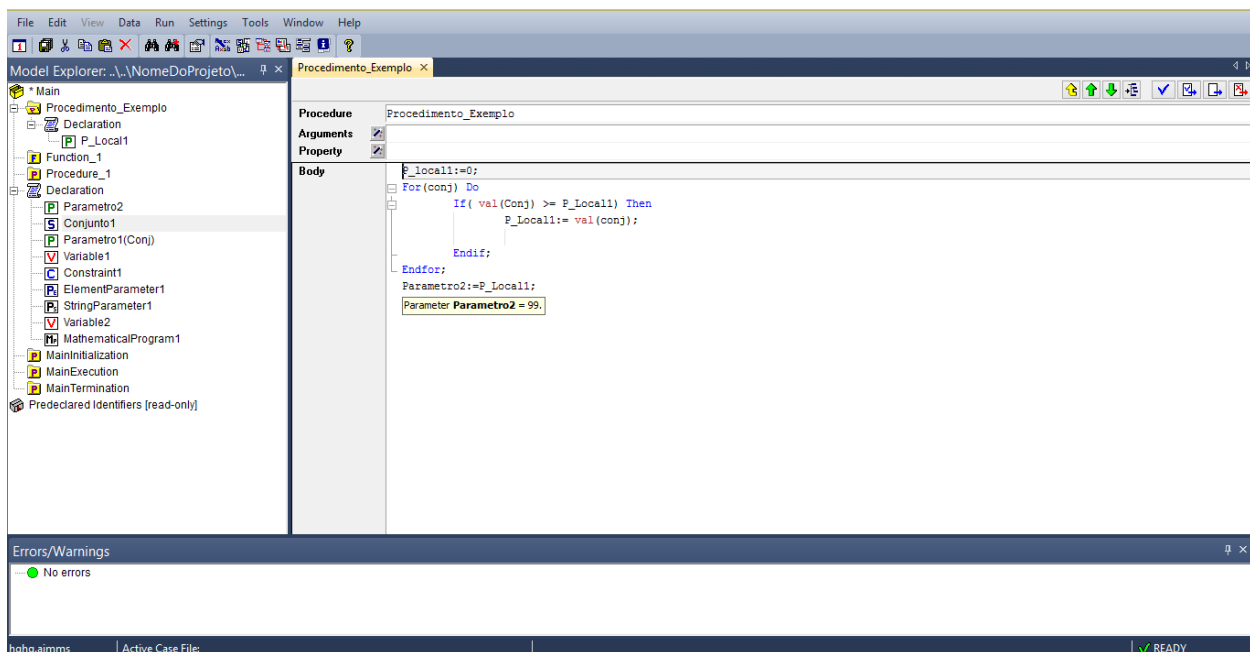


Figura 32

Esse procedimento preenche Parametro2 com o maior numero do conjunto. Se declarmos o conjunto com (45,5,23,1,57,8,99,6,72) por exemplo, vamos ver  $\text{parametro2} = 99$ , mas e o  $\text{p\_local1}$  ? Se tentarmos abrir ele depois não aparecerá nada. No modo debugger podemos ver os valores sendo adicionados a ele. Então vamos abrir o modo debugger:

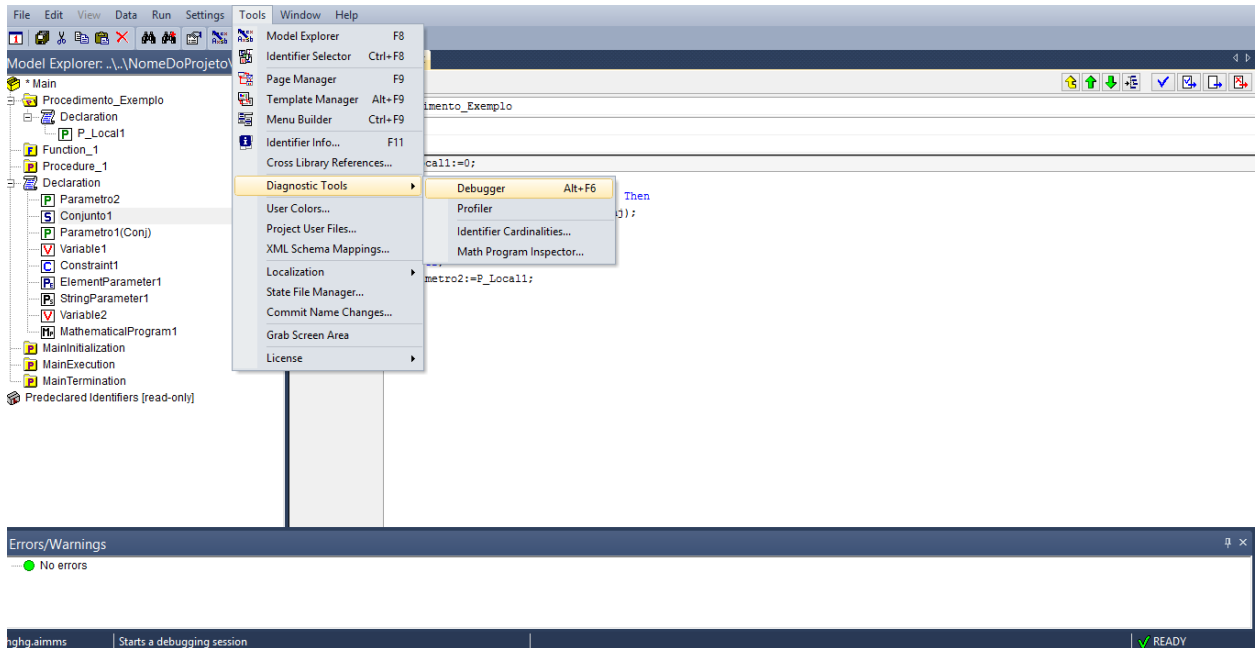


Figura 33

Ficará como a figura abaixo:

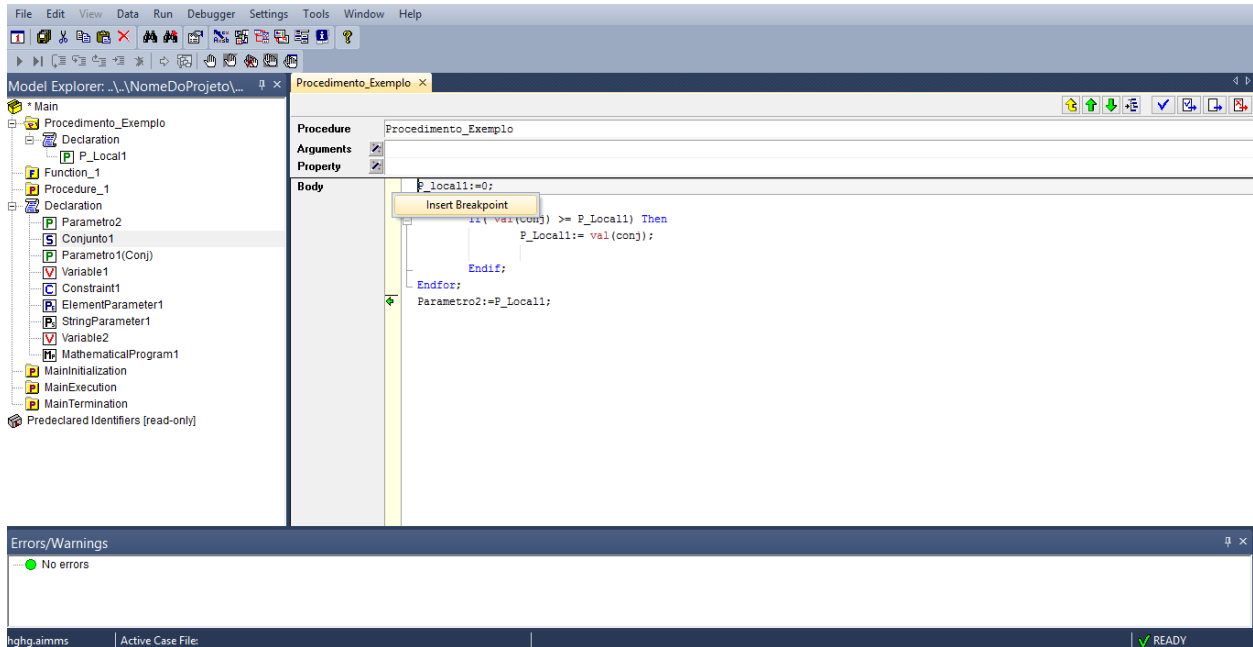


Figura 34

Para adicionar o ponto de parada basta o local que quer adicioná-lo, clicar com o botão direito e coloca-lo, como na figura. Coloque no começo e vamos rodar o procedimento.

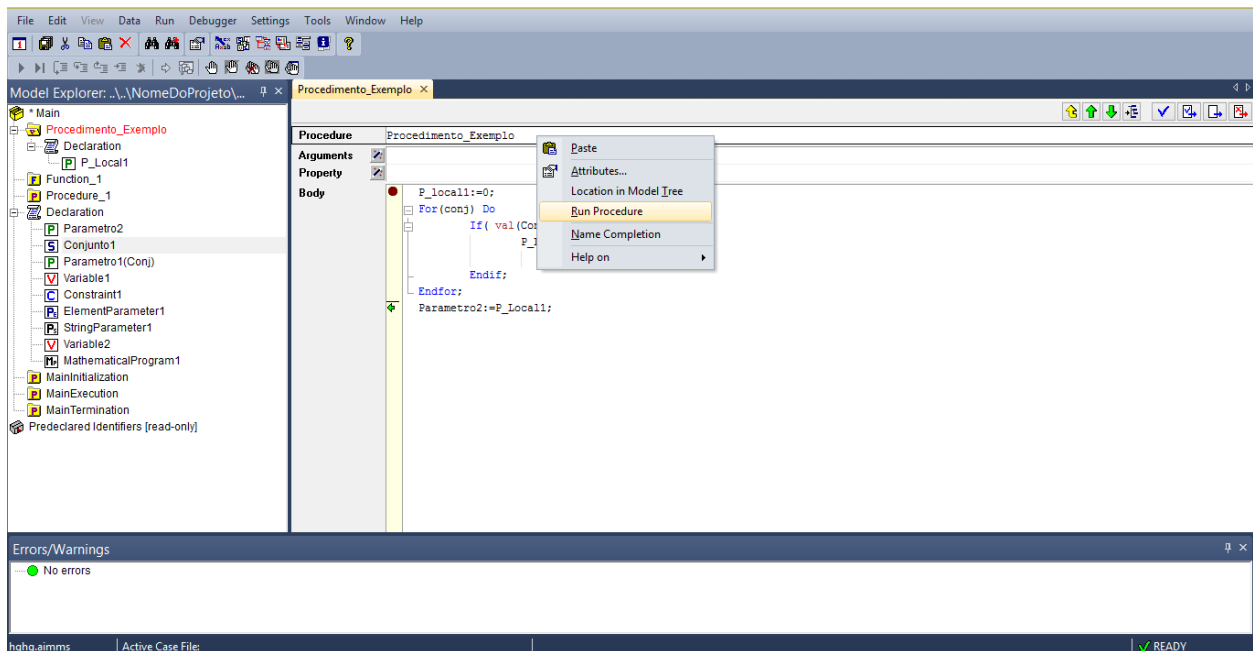
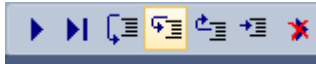


Figura 35

Agora vamos avançar um a um por essa barra  o primeiro item irá avançar até o próximo breakpoint, o segundo até o final do procedimento, o terceiro

de 1 em 1, o quinto irá voltar e o ultimo parara o debugger, lembrando que para fechar o debugger é só em :

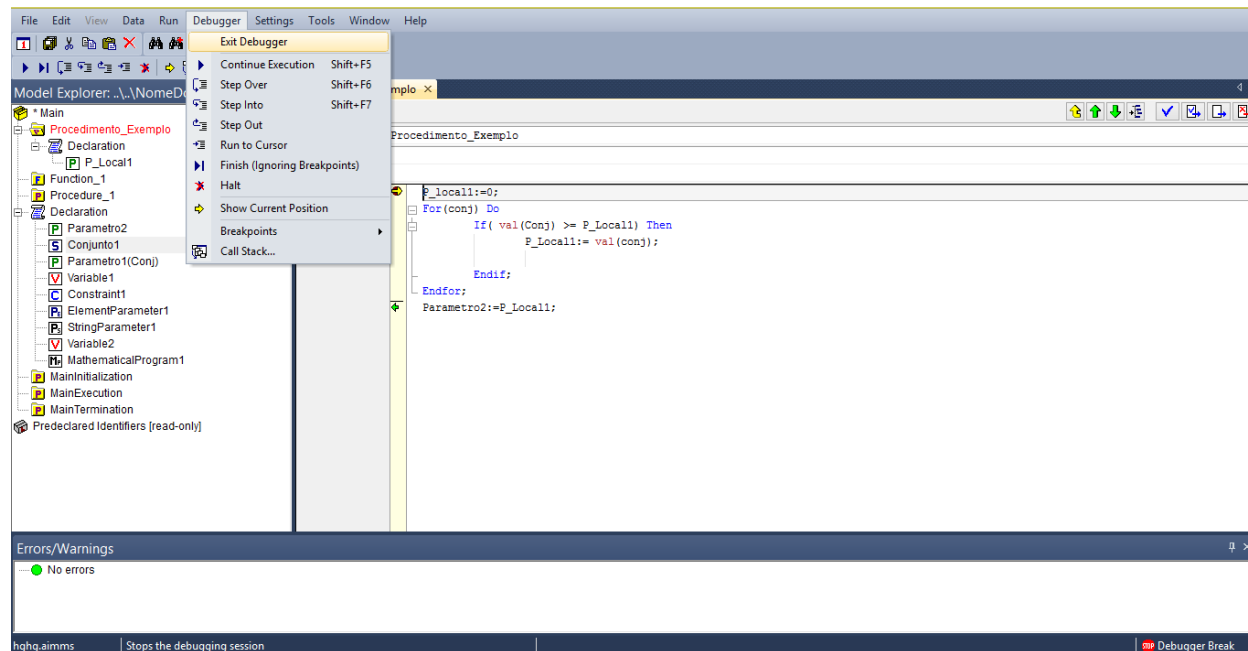


Figura 36

Vamos voltar ao procedimento, avançando de 1 em um vamos ver os valor atribuídos a variável local, se entra no if ou não, etc.

