PowerManager

1.0

Generated by Doxygen 1.8.1.2

Mon Jan 14 2013 18:25:43

Contents

Chapter 1

Data Type Index

1.1 Data Types List

Here are the data types with brief descriptions:

Chapter 2

File Index

2.1 File List

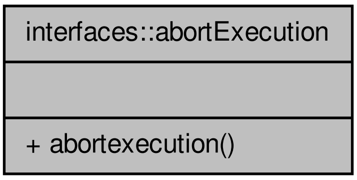
Here is a list of all files with brief descriptions:

Chapter 3

Data Type Documentation

3.1 interfaces::abortExecution Interface Reference

Collaboration diagram for interfaces::abortExecution:



Public Member Functions

- subroutine abortexecution (i, j, line, word, r1, r2)

3.1.1 Detailed Description

Definition at line 38 of file interfaces.f90.

3.1.2 Member Function/Subroutine Documentation

3.1.2.1 subroutine interfaces::abortExecution::abortexecution (integer, intent(in), optional *i,* integer, intent(in), optional *j,* integer, intent(in), optional *line,* character(len=∗), intent(in), optional *word,* real(kind(1.d0)), intent(in), optional *r1,* real(kind(1.d0)), intent(in), optional *r2*)

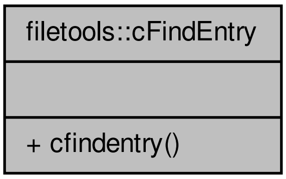
Definition at line 38 of file interfaces.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/interfaces.f90

3.2 filetools::cFindEntry Interface Reference

Collaboration diagram for filetools::cFindEntry:



Public Member Functions

- subroutine cfindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

3.2.1 Detailed Description

Definition at line 128 of file fileTools.f90.

3.2.2 Member Function/Subroutine Documentation

3.2.2.1 subroutine filetools::cFindEntry::cfindentry (character(len=100), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

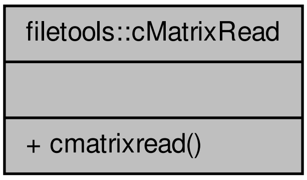
Definition at line 128 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.3 filetools::cMatrixRead Interface Reference

Collaboration diagram for filetools::cMatrixRead:



Public Member Functions

- character(len=100) function,
  dimension(nline, ncol) cmatrixread (theUnit, nline, ncol, first_, last_)

3.3.1 Detailed Description

Definition at line 84 of file fileTools.f90.

3.3.2 Member Function/Subroutine Documentation

3.3.2.1 character(len=100) function, dimension(nline,ncol) filetools::cMatrixRead::cmatrixread (integer, intent(in) *theUnit*, integer, intent(in) *nline*, integer, intent(in) *ncol*, character(len=1), optional *first_*, character(len=1), optional *last_*)

Definition at line 84 of file fileTools.f90.

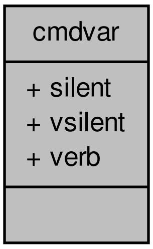
The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.4 cmdvar Module Reference

Collects the variable read from command line.

Collaboration diagram for cmdvar:



Public Attributes

- logical silent

  Controls the input to the screen.
- logical vsilent
- logical verb

3.4.1 Detailed Description

Collects the variable read from command line.

Author

Andrea Facci

Definition at line 39 of file cmdVar.f90.

3.4.2 Member Data Documentation

3.4.2.1 logical cmdvar::silent

Definition at line 42 of file cmdVar.f90.

3.4.2.2 logical cmdvar::verb

Definition at line 42 of file cmdVar.f90.