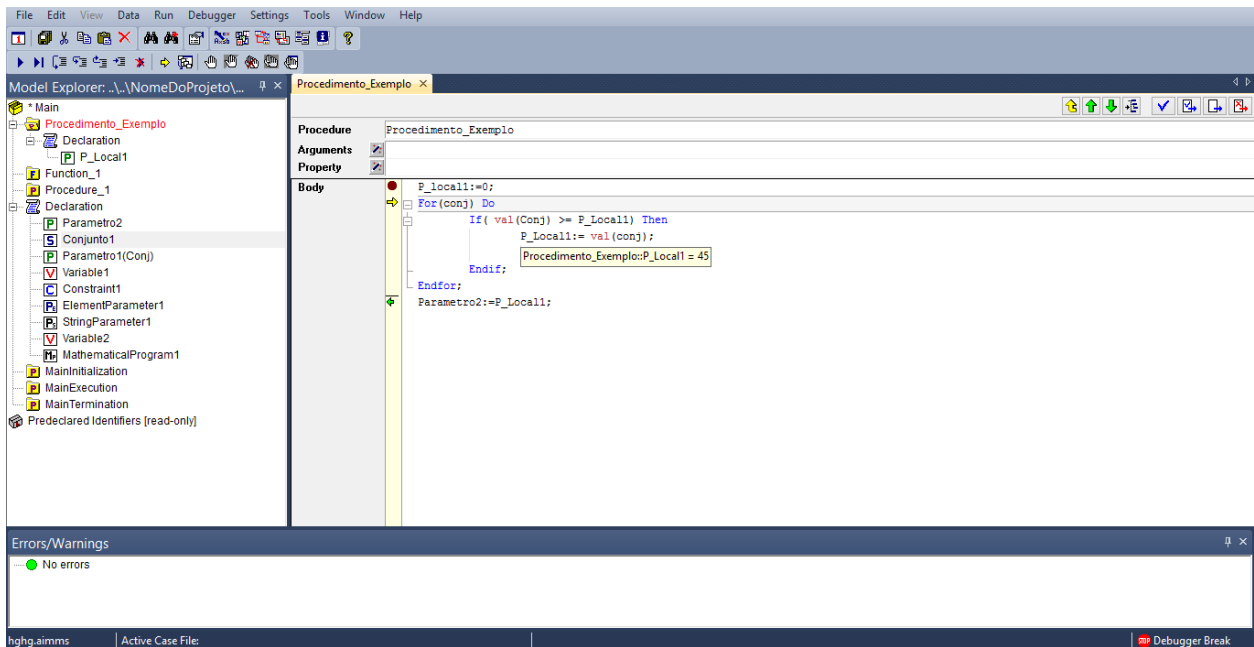


Figura 37

## 11 Problema da mochila

Suponha que temos uma mochila e precisamos colocar coisas dentro dela para levar, mas ela tem um espaço limitado, logo o que levar ? Então a modelagem fica :

Seja  $x_i$  a quantidade do elemento  $i$  que será colocado na mochila ,  $b$  o peso máximo que a mochila aguenta,  $c_i$  o valor ( de importância) de cada elemento  $i$ ,  $k_i$  o peso do elemento  $i$ , vamos adicionar uma restrição que diz que a quantidade máxima que pode ser levado de um elemento é 6. Logo:

$$\text{Max } \sum_i c_i * x_i$$

$$\text{s.a. } \sum_i k_i * x_i \leq b$$

$$x_i \leq 6 \text{ para todo } i$$

Vamos fazer o desenvolvimento do problema:

Então começando declarando os *identifiers*:

**Set** – Objetos\_da\_mochila

*Index:* Obj

**Variavel** – Quantidade\_de\_cada\_objeto

*Index:* domain Obj

*Range:* Integer

**Parametro** – Coeficiente\_Peso\_Objeto

*Index:* domain Obj

*Range:* Nonegative

**Parametro** – Coeficiente\_Valor\_Objeto

*Index:* domain Obj

*Range:* Nonegative

**Parametro** - Capacidade\_Peso \_Mochila

**Constraint** - Numero\_Max\_Objeto

*Index Domain:* Obj

*Body:* Quantidadede\_cada\_objeto(Obj)<=6;

**Constraint** - Capacidade\_Mochila

*Body* : sum(Obj,Quantidadede\_cada\_objeto(Obj)) \*

Coeficiente\_Peso\_Objeto(Obj) <= Capacidade\_Peso \_Mochila;

Adicionando outra variável para a função objetivo :

**Variable** Funcao\_objetivo

*Definition:* sum(obj, Quantidadede\_cada\_objeto(obj))\*

Coeficiente\_Valor\_Objeto

(obj))

E declarando a função matemática (MP):

*Mathematical Program:* OtimizacaoMochila

*Objective :* Funcao\_objetivo

*Direction:* Maximize

*Constraints:* AllConstraints

*Variables:* AllVariables

*Type:* Automatic



Vamos Construir uma procedure para solucionar o problema de otimização:

Procedure: Roda\_Otimizacao

Body: *Solve* OtimizacaoMochila;

Após construir todos os identificadores no AIMMS, teremos :

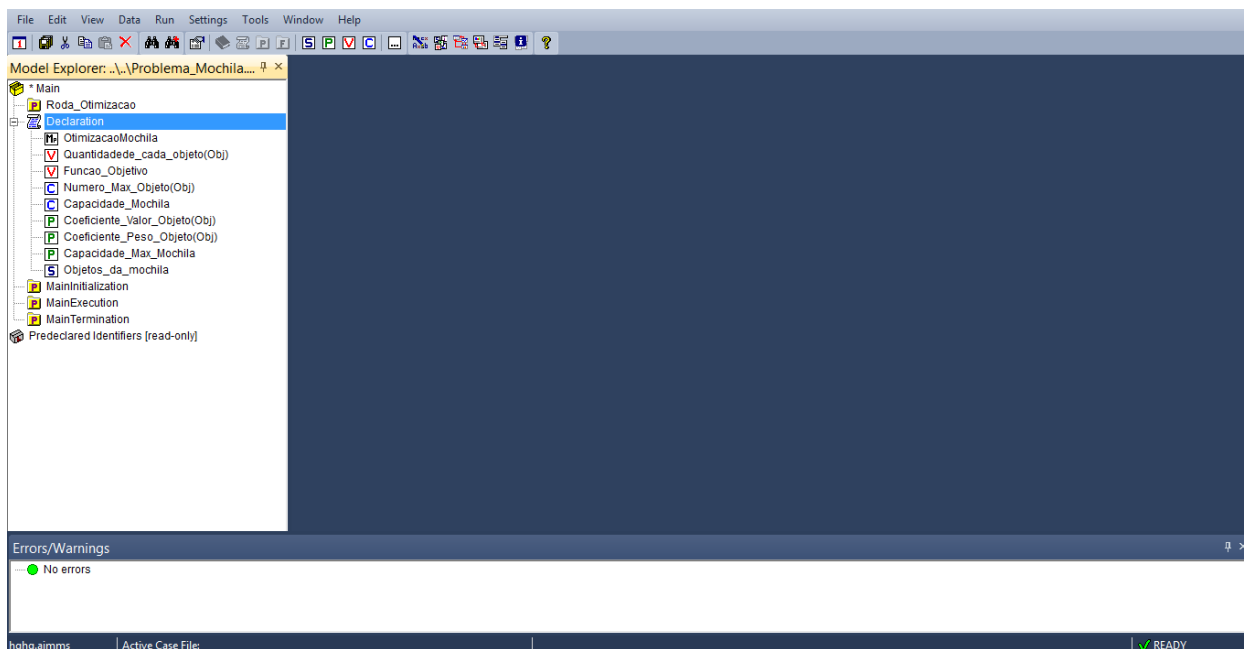



Figura 38

Lembrando que antes de rodar a otimização devemos preencher os dados. Logo vamos preencher os dados para testar a otimização:

Lembrando que para preencher os dados, ou verifica-los, basta abrir o identificador e clicar em data .

Adicione a objetos\_da\_mochila : Água, Comida, Roupas, Produtos limpeza

Declare no parâmetro Coeficiente\_Peso\_Objeto : 45, 32, 30, 43 para os objetos acima respectivamente.

Declare no parâmetro Coeficiente\_valor\_Objeto : 60, 50, 20, 10

Declare no parâmetro Capacidade\_Peso \_Mochila : 150

E agora clique com botão direito em cima da procedure e dê “*Run Procedure*”.

Para ver os resultados da otimização vá na variável Quantidade\_cada\_objeto e veja seu valor.

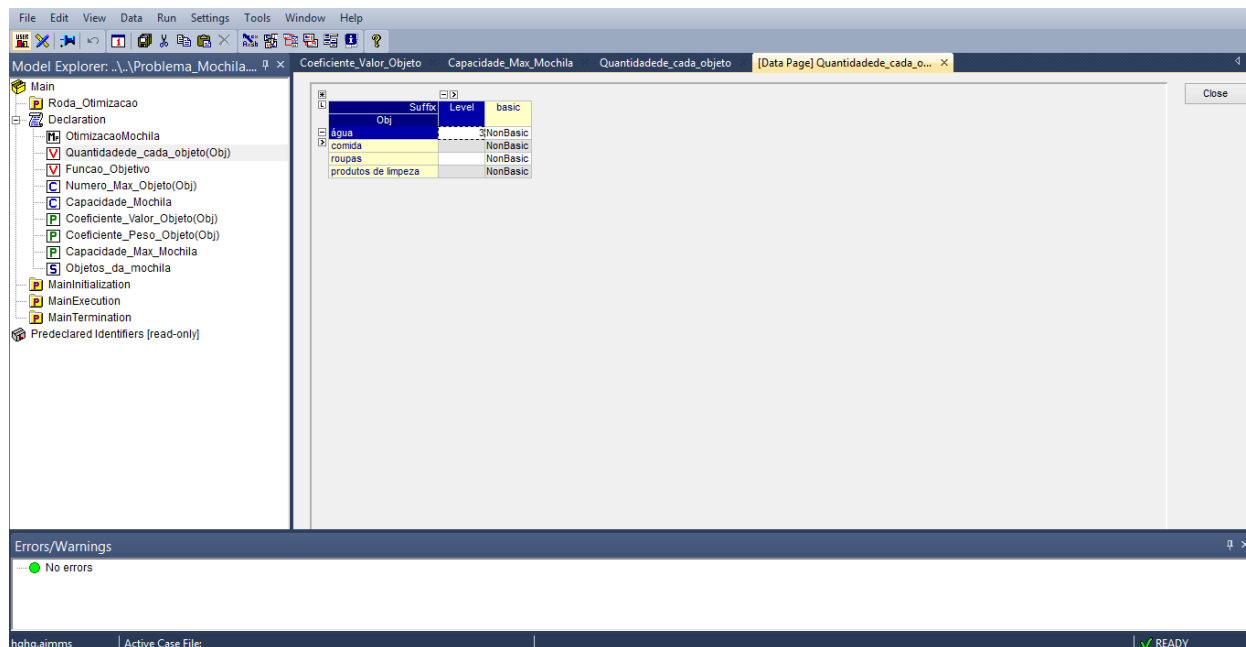


Figura 39

Suponhamos agora que não queremos preencher diretamente o parâmetro, mas queremos deixar para o usuário preencher em uma interface gráfica e verificar o resultado na mesma.

Logo vamos fazer:

Tabela para inserção de dados em

- objetos\_da\_mochila
- Coeficiente\_Peso\_Objeto
- Coeficiente\_Valor\_Objeto
- Capacidade\_Peso \_Mochila

Tabela para visualização de dados em

- Quantidade cada objeto

E um botão para rodar o procedimento

- Roda\_Otimizacao

## 12 Interface Gráfica

É a parte do AIMMS que nos possibilita fornecer ao cliente uma visualização do programa. Aqui podemos fazer telas, botões e barras de ferramentas para que usuário

possa se conectar ao modulo, para que possa preencher os dados, rodar o sistema e ver resultados. Para isso vamos abrir o *page manager*.

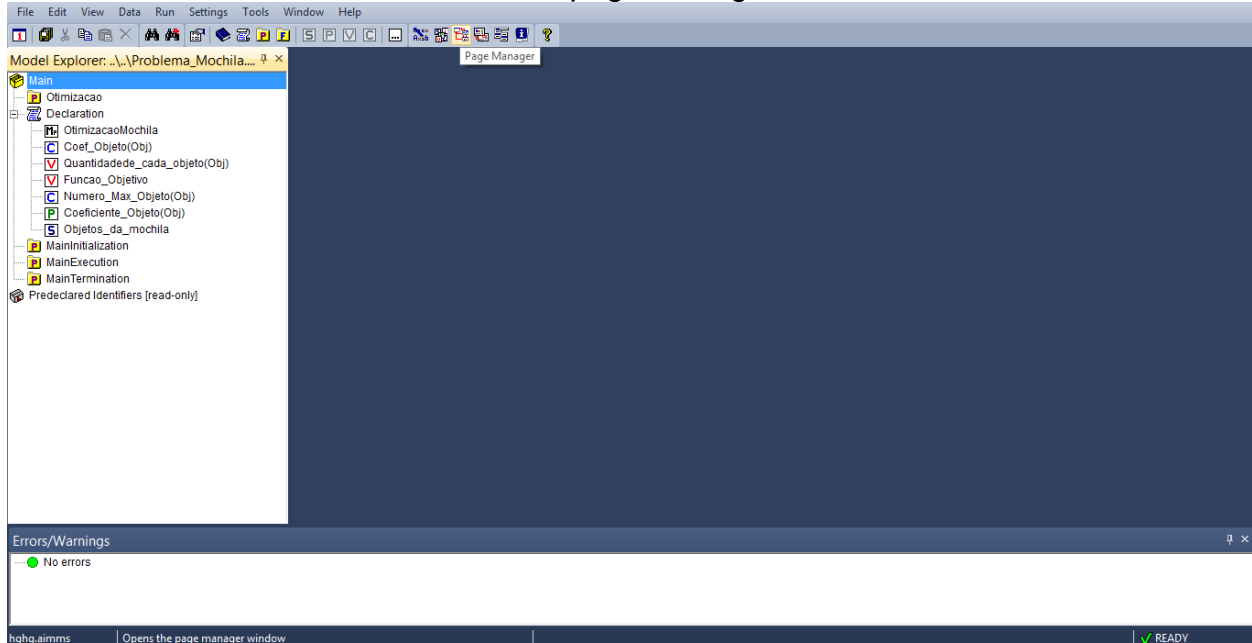


Figura 40

Note que a barra irá ficar em cima do *model explorer* quando for aberta se a barra de erros/*warnings* estiver aberta. Para isso basta fechar a barra de erros e verá que em baixo ficara a opção para alternar entre o *model explorer* e o *page manager*.

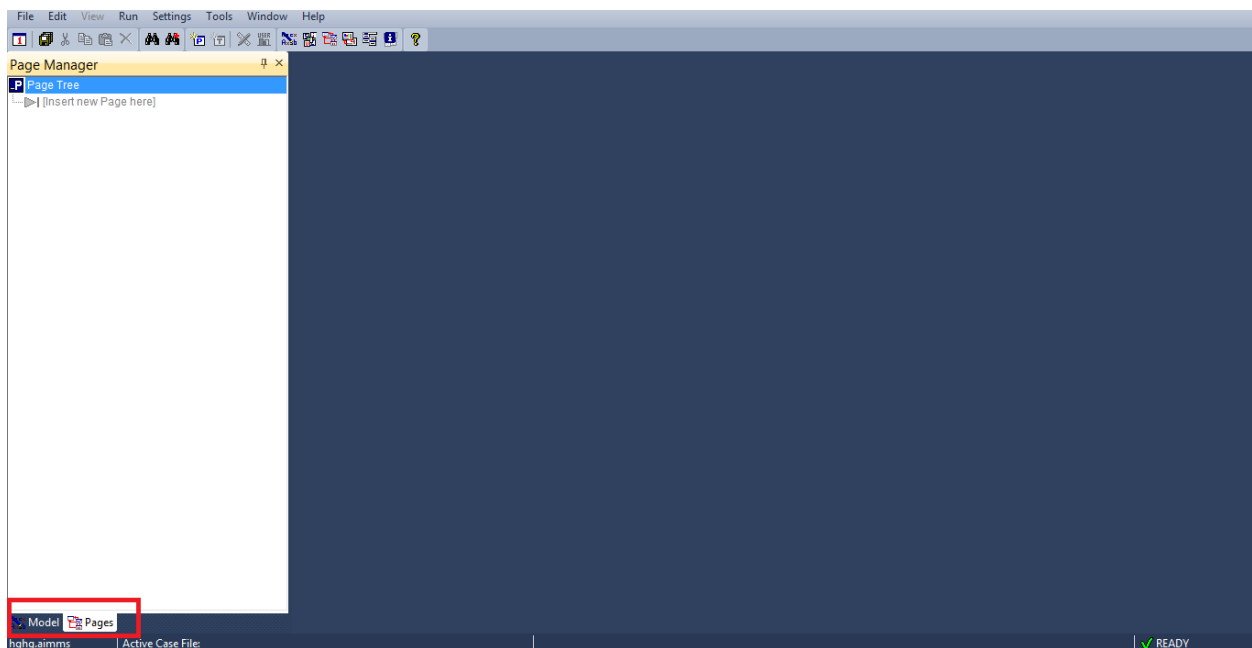


Figura 41

## 12.1 Criando nova página

Selecione *Insert new page here* e clique no P na barra de ferramentas :

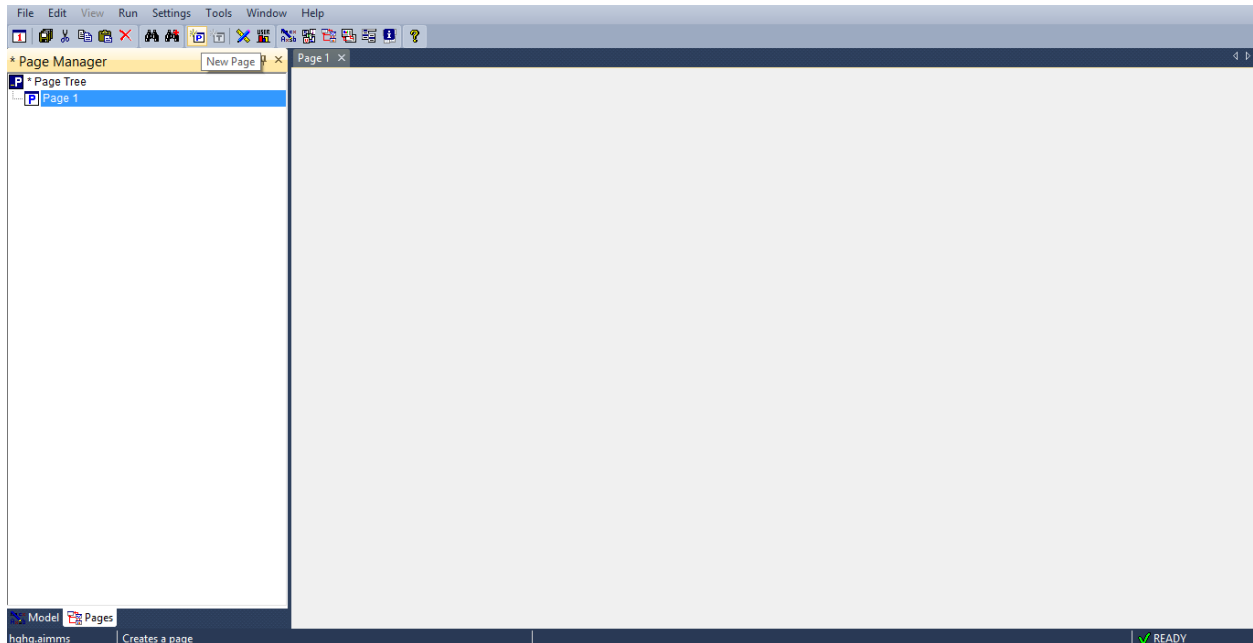


Figura 42

Como nesse tutorial só será apresentado o básico, você poderá ver mais detalhes no manual do AIMMS como descrito no começo.

Para começar a editar a página, precisamos clicar F4, isso destravará a pagina, para travá-la basta clicar em F4 novamente.

## 12.2 Inserindo texto

Para isso basta ir em *object – text*

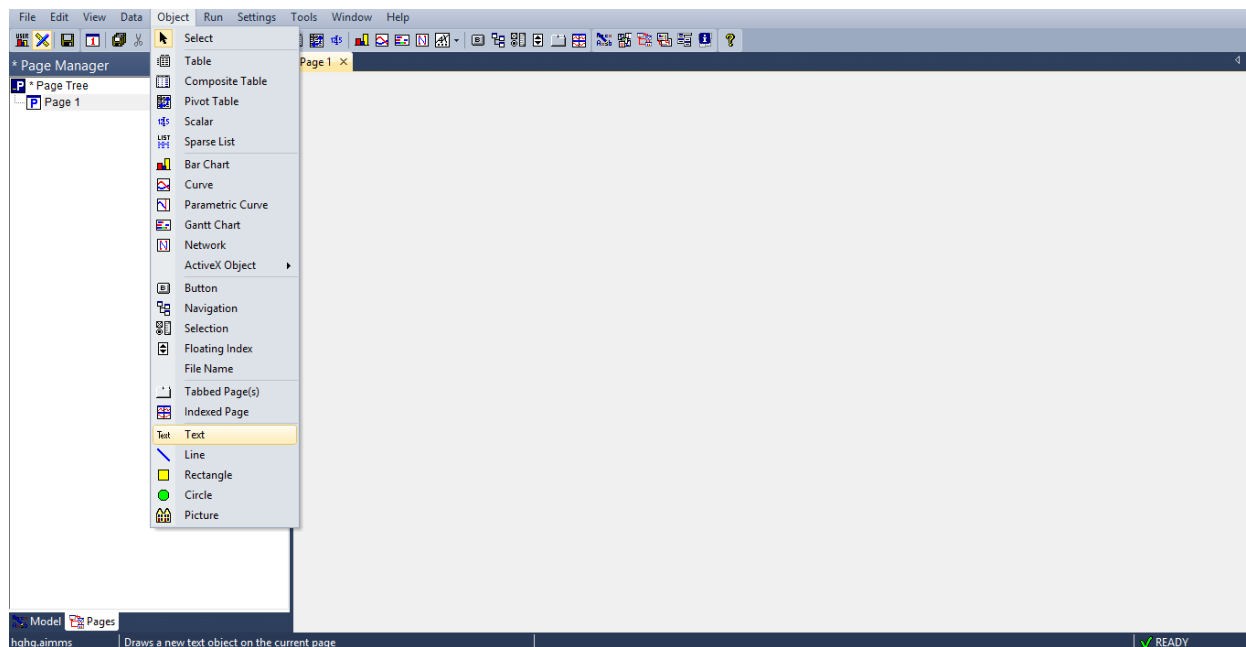


Figura 43

Selecionar onde queremos colocar ele e escrever. Também poderão ser feitas personalizações de cores e etc nas outras abas.

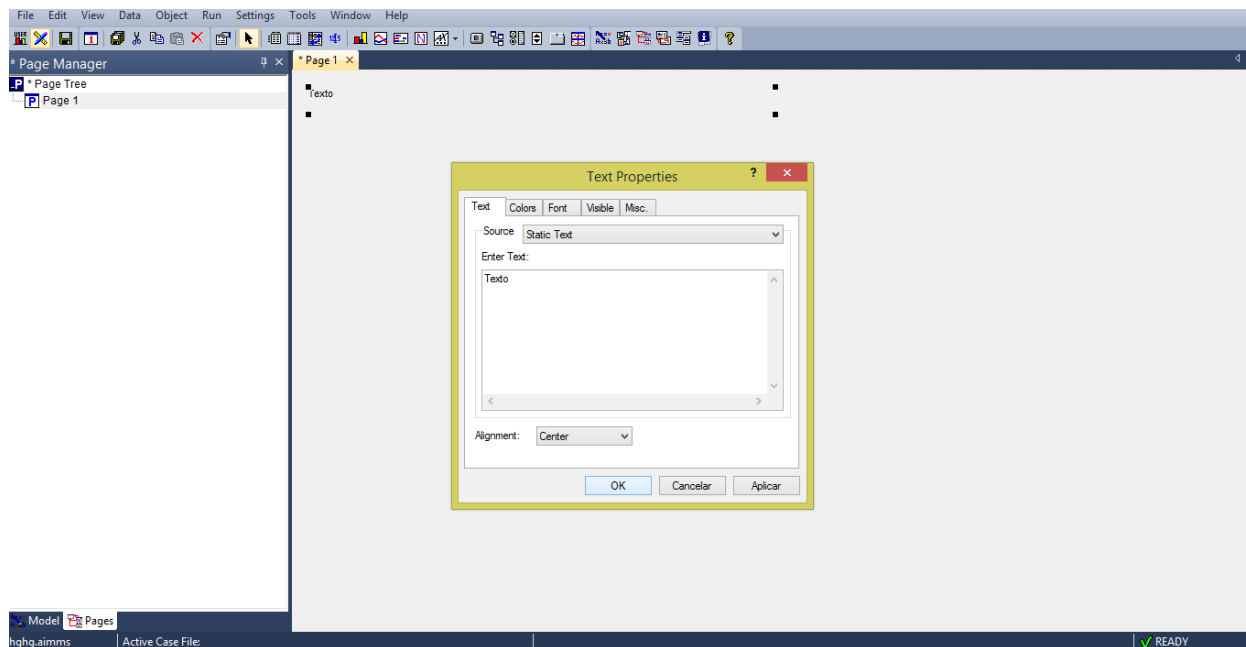


Figura 44

## 12.3 Inserindo tabela

Adicionando tabela para inserção de dados ou só visualização pelo usuário em um parâmetro.

A diferença de inserção de dados ou só visualização pode ser programada nas configurações da tabela, mas por *default*, a tabela poderá ser editada se o parâmetro não for definido e caso contrário de for definido.

Para isso clique em Object – Pivot Table, selecione onde quer que ela seja colocada.

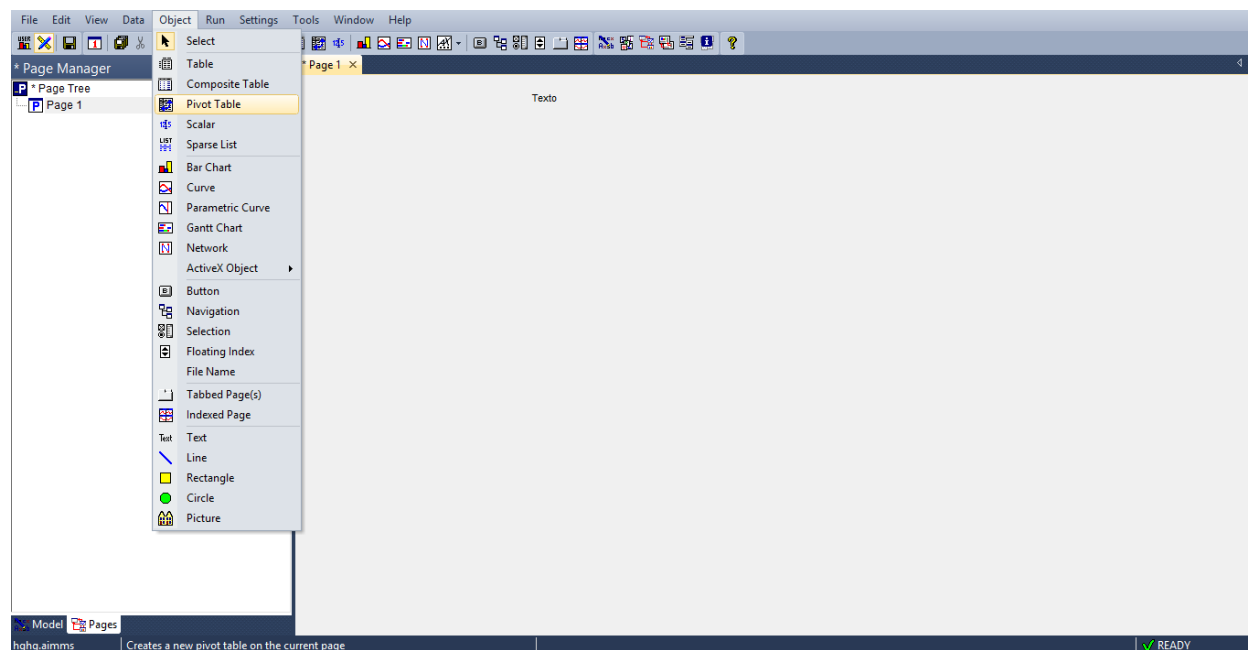


Figura 45

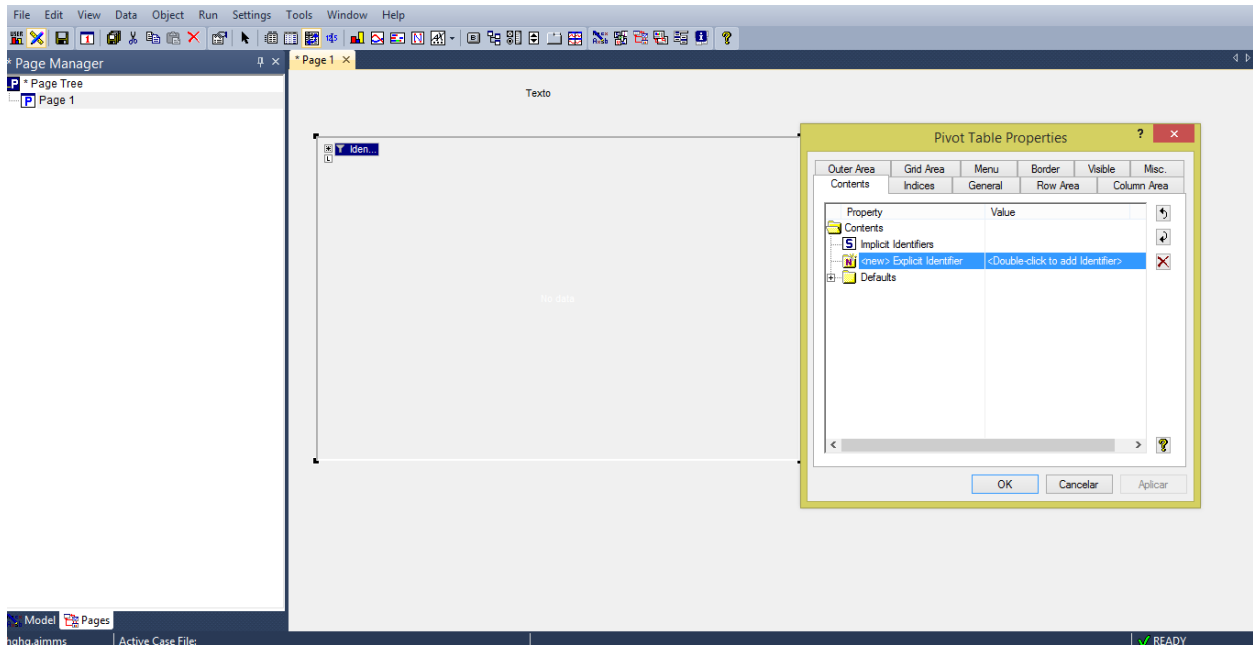


Figura 46

E em *Explicit identifier* colocaremos o parâmetro a ser mostrado (coeficiente\_valor\_objeto(obj)). Após isso aparece a próxima tela :