3.4.2.3 logical cmdvar::vsilent

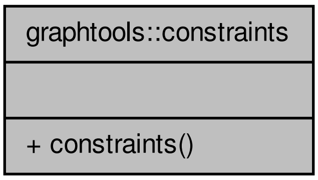
Definition at line 42 of file cmdVar.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/cmdVar.f90

3.5 graphtools::constraints Interface Reference

Collaboration diagram for graphtools::constraints:



Public Member Functions

- logical function constraints (c, t)

3.5.1 Detailed Description

Definition at line 60 of file graphTools.f90.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 logical function graphtools::constraints::constraints (integer, dimension(nm), intent(in) *c,* integer, intent(in) *t*)

Definition at line 60 of file graphTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/graphTools.f90

3.6 filetools::dFindEntry Interface Reference

Collaboration diagram for filetools::dFindEntry:

Public Member Functions

- subroutine dfindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

3.6.1 Detailed Description

Definition at line 115 of file fileTools.f90.

3.6.2 Member Function/Subroutine Documentation

3.6.2.1 subroutine filetools::dFindEntry::dfindentry (character(len=100), intent(in) *entry,* integer, intent(in) *n,* integer, intent(in) *theUnit,* logical, intent(in), optional *rew,* real(kind(1.d0)), dimension(n), intent(out), optional *valore,* logical, intent(out), optional *isPresent,* integer, intent(out), optional *nRow*)

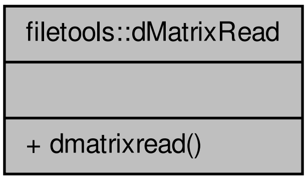
Definition at line 115 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.7 filetools::dMatrixRead Interface Reference

Collaboration diagram for filetools::dMatrixRead:



Public Member Functions

- real(kind(1.d0)) function,
  dimension(nline, ncol) dmatrixread (theUnit, nline, ncol, first_, last_)

3.7.1 Detailed Description

Definition at line 66 of file fileTools.f90.

3.7.2 Member Function/Subroutine Documentation

3.7.2.1 real(kind(1.d0)) function, dimension(nline,ncol) filetools::dMatrixRead::dmatrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* integer, intent(in) *ncol,* character(len=1), optional *first_,* character(len=1), optional *last_*)

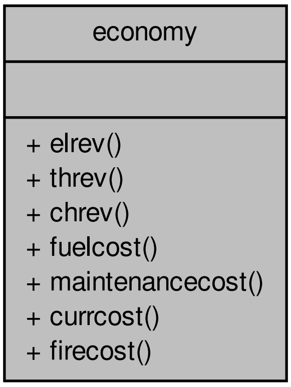
Definition at line 66 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.8 economy Module Reference

Collaboration diagram for economy:



Public Member Functions

- real(kind(1.d0)) function elrev (c, t)

  Electric energy revenues.
- real(kind(1.d0)) function threv (t)

  thermal energy revenues.
- real(kind(1.d0)) function chrev (t)

  Chilling energy revenues.
- real(kind(1.d0)) function fuelcost (c, t)

  Calculates the costs to buy the fuel.
- real(kind(1.d0)) function maintenancecost (c, t)

  Calculates the mintenance costs.
- real(kind(1.d0)) function currcost (c, t)

  Calculates the profit during a time step.
- real(kind(1.d0)) function firecost (cNew, cOld)

  Calculates the lighting cost.

3.8.1 Detailed Description

Author

This module contains the definition of all the procedures that perform economic calculations for a give set-point and time-step, that are, fuel costs, O&M costs, and the revenues from thermal, electric, and chilling, energy selling.

Author

Definition at line 44 of file economy.f90.

3.8.2 Member Function/Subroutine Documentation

3.8.2.1 real(kind(1.d0)) function economy::chrev (integer, intent(in) *t*)

Calculates the revenues (in euro or any other currency according to the one used in the input) from chilling energy selling to the various clients.

$$R_{ch} = \sum_{clients} U_{ch}(i)c_{ch}(i)dt$$

where $U_{ch}(i)$ is the power demand (in kW) of the i'th client, $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 183 of file economy.f90.

Referenced by currcost().