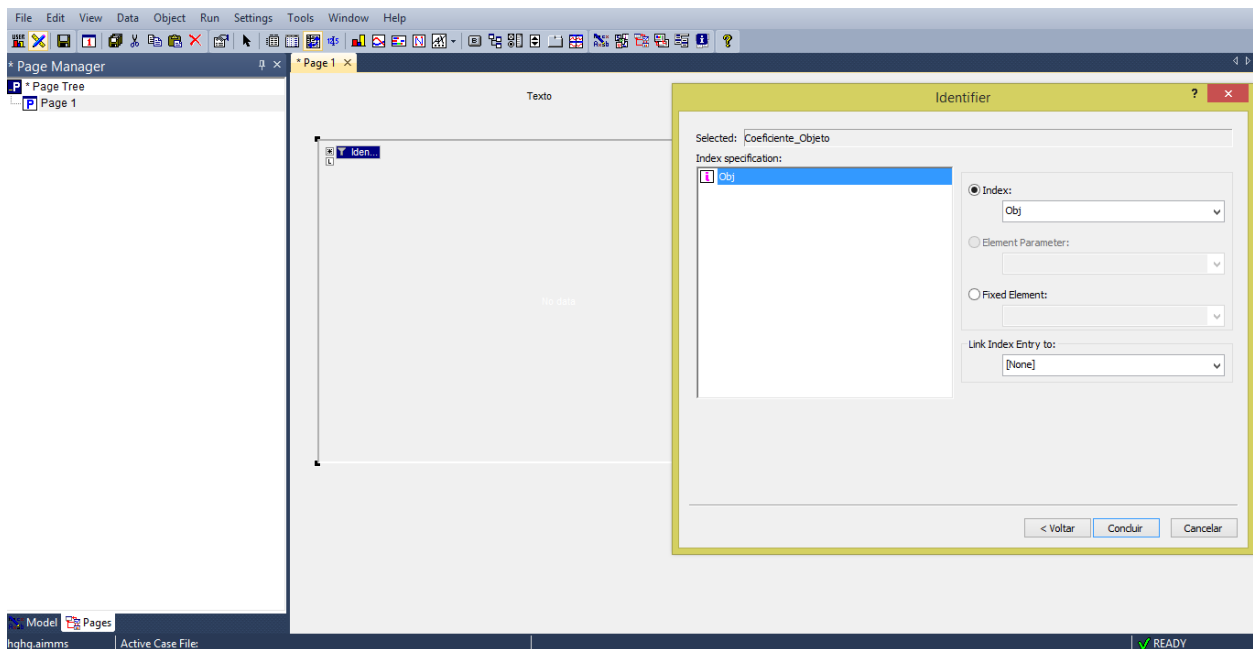


Figura 47

Lembre-se, aqui o conjunto de elementos da mochila (água, roupas, comida...) já está definido.

Aqui é importante observar que aparece os índices que o parâmetro está indexado. Podemos deixar o parâmetro mostrar valores para todo os elementos do conjunto se deixarmos o índice ou podemos atribuir ao índice um *element parameter* ou *fixed* elemento. Se for *element parameter*, então ele só mostrará o valor para o elemento apontando do *element parameter*. Note que aqui não temos nenhuma opção pois não existe *element parameter* associado ao conjunto.

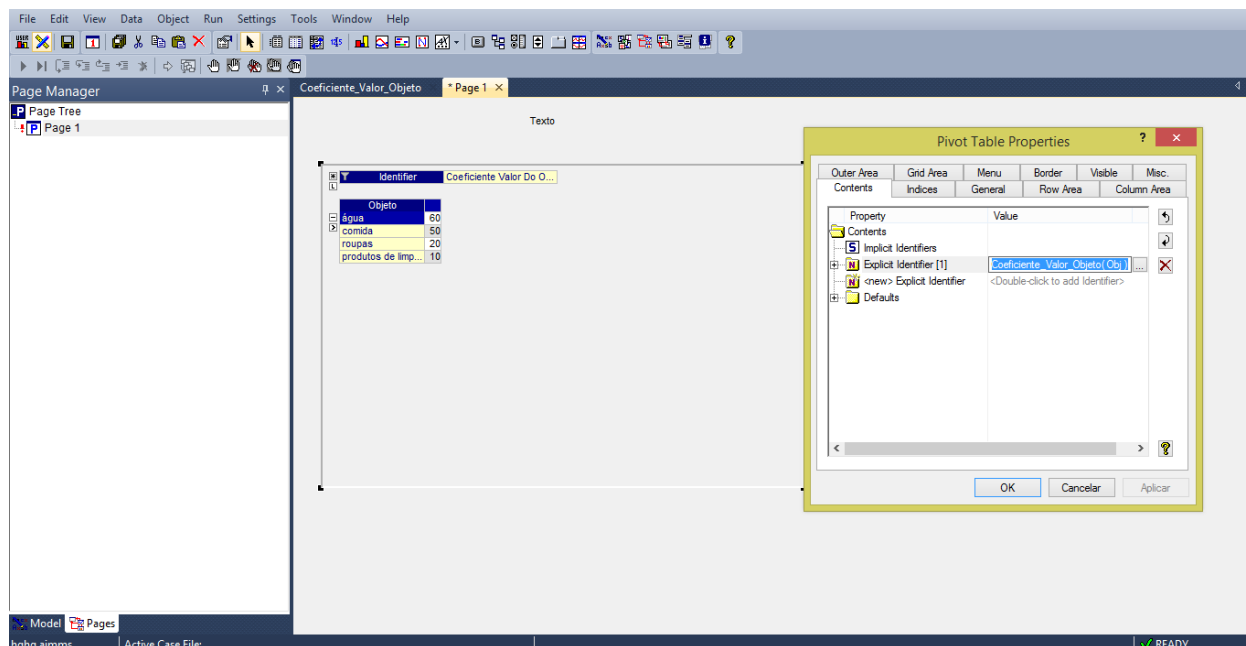


Figura 48

Ao voltar temos a opção de especificar o índice, isso nos possibilita da nome a ele na tabela, mudar cor, escolher a opção de esconder ou não valores da função que só dão 0 para um valor de índice especificado, etc.

Para especificar o índice, mude a aba para índices e coloque o índice obj :



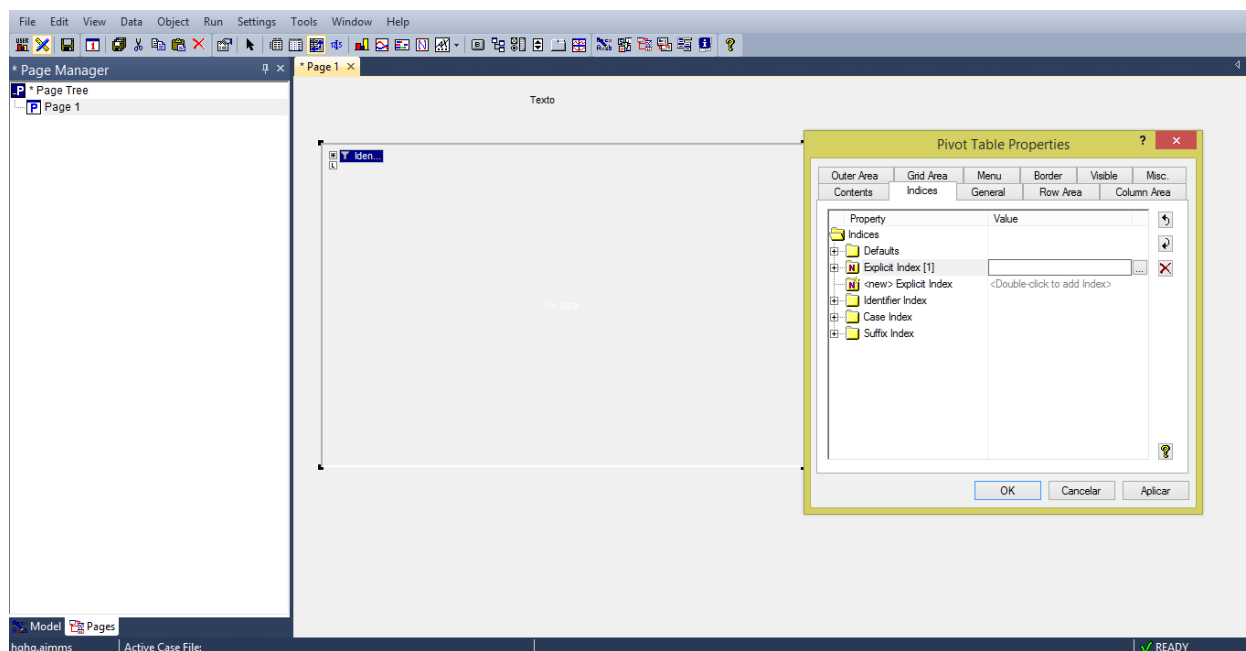


Figura 49

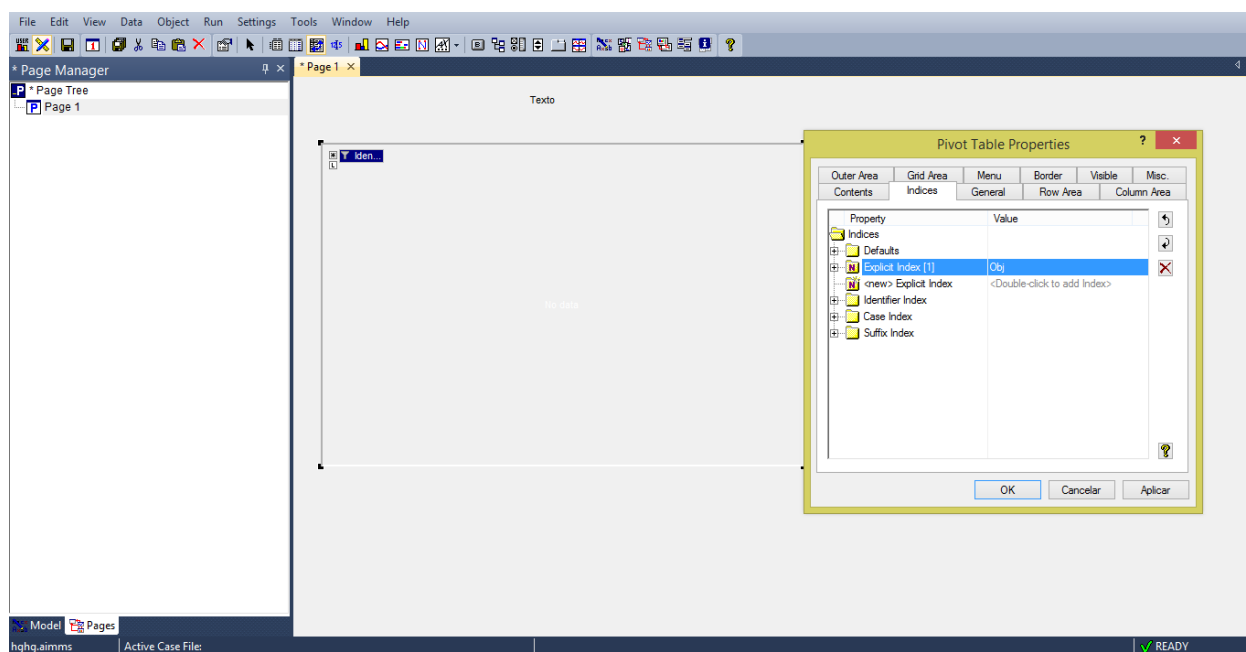


Figura 50

Se quiser explorar as especificações clique em + do lado de *explicit index* :

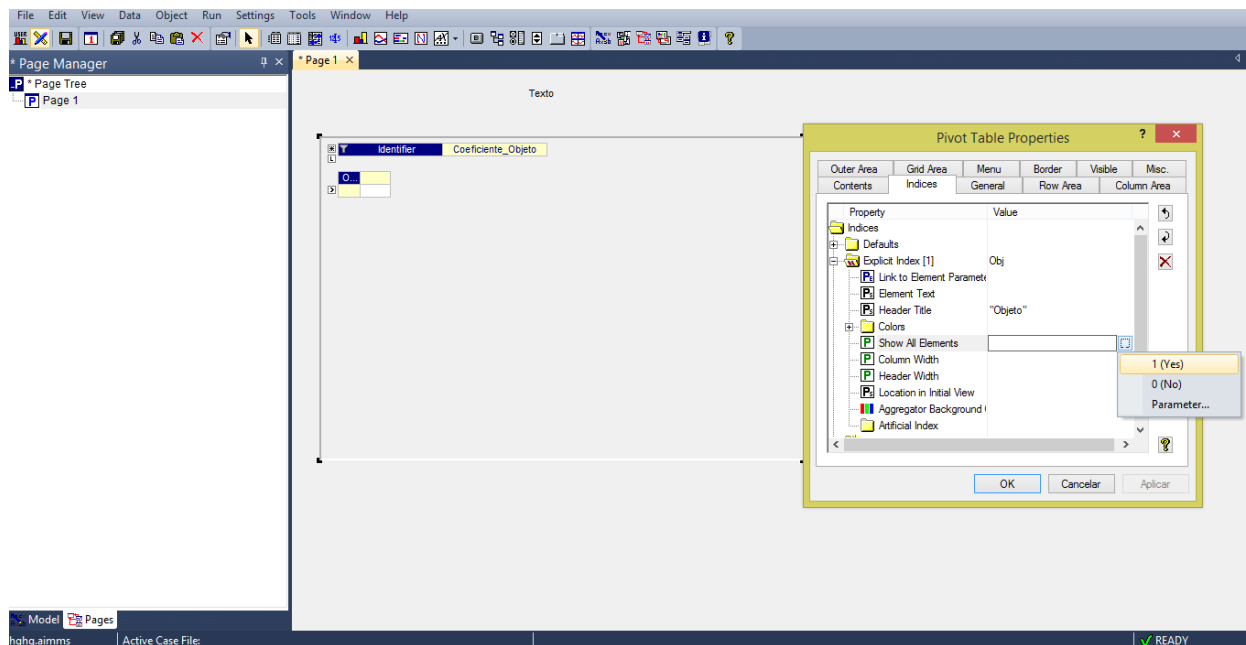


Figura 51

O mesmo pode ser feito com o identificador. Se preenchermos os valores do parâmetro e do conjunto, aparecerá:

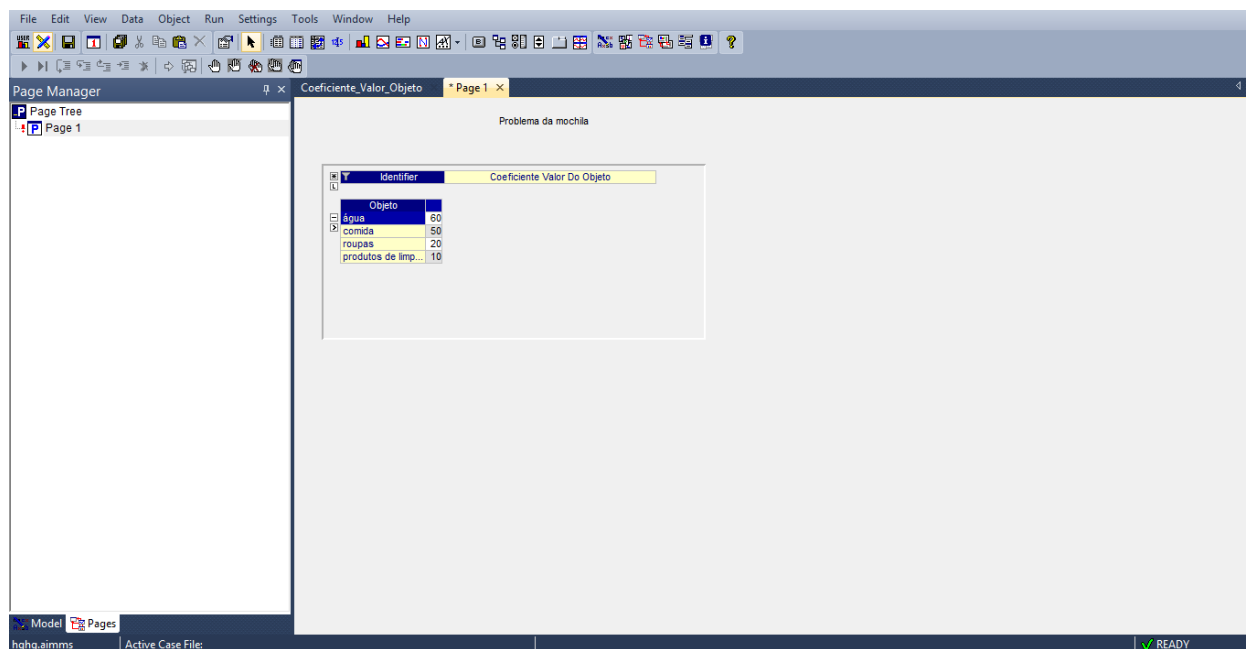


Figura 52

Vamos fazer o mesmo processo para inserir o parâmetro (coeficiente\_peso\_objeto(obj))

Como o processo é muito similar ao primeiro, basta copiar a primeira tabela e colar em outra área, lembrando de mudar nome e identificador, ficará:

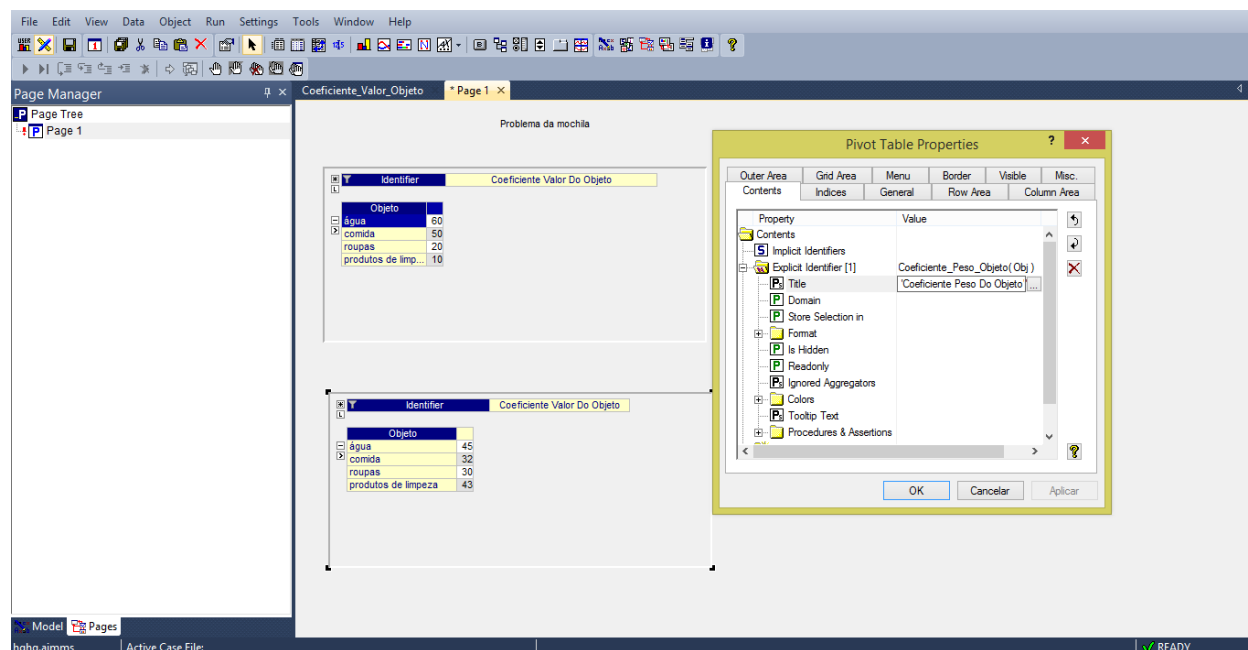


Figura 53

Agora insira a tabela para a variável Quantidade\_de\_cada\_objeto(Obj). Basta selecionar a variável no lugar do parâmetro. Ficar assim:

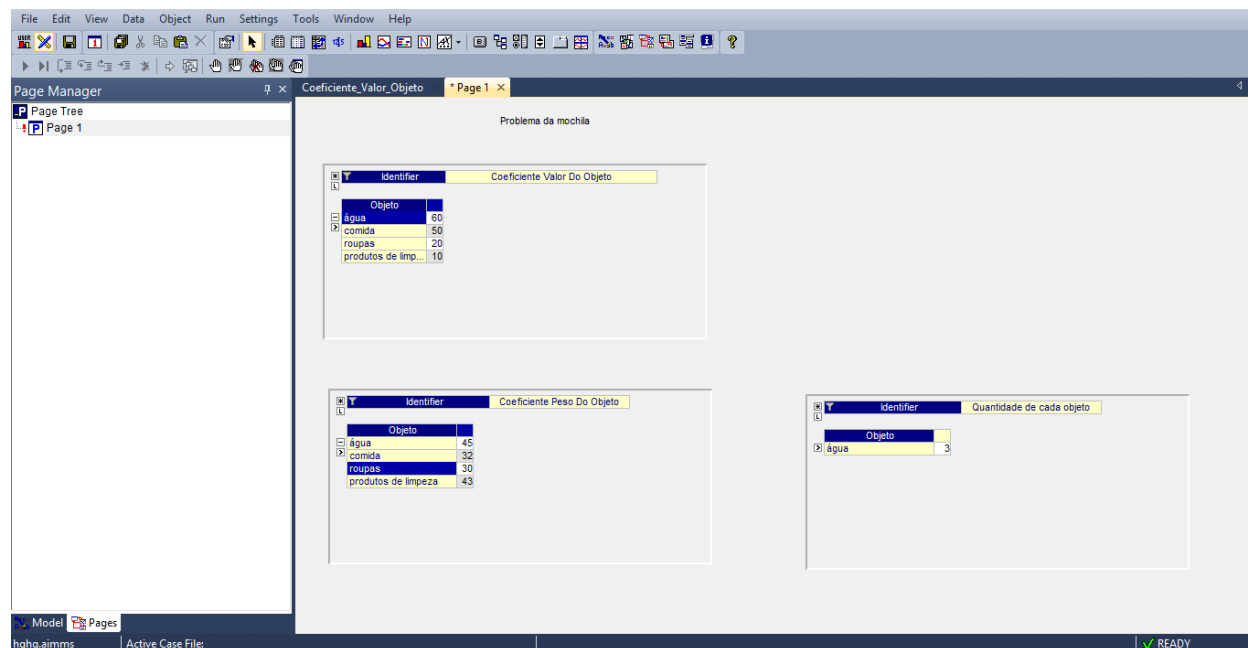


Figura 54

Lembrando que os parâmetros poderão ser modificados em tela, mas a variável não , já que seu valor se dá somente por otimização. Para ver isso, basta colocar a tela em modo usuário (F4) e tentar modificá-las.

Note que a tabela não ficou em branco, isso por que já rodamos o procedimento de otimização antes. Veja também que só o elemento água está aparecendo, isso pois só ele tem valor, por default, a tabela só mostra elementos diferentes de vazios. Então vamos criar um botão para o usuário rodar a otimização :

## 12.4 Inserindo botão

Para inserir um botão, vá em *Object – Button*

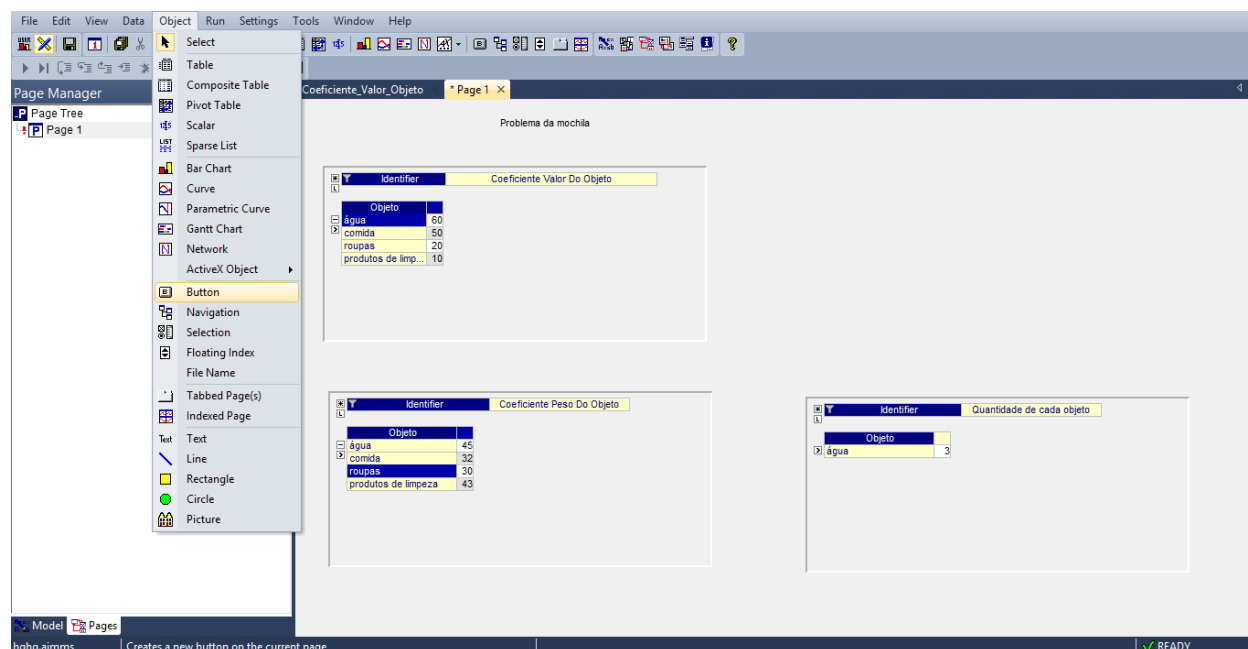


Figura 55

Selecione a área e preencha o nome do botão. Note que também poderá ser inserido imagens, apontador de imagens e etc.

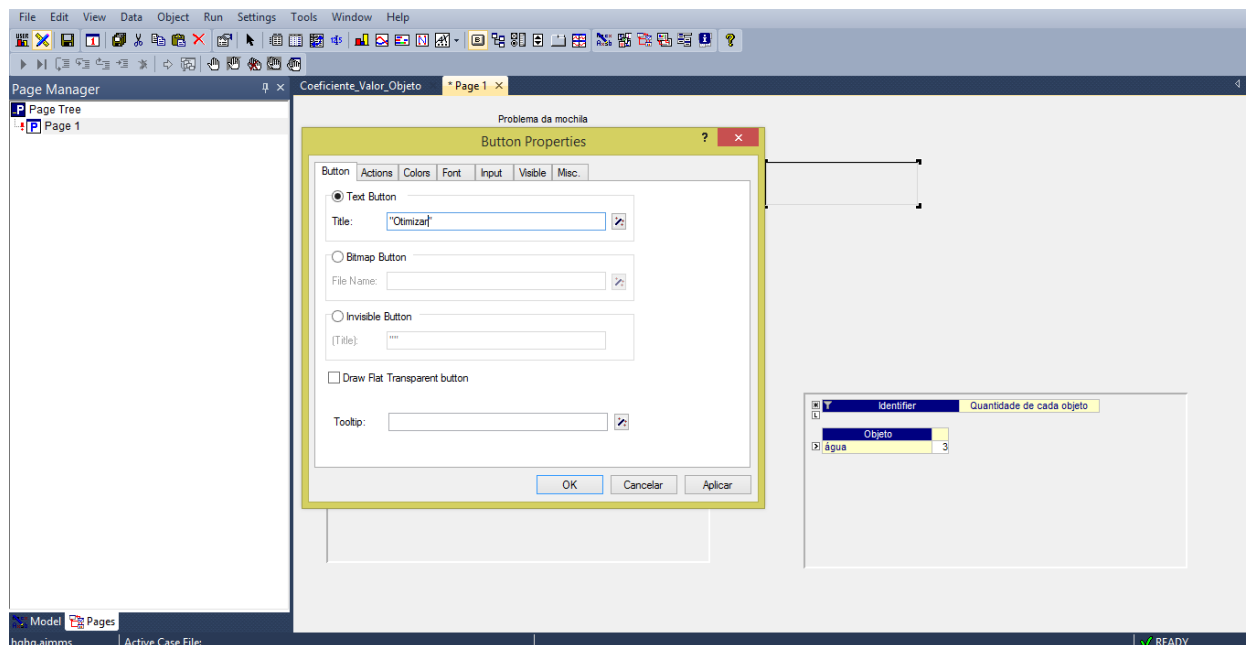


Figura 56

Vá em *actions*. Aqui temos muitas opções, como Fechar janela, irá para outra pagina e etc. Vá em *Run* e adicione.