3.8.2.2 real(kind(1.d0)) function economy::currcost (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

Calculates the profit of a time-step(in euro or any other currency according to the one used in the input) for a given set-point, all the costs, except for the costs associated to equipment ignition.

$$G = R_{el} + R_{th} + R_{ch} + C_f + C_m$$

where

- R_{el} are the electrical revenues;
- R_{th} are the thermal revenues;
- R_{ch} are the chilling revenues
- C_f are the fuel costs;
- C_m are the maintenance costs;
- dt is the time-step duration.

  Having discarded lighting costs, this profit depends only on the state of the plant at a determined time-step and not on the state at pre previous or subsequent time-step and will be associated to a vertex of the graph.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 315 of file economy.f90.

References chrev(), elrev(), fuelcost(), maintenancecost(), and threv().

Referenced by objfunction().

3.8.2.3 real(kind(1.d0)) function economy::elrev (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

Calculates the revenues (in euro or any other currency according to the one used in the input) from electric energy selling to the various clients and to the grid. Electric energy revenues are calculated in a different way for each kind of grid connection. Specifically:

- Stand Alone:

  $$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt$$

  where $U_{el}(i)$ is the power demand (in kW) of the i'th client, $c_{el}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

- Grid connected with net metering:

  $$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt + P_{el} c_{s_{grid}} - U_{el}^{t} c_{b_{grid}}$$

  where P_{el} is the total electric power produced by the power plant, $c_{s_{grid}}$ is the selling price to the grid, $U_{el}^{t} = \sum_{clients} U_{el}(i) + U_{el}^{self}$ is the total electric demand, including the power plant self-consumption U_{el}^{self}

- Grid Connected without net metering:

  $$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt + (P_{el} - U_{el}^{t}) c_{grid}$$

  where $c_{grid} = c_{s_{grid}}$ if $P_{el} \geq U_{el}^{t}$ and $c_{grid} = c_{b_{grid}}$ otherwise

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 80 of file economy.f90.

References energy::elprod(), and energy::elselfcons().

Referenced by currcost().

3.8.2.4 real(kind(1.d0)) function economy::firecost (integer, dimension(nm), intent(in) *cNew*, integer, dimension(nm), intent(in) *cOld*)

Calculates the cost associated to each lighting of a machinery.

$$C_l > 0 \qquad \text{if} \qquad sp(t,i) > 0 \text{ and } sp(t-1,i) = 0$$

this cost will be added to the operative profit to for the arc profit/cost.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 347 of file economy.f90.

Referenced by graphtools::grapharcs().

3.8.2.5 real(kind(1.d0)) function economy::fuelcost (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

Calculates the costs to buy the fuel (in euro or any other currency according to the one used in the input)

$$C_f = \sum_{Trig+Boi} \frac{E_{in}(i)c_f(i)}{H_i(i)} dt$$

where $c_f(i)$ is the specific cost (per unit mass or volume) of the fuel for the i'th machine, $H_i(i)$ is the fuel LHV (kJ per unit mass or volume), $E_{in}(i)$ is the primary energy input. $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Note that even though international units are strongly suggested, any units of mass and/or volume is valid for c_f and H_i provided that coherence between the units of these two variables is respected. Moreover prices and LVSs may may be expressend in different units for differend machines.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 228 of file economy.f90.

References energy::fuelcons().

Referenced by currcost().

3.8.2.6 real(kind(1.d0)) function economy::maintenancecost (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

Calculates the maintenance costs (in euro or any other currency according to the one used in the input)

$$C_m = \sum c_m(i)dt \qquad \text{if} \qquad sp(i) > 0$$

where $c_m(i)$ is the maintenance cost per unit time and dt is the time-step duration.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 270 of file economy.f90.

Referenced by currcost().

3.8.2.7 real(kind(1.d0)) function economy::threv (integer, intent(in) *t*)

Calculates the revenues (in euro or any other currency according to the one used in the input) from thermal energy selling to the various clients.

$$R_{th} = \sum_{clients} U_{th}(i) c_{th}(i) dt$$

where $U_{th}(i)$ is the power demand (in kW) of the i'th client, $c_{th}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 144 of file economy.f90.