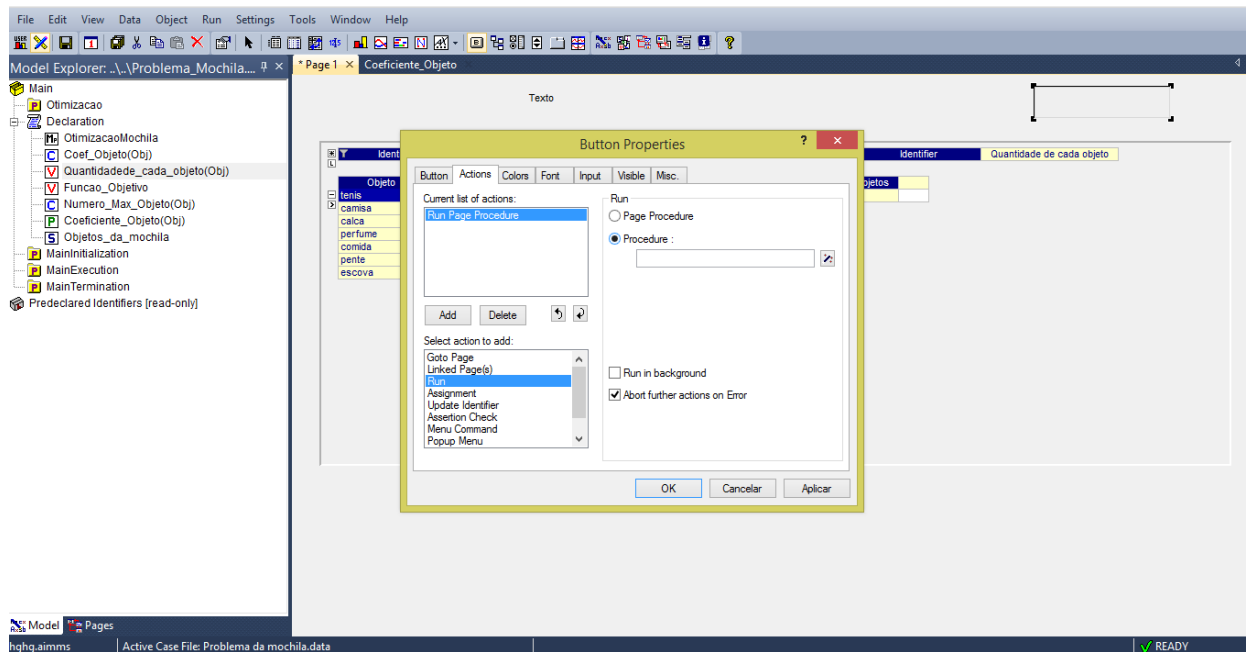


Figura 57

Selecione procedure e insira o procedimento otimização (Rodar\_Otimizacao).

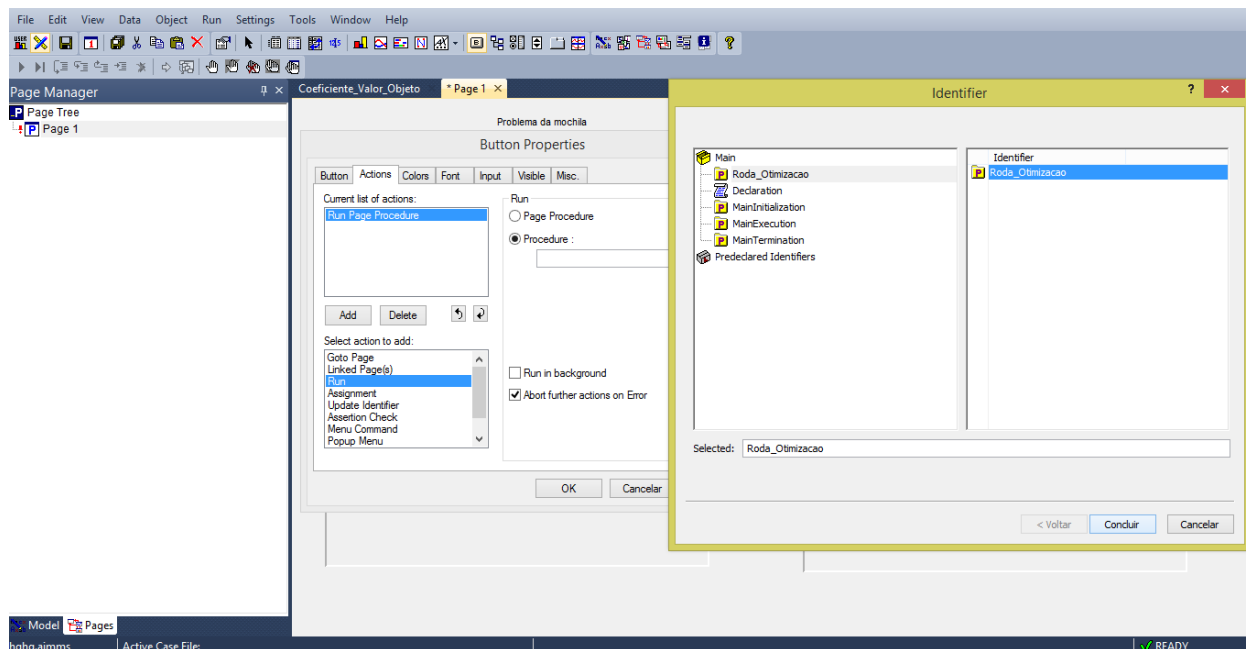


Figura 58

Inserindo o parâmetro da capacidade máxima da mochila, vamos adicionar um *scalar* :

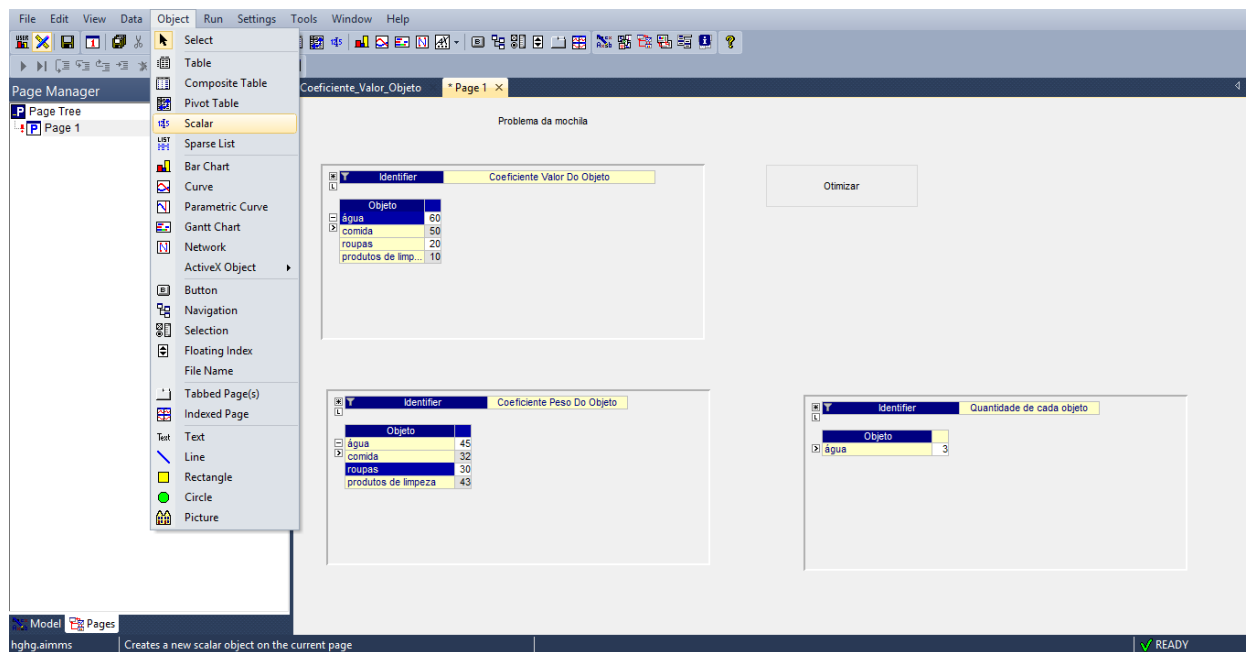


Figura 59

Selecione o espaço que deve ser colocado e depois o parâmetro :

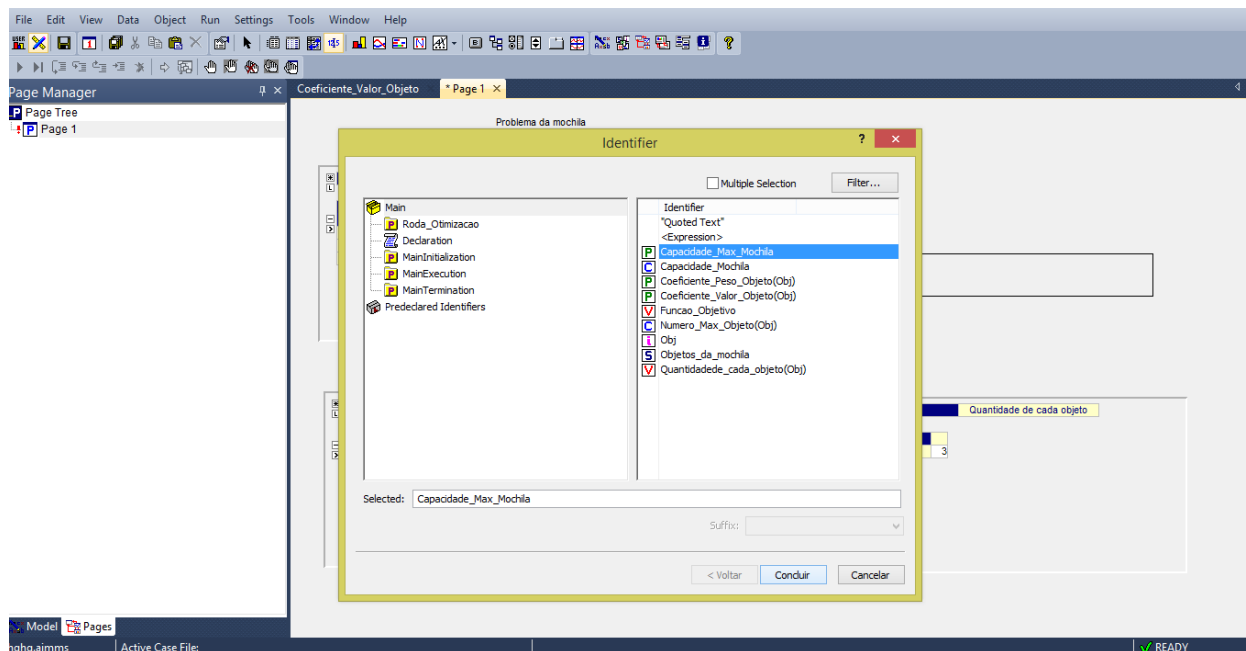


Figura 60

Deverá ficar :

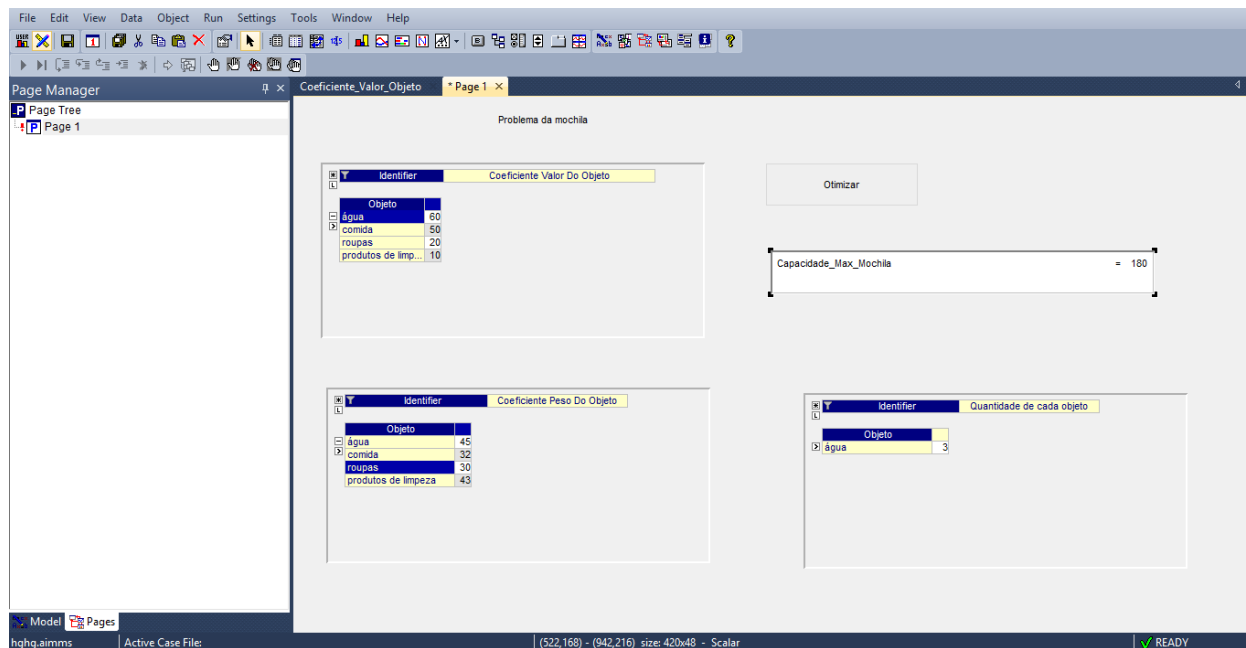


Figura 61

Termine, coloque a pagina em modo usuário selecionando F4 e teste o botão. Veja que, se todos os dados estiverem preenchidos, irá aparecer valor na tabela da variável.

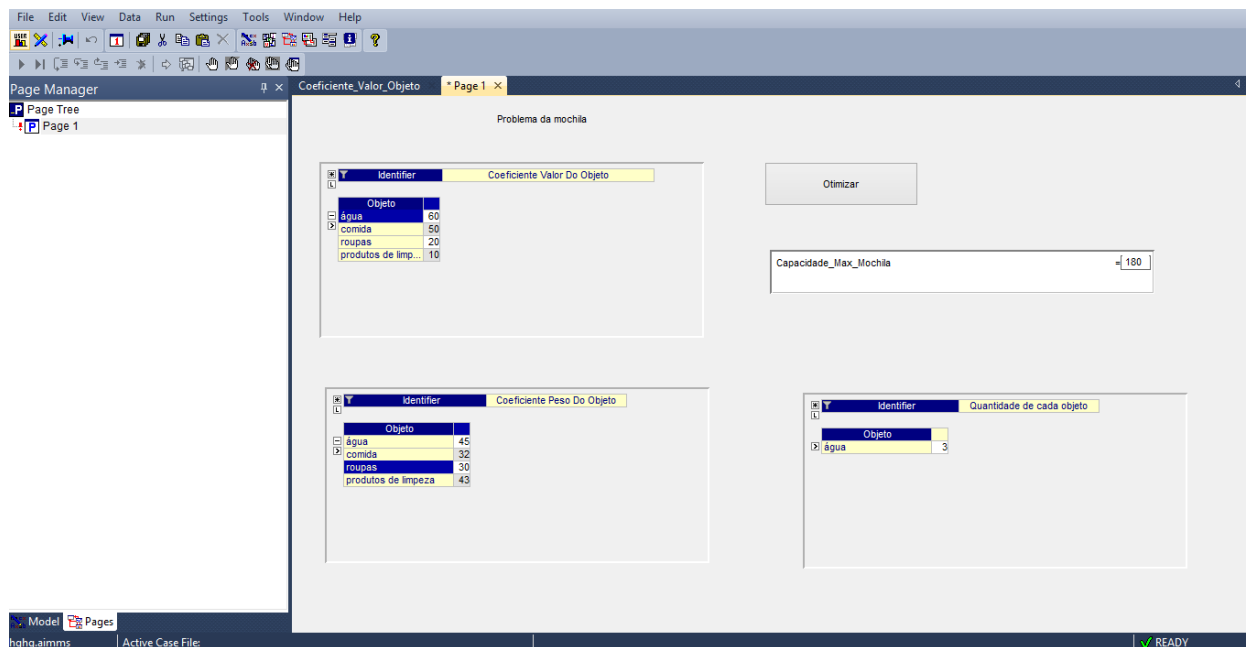


Figura 62

Suponha agora que não queremos o conjunto de elementos já definido, e sim a ser adicionado na interface gráfica, logo apagando os elementos, e reordenando a página, vamos adicionar um *Floating Index*:

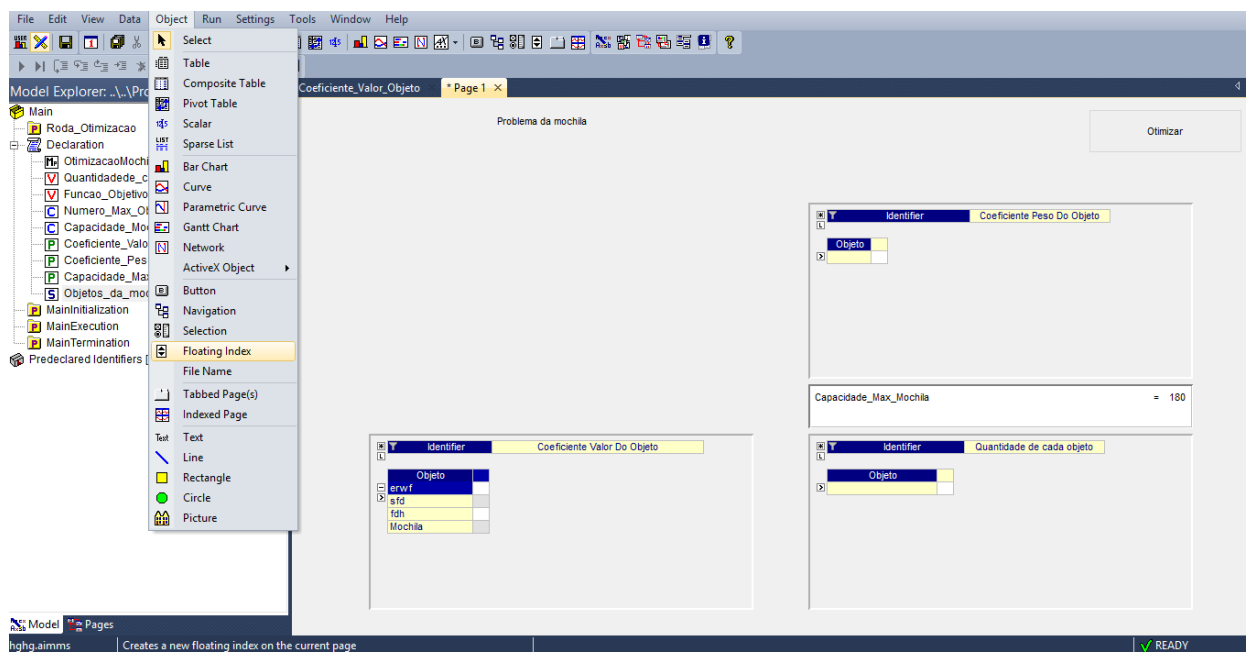


Figura 63

Selecione a área e o conjunto :



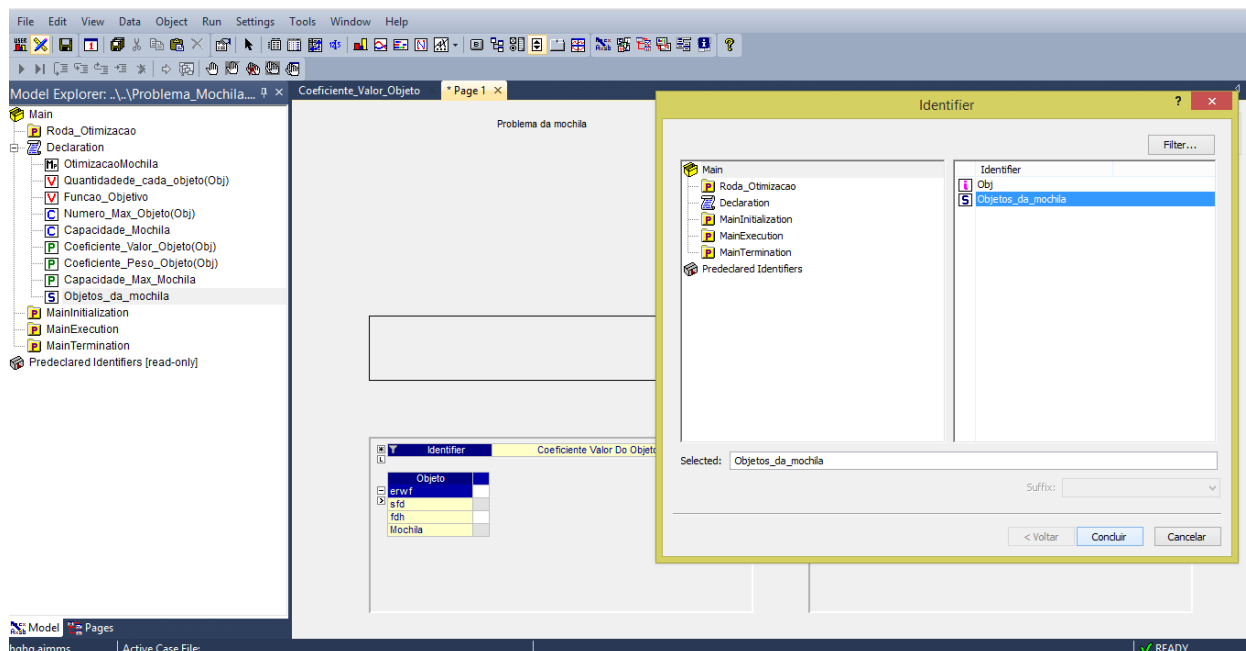


Figura 64

Mas agora precisamos modificar o *Floating index* para que possamos adicionar elementos ou excluir. Para isso, conclua o processo acima e clique duas vezes na caixa já pronta:

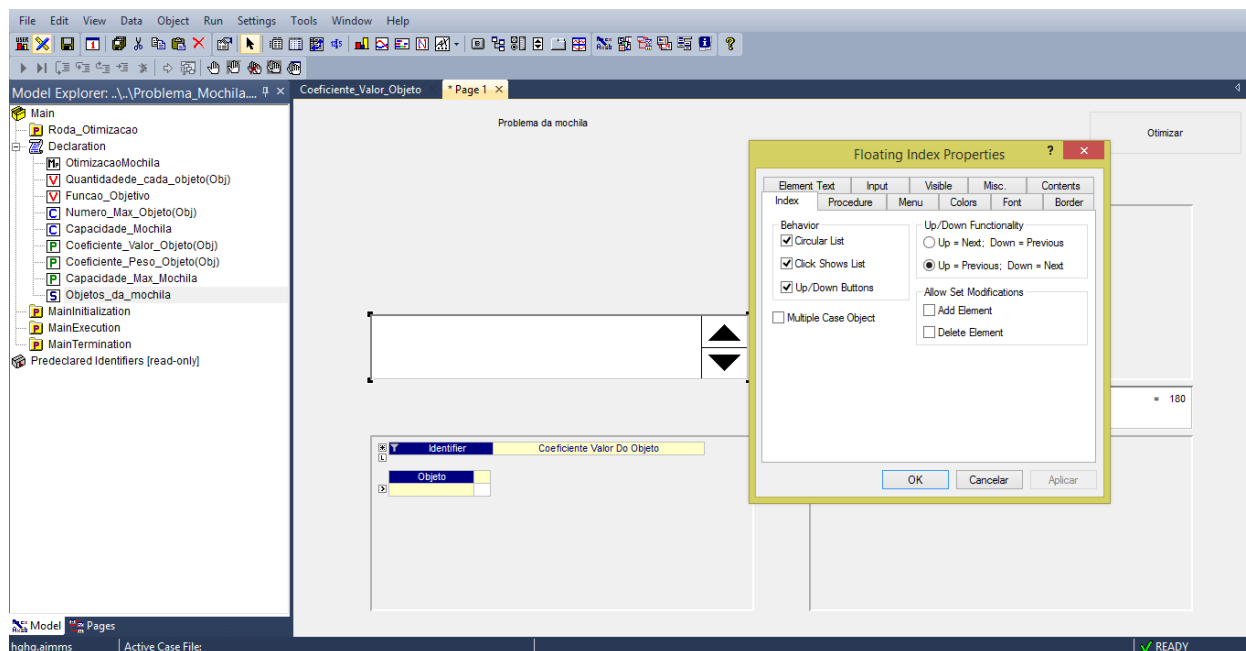


Figura 65

Tique *Add Element* e *Delete Element*.  
Dê ok e ficará assim:

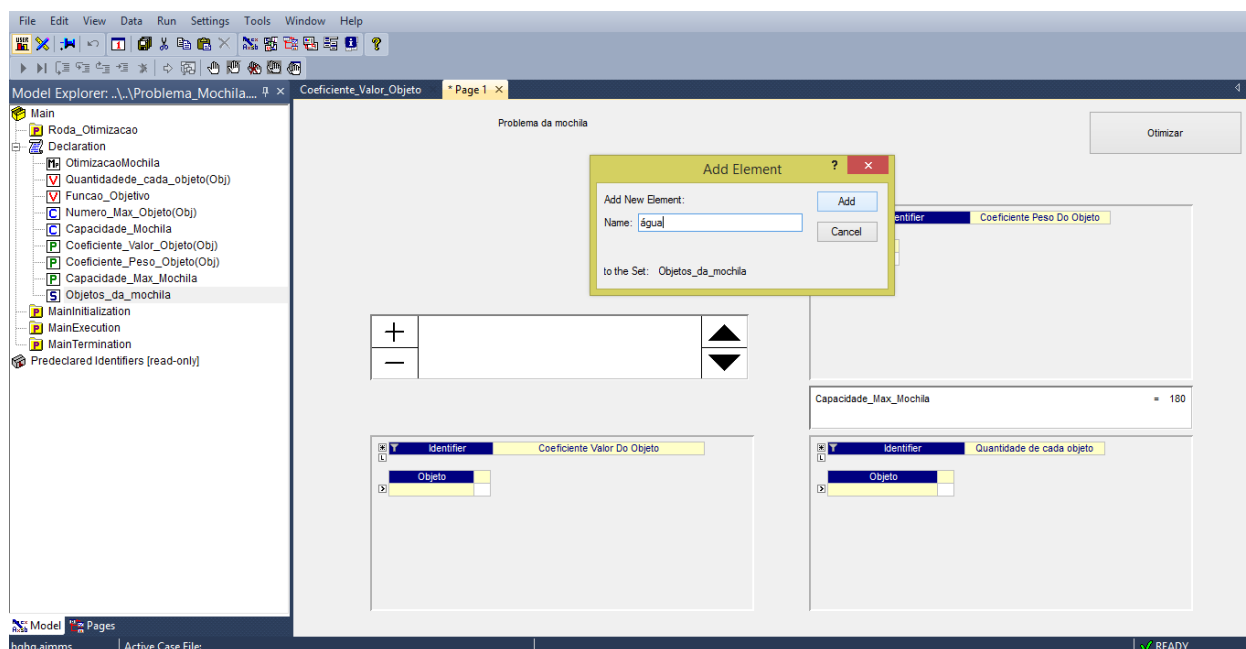


Figura 66

Agora o usuário poderá adicionar ou excluir elementos do conjunto em tela.

Um observação importante é que mesmo adicionando elementos, as tabelas dos parâmetros não apareceram nada. Isto ocorre pois elas estão programadas a só aparecer elementos que tem dados. Logo vamos alterar essa configuração. Para isso, clique duas vezes na tabela e vá em *Indices-> Explicit Index -> + -> Show all elements* e selecione *yes*. Isto fará com que a tabela mostre os elementos sem valor. Faça isso para todas as tabelas.

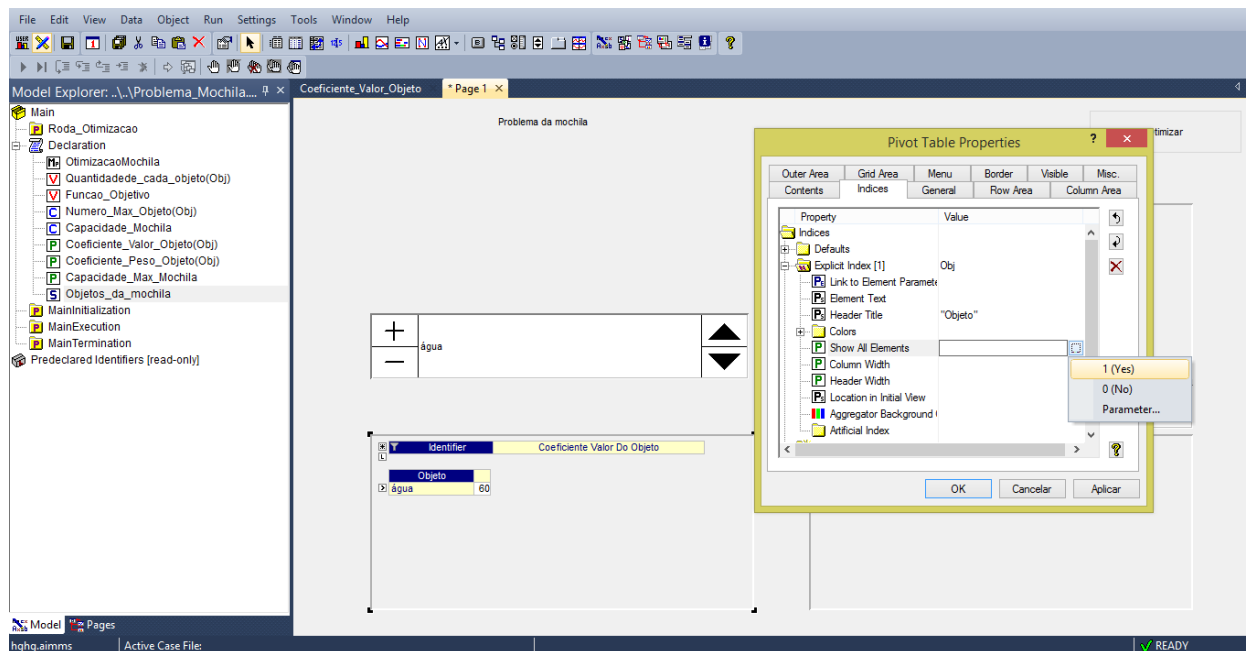


Figura 67

E assim nossa página estará pronta.