Referenced by currcost().

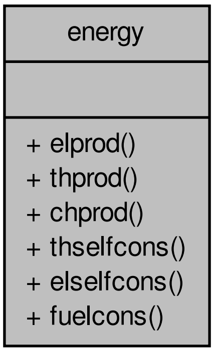
The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/economy.f90

3.9 energy Module Reference

module for energy calculations.

Collaboration diagram for energy:



Public Member Functions

- real(kind(1.d0)) function elprod (c_)

  Electrical production.

- real(kind(1.d0)) function thprod (c_)

  Thermal production.
- real(kind(1.d0)) function chprod (c_)

  Chilling production.
- real(kind(1.d0)) function thselfcons (c_)

  Internal thermal consumption of the power plant.
- real(kind(1.d0)) function elselfcons (c_)

  Internal electrical consumption of the power plant.
- real(kind(1.d0)) function,
  dimension(nm) fuelcons (c_)

  Primary energy input.

3.9.1 Detailed Description

This module contains all procedures tu calculate the enrgy fluxes inside the power plant and bertween the power plant and the clients.

Author

Andrea Facci

Definition at line 38 of file energy.f90.

3.9.2 Member Function/Subroutine Documentation

3.9.2.1 real(kind(1.d0)) function energy::chprod (integer, dimension(nm), intent(in) *c_*)

Calculates the chilling production in kW of the whole power plant, for a given set-point Note that only trigeneration machines and Chillers produce chilling power so far. Thus chilling power is:

$$P_{ch} = \sum_{Trig} \frac{sp(i) \cdot P_{max}(i)}{\eta_{el}(i, sp(i))} \eta_{ch}(i, sp(i)) + \sum_{Chi} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i'th machine, η_{ch} and η_{el} are the chilling and electrical efficiencies, respectively, and $P_{max}(i)$ is its rated power.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 144 of file energy.f90.

Referenced by constraints().

3.9.2.2 real(kind(1.d0)) function energy::elprod (integer, dimension(nm), intent(in) *c_*)

Calculates the electrical production in kW of the whole power plant, for a given set-point Note that only trigeneration machines produce electrical power so far. Thus electrical power is:

$$P_{el} = \sum_{Trig} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i'th trigenerative machine and $P_{max}(i)$ is its rated power.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 55 of file energy.f90.

Referenced by economy::elrev().

3.9.2.3 real(kind(1.d0)) function energy::elselfcons (integer, dimension(nm), intent(in) *c_*)

Calculates the electrical self-consumption of the trigeneration plant, that is, the electrical power needed by the mechanical chillers, for a given set-point

$$U_{el}^{self} = \sum_{MecChi} \frac{sp(i) \cdot P_{max}(i)}{\eta_{ch}(i, sp(i))}$$

where $sp(i)$ is the set point of the i'th machine, η_{ch} is the chilling efficiency, and $P_{max}(i)$ is its rated power. The summation is extended over the nmber of mechanical chillers.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 231 of file energy.f90.

Referenced by economy::elrev().

3.9.2.4 real(kind(1.d0)) function, dimension(nm) energy::fuelcons (integer, dimension(nm), intent(in) *c_*)

Calculates the eprimary energy input of the trigeneration plant, for a given set-point

$$E_{in}(i) = \frac{sp(i) \cdot P_{max}(i)}{\eta(i, sp(i))}$$

where $sp(i)$ is the set point of the i'th machine, $\eta(i, sp(i)) = \eta_{el}(i, sp(i))$ for trigenerative equipment and $\eta(i, sp(i)) = \eta_{th}(i, sp(i))$ for boilers and, and $P_{max}(i)$ is their rated power.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 273 of file energy.f90.

Referenced by economy::fuelcost().

3.9.2.5 real(kind(1.d0)) function energy::thprod (integer, dimension(nm), intent(in) *c_*)

Calculates the Thermal production in kW of the whole power plant, for a given set-point Note that only trigeneration machines and Boilers produce thermal power so far. Thus Thermal power is:

$$P_{th} = \sum_{Trig} \frac{sp(i) \cdot P_{max}(i)}{\eta_{el}(i, sp(i))} \eta_{th}(i, sp(i)) + \sum_{Boi} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i'th machine, η_{th} and η_{el} are the thermal and electrical efficiencies, respectively, and $P_{max}(i)$ is its rated power.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 97 of file energy.f90.

Referenced by constraints().

3.9.2.6 real(kind(1.d0)) function energy::thselfcons (integer, dimension(nm), intent(in) *c_*)

Calculates the thermal self-consumption of the trigeneration plant, that is, hte thermal power needed by the absorption chillers.

$$U_{th}^{self} = \sum_{AbsChi} \frac{sp(i) \cdot P_{max}(i)}{\eta_{ch}(i, sp(i))}$$

where $sp(i)$ is the set point of the i'th machine, η_{ch} is the chilling efficiency, and $P_{max}(i)$ is its rated power.

Parameters

in	*c_*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$

Author

Andrea Facci

Definition at line 188 of file energy.f90.

Referenced by constraints().

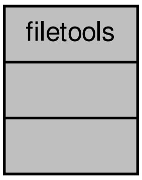
The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/energy.f90

3.10 filetools Module Reference

Interfaces of procedures to read from file.

Collaboration diagram for filetools:



Data Types

- interface cFindEntry
- interface cMatrixRead
- interface dFindEntry
- interface dMatrixRead
- interface findEntry
- interface hCount
- interface iFindEntry
- interface iMatrixRead
- interface matrixRead
- interface readKeyword
- interface rewUnit
- interface vCount

3.10.1 Detailed Description

This module collects all the interfaces of the procedures useful to read data from files.

Author

Andrea Facci.

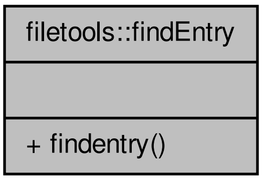
Definition at line 39 of file fileTools.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.11 filetools::findEntry Interface Reference

Collaboration diagram for filetools::findEntry:



Public Member Functions

- subroutine findentry (entry, theUnit, rew, valore, isPresent, nRow)

3.11.1 Detailed Description

Definition at line 141 of file fileTools.f90.

3.11.2 Member Function/Subroutine Documentation

3.11.2.1 subroutine filetools::findEntry::findentry (character(len=100), intent(in) *entry*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

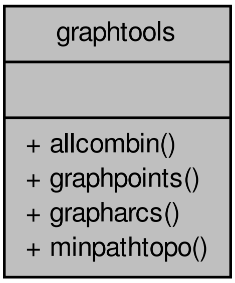
Definition at line 141 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.12 graphtools Module Reference

Collaboration diagram for graphtools:



Data Types

- interface constraints
- interface objFunction

Public Member Functions

- subroutine allcombin (cm, imax, m)

  Generates all the combinations of the set points, staring from the set-point vectors that are stored in the columns of "cm".
- subroutine graphpoints

  Generates the graph veteces, starting from the array of the set-point combinations.
- subroutine grapharcs

  Generates the grapsh arcs.
- subroutine minpathtopo (minPath, minCost)

  Miniumum path detemination.

3.12.1 Detailed Description

Definition at line 34 of file graphTools.f90.

3.12.2 Member Function/Subroutine Documentation

3.12.2.1 subroutine graphtools::allcombin (integer, dimension(maxval(imax),m), intent(in) *cm,* integer, dimension(m), intent(in) *imax,* integer, intent(in) *m*)

Generates all the combinations of the set points, staring from the set-point vectors that are stored in the columns of "cm". Each row of the array "comb" represents a set-point of the plant. The total number of states of the plant is also returned in the variable "nComb"

Author

Andrea Facci

Definition at line 80 of file graphTools.f90.

Referenced by main().

3.12.2.2 subroutine graphtools::grapharcs ( )

Generates the grapsh arcs. Note that only vertex relative to consecutive time-steps are connected and that arcs are oriented in the direction of increasing time. Arcs are stored in the form of a predecessor list "predList", that associates to each node all its predecessors. A weight is assiciated to each element of the predecessor list, equal to the weight of the predecessor vertex plus a cost connected to the variation of state between the actual and predecessor state.

$$arcCost(i,j) = pointCost(c(i)) + fireCost(i,j)$$

The number of predecessors for each vertex is asle stored in the "nPre(i)" array.

Definition at line 215 of file graphTools.f90.

References economy::firecost(), myarithmetic::inan(), and myarithmetic::rnan().

Referenced by main().

3.12.2.3 subroutine graphtools::graphpoints ( )

Generates the graph veteces, starting from the array of the set-point combinations. For each time-step determines which plant state respects the energy (staitc) constraints, and associates them to a graph vertex. A weight, that accounts for the costs/revenues of operating the power plant from time-step t to t+1 at the vertex state, is also calculated for each vertex. The time-step relative to each vertex and the number of verteces for each time step are associated to "pointTime" and "nt" vectors respectively. Vertex 0 will be the starting point of the graph and vertex nPoint + 1 the arriving point

Author

Andrea Facci.

Definition at line 139 of file graphTools.f90.

References objfunction().

Referenced by main().

3.12.2.4 subroutine graphtools::minpathtopo (integer, dimension(0:ntime+1), intent(out) *minPath,* real(kind(1.d0)), intent(out) *minCost*)

This function determies the minumum path that connects the start point (0) to the arrival point of the graph (nPoint + 1), using dynamic programming. Specifically the oprimizazion the algorithm is tailored to sort acyclic graphs with topolgical ordering.

Author

Andrea Facci.

Definition at line 283 of file graphTools.f90.

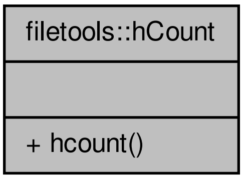
Referenced by main().

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/graphTools.f90

3.13 filetools::hCount Interface Reference

Collaboration diagram for filetools::hCount:



Public Member Functions

- integer function hcount (value_, first_, last_)

3.13.1 Detailed Description

Definition at line 58 of file fileTools.f90.

3.13.2 Member Function/Subroutine Documentation

3.13.2.1 integer function filetools::hCount::hcount (character(len=100), intent(in) *value_*, character(len=1), optional *first_*, character(len=1), optional *last_*)

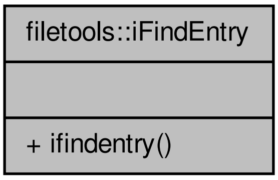
Definition at line 58 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.14 filetools::iFindEntry Interface Reference

Collaboration diagram for filetools::iFindEntry:



Public Member Functions

- subroutine ifindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

3.14.1 Detailed Description

Definition at line 102 of file fileTools.f90.

3.14.2 Member Function/Subroutine Documentation

3.14.2.1 subroutine filetools::iFindEntry::ifindentry (character(len=100), intent(in) *entry,* integer, intent(in) *n,* integer, intent(in) *theUnit,* logical, intent(in), optional *rew,* integer, dimension(n), intent(out), optional *valore,* logical, intent(out), optional *isPresent,* integer, intent(out), optional *nRow*)

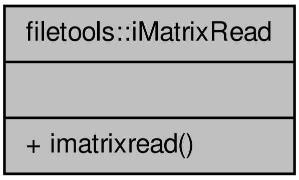
Definition at line 102 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.15 filetools::iMatrixRead Interface Reference

Collaboration diagram for filetools::iMatrixRead:



Public Member Functions

- real(kind(1.d0)) function,
  dimension(nline, ncol) imatrixread (theUnit, nline, ncol, first_, last_)

3.15.1 Detailed Description

Definition at line 75 of file fileTools.f90.

3.15.2 Member Function/Subroutine Documentation

3.15.2.1 real(kind(1.d0)) function, dimension(nline,ncol) filetools::iMatrixRead::imatrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* integer, intent(in) *ncol,* character(len=1), optional *first_,* character(len=1), optional *last_*)

Definition at line 75 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.16 inputvar Module Reference

Input variables collection.

Collaboration diagram for inputvar:

Public Attributes

- character(len=20) gridconnection
- integer ntimes
- logical ideg
- real(kind(1.d0)), dimension(:), allocatable startpoint
- real(kind(1.d0)) dt1
- character(len=100) obj
- integer ntrig
- integer, dimension(:), allocatable nspt
- integer, dimension(:), allocatable nsizet
- integer, dimension(:), allocatable netaelt
- integer, dimension(:), allocatable netatht
- integer, dimension(:), allocatable netacht
- real(kind(1.d0)), dimension(:), allocatable pmaxt
- real(kind(1.d0)), dimension(:), allocatable degratet
- real(kind(1.d0)), dimension(:), allocatable fuelcostt

- real(kind(1.d0)), dimension(:), allocatable fuellhvt
- real(kind(1.d0)), dimension(:), allocatable invt
- real(kind(1.d0)), dimension(:), allocatable lifet
- real(kind(1.d0)), dimension(:), allocatable firecostt
- real(kind(1.d0)), dimension(:), allocatable maintcostt
- real(kind(1.d0)), dimension(:,:), allocatable spt
- real(kind(1.d0)), dimension(:,:), allocatable ksizet
- real(kind(1.d0)), dimension(:,:,:), allocatable etaelt
- real(kind(1.d0)), dimension(:,:,:), allocatable etatht
- real(kind(1.d0)), dimension(:,:,:), allocatable etacht
- character(len=50), dimension(:), allocatable tect
- integer nboi
- integer, dimension(:), allocatable nspb
- integer, dimension(:), allocatable nsizeb
- integer, dimension(:), allocatable netab
- real(kind(1.d0)), dimension(:), allocatable pmaxb
- real(kind(1.d0)), dimension(:), allocatable degrateb
- real(kind(1.d0)), dimension(:), allocatable fuelcostb
- real(kind(1.d0)), dimension(:), allocatable fuellhvb
- real(kind(1.d0)), dimension(:), allocatable invb
- real(kind(1.d0)), dimension(:), allocatable lifeb
- real(kind(1.d0)), dimension(:), allocatable firecostb
- real(kind(1.d0)), dimension(:), allocatable maintcostb
- real(kind(1.d0)), dimension(:,:), allocatable spb
- real(kind(1.d0)), dimension(:,:), allocatable ksizeb
- real(kind(1.d0)), dimension(:,:,:), allocatable etab
- character(len=50), dimension(:), allocatable tecb
- integer nchi
- integer, dimension(:), allocatable nspc
- integer, dimension(:), allocatable nsizec
- integer, dimension(:), allocatable netac
- real(kind(1.d0)), dimension(:), allocatable pmaxc

- real(kind(1.d0)), dimension(:), allocatable degratec
- real(kind(1.d0)), dimension(:), allocatable invc
- real(kind(1.d0)), dimension(:), allocatable lifec
- real(kind(1.d0)), dimension(:), allocatable firecostc
- real(kind(1.d0)), dimension(:), allocatable maintcostc
- real(kind(1.d0)), dimension(:,:), allocatable spc
- real(kind(1.d0)), dimension(:,:), allocatable ksizec
- real(kind(1.d0)), dimension(:,:,:), allocatable etac
- character(len=50), dimension(:), allocatable tecc
- integer ntime
- integer nload
- integer itime
- integer iel
- integer ith
- integer ich
- integer ielp
- integer ithp
- integer ichp
- real(kind(1.d0)), dimension(:,:), allocatable uel
- real(kind(1.d0)), dimension(:,:), allocatable uth
- real(kind(1.d0)), dimension(:,:), allocatable uch
- real(kind(1.d0)), dimension(:,:), allocatable cel
- real(kind(1.d0)), dimension(:,:), allocatable cth
- real(kind(1.d0)), dimension(:,:), allocatable cch
- real(kind(1.d0)), dimension(:), allocatable time
- real(kind(1.d0)), dimension(:), allocatable gridbuycost
- real(kind(1.d0)), dimension(:), allocatable gridsellcost
- integer, dimension(:), allocatable nld
- integer, dimension(:), allocatable nlp

3.16.1 Detailed Description

This module collects all the variables read from input files. Include this module To use these variable anywhere in the code.

Author

Andrea Facci.

Definition at line 40 of file inputVar.f90.

3.16.2 Member Data Documentation

3.16.2.1 real(kind(1.d0)), dimension(:,:), allocatable inputvar::cch

Definition at line 78 of file inputVar.f90.

3.16.2.2 real(kind(1.d0)), dimension(:,:), allocatable inputvar::cel

Definition at line 78 of file inputVar.f90.

3.16.2.3 real(kind(1.d0)), dimension(:,:), allocatable inputvar::cth

Definition at line 78 of file inputVar.f90.

3.16.2.4 real(kind(1.d0)), dimension(:), allocatable inputvar::degrateb

Definition at line 62 of file inputVar.f90.

3.16.2.5 real(kind(1.d0)), dimension(:), allocatable inputvar::degratec

Definition at line 71 of file inputVar.f90.

3.16.2.6 real(kind(1.d0)), dimension(:), allocatable inputvar::degratet

Definition at line 53 of file inputVar.f90.

3.16.2.7 real(kind(1.d0)) inputvar::dt1

Definition at line 47 of file inputVar.f90.

3.16.2.8 real(kind(1.d0)), dimension(:,:,:), allocatable inputvar::etab

Definition at line 65 of file inputVar.f90.

3.16.2.9 real(kind(1.d0)), dimension(:,:,:), allocatable inputvar::etac

Definition at line 73 of file inputVar.f90.

3.16.2.10 real(kind(1.d0)), dimension(:,:,:), allocatable inputvar::etacht

Definition at line 56 of file inputVar.f90.

3.16.2.11 real(kind(1.d0)), dimension(:,:,:), allocatable inputvar::etaelt

Definition at line 56 of file inputVar.f90.

3.16.2.12 real(kind(1.d0)), dimension(:,:,:), allocatable inputvar::etatht

Definition at line 56 of file inputVar.f90.

3.16.2.13 real(kind(1.d0)), dimension(:), allocatable inputvar::firecostb

Definition at line 62 of file inputVar.f90.

3.16.2.14 real(kind(1.d0)), dimension(:), allocatable inputvar::firecostc

Definition at line 71 of file inputVar.f90.

3.16.2.15 real(kind(1.d0)), dimension(:), allocatable inputvar::firecostt

Definition at line 53 of file inputVar.f90.

3.16.2.16 real(kind(1.d0)), dimension(:), allocatable inputvar::fuelcostb

Definition at line 62 of file inputVar.f90.

3.16.2.17 real(kind(1.d0)), dimension(:), allocatable inputvar::fuelcostt

Definition at line 53 of file inputVar.f90.

3.16.2.18 real(kind(1.d0)), dimension(:), allocatable inputvar::fuellhvb

Definition at line 62 of file inputVar.f90.

3.16.2.19 real(kind(1.d0)), dimension(:), allocatable inputvar::fuellhvt

Definition at line 53 of file inputVar.f90.

3.16.2.20 real(kind(1.d0)), dimension(:), allocatable inputvar::gridbuycost

Definition at line 79 of file inputVar.f90.

3.16.2.21 character(len=20) inputvar