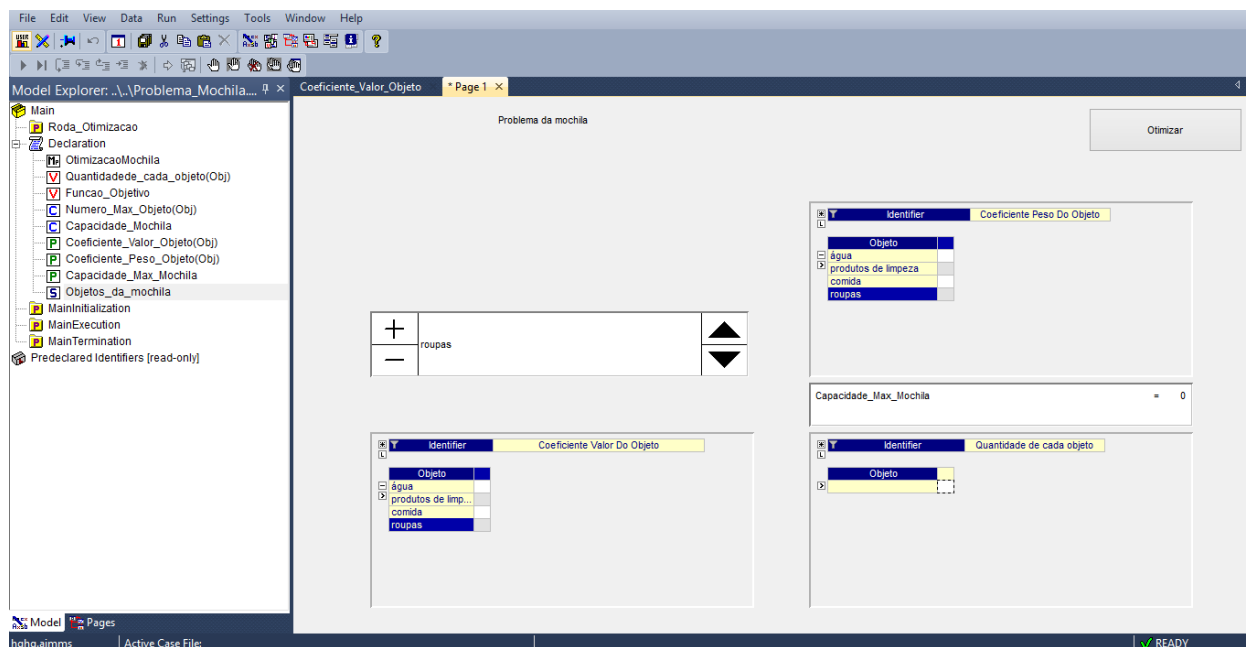


Figura 68

É uma barra em baixo que mostra todos os erros do programa, como as vezes esquecemos o ; por exemplo.

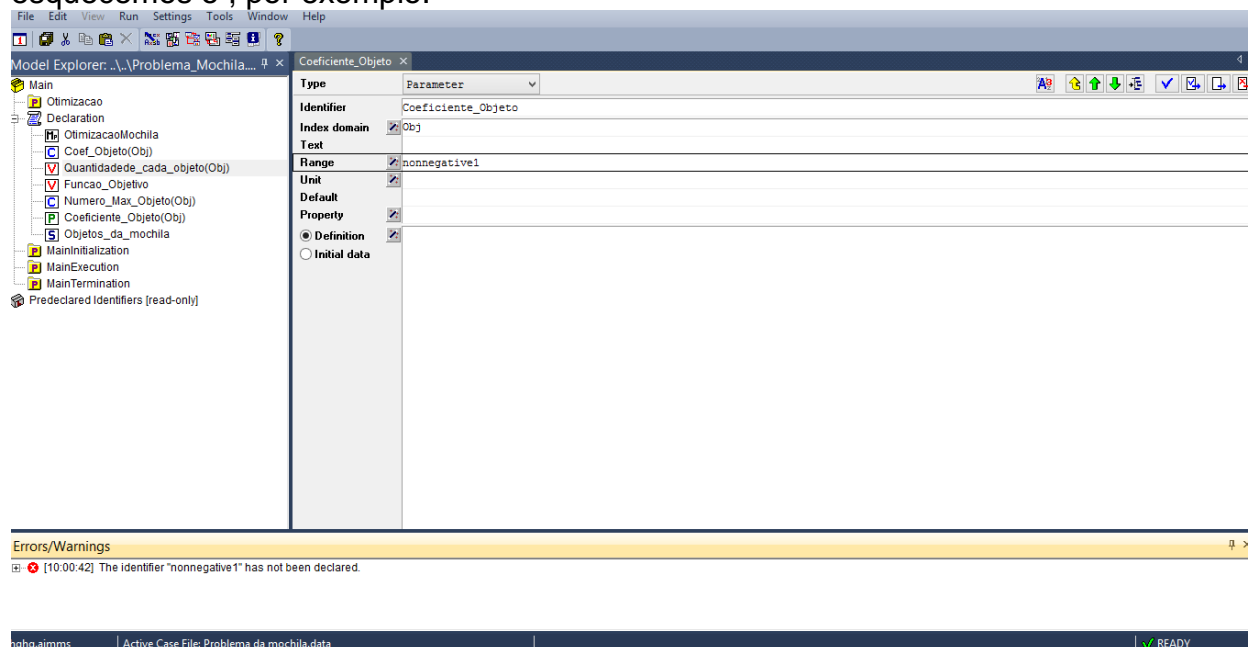


Figura 69

Suponha que colocamos *nonnegative1* ao invés de *nonnegative* no range do coeficiente\_objeto, na hora de dar *check*, ele irá nos avisar o erro e onde ele está (clitando em cima do erro). As vezes o erro só aparece quando colocamos para rodar algo ou compilar, alguma atribuição errada e etc.

## 14 Salvar Data

Suponha que preenchamos todos os dados que queremos usar eles na próxima vez que entramos no programa (não queremos preencher tudo de novo) . O programa não salva os dados sozinho, para isso devemos dar *Salve data*, vamos fazer isso, vá em *Data- Salve Case/ salve Case as*

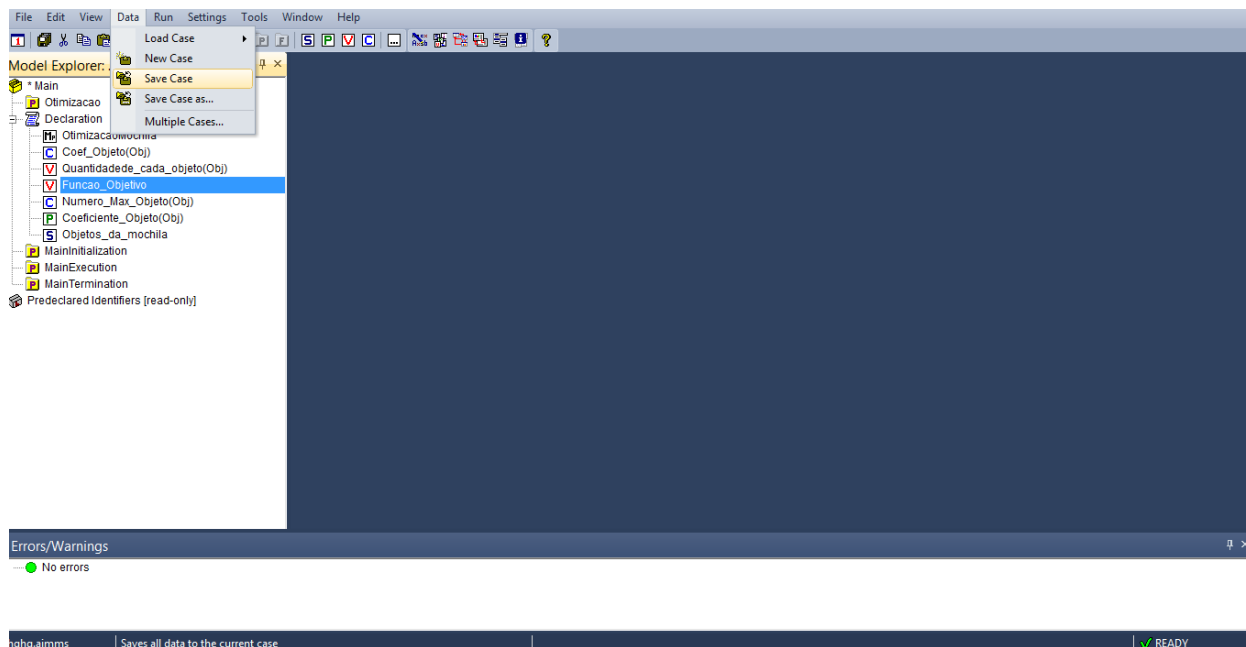


Figura 70

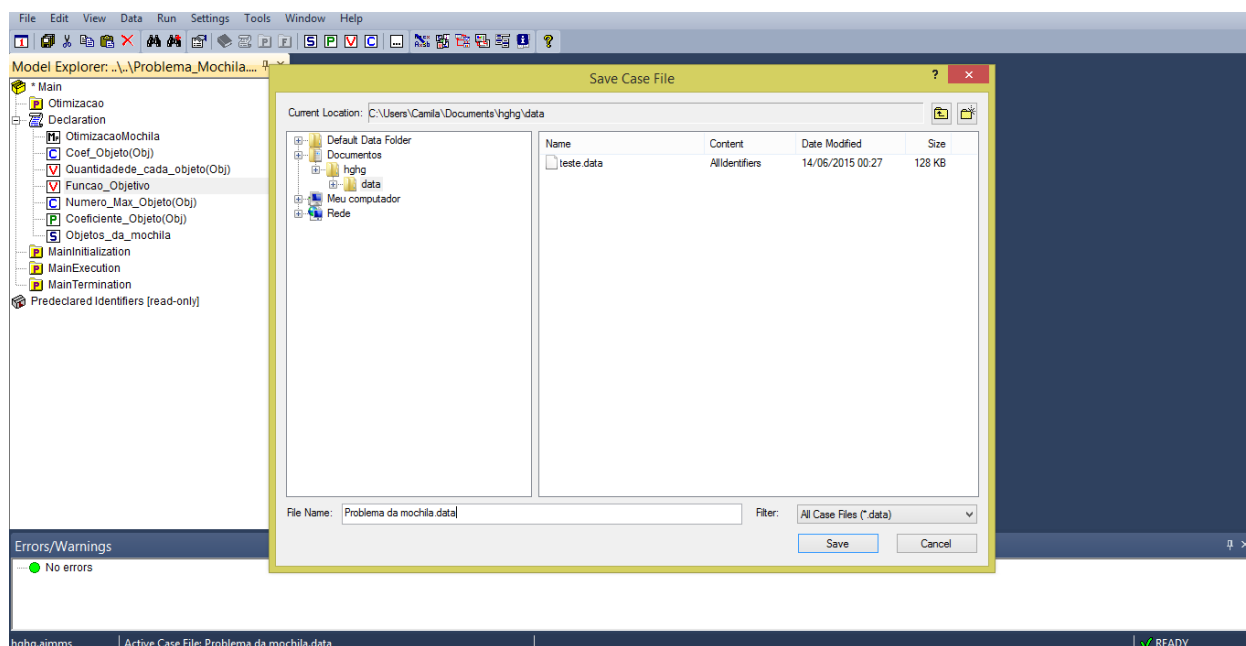


Figura 71

Salve com seu nome de preferência e pronto, cada vez que abrir o projeto basta dar *load* no caso em *Load Case – As Actived*.

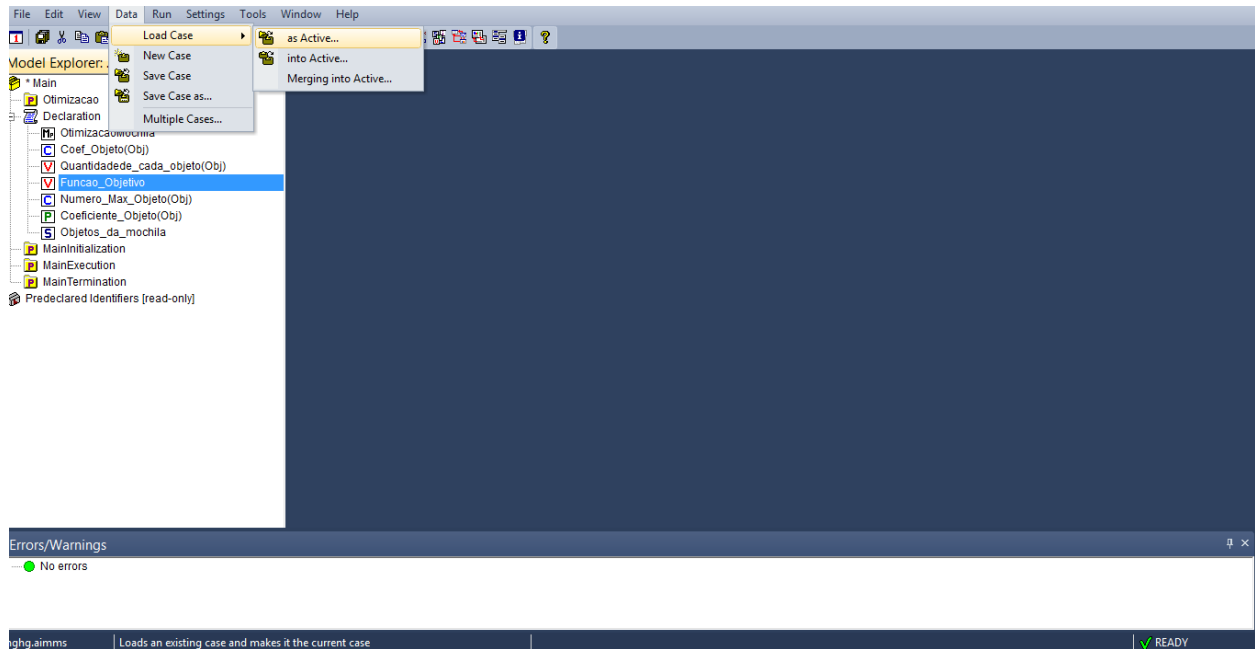


Figura 72

Lembrando que para os dados da otimização fiquem salvos, você deve copiar os dados da variável para um parâmetro.

## Conclusão

Depois de ter lido esse manual, o leitor deve estar apto a usar o AIMMS de forma básica, como construir um pequeno programa de otimização e a aprender mais sozinho procurando no *help* do AIMMS, como *language reference*, *function reference* etc..

## Referência Bibliográfica

Tutorial for Beginners : <http://www.aimms.com/downloads/tutorials/tutorial-for-beginners/>

Tutorial for Professionals : <http://www.aimms.com/downloads/tutorials/tutorial-for-professionals/>

AIMMS User Guide : [http://www.ime.unicamp.br/~moretti/aimms/AIMMS\\_user.pdf](http://www.ime.unicamp.br/~moretti/aimms/AIMMS_user.pdf)

AIMMS Reference Language:

[http://www.ime.unicamp.br/~moretti/aimms/AIMMS\\_ref.pdf](http://www.ime.unicamp.br/~moretti/aimms/AIMMS_ref.pdf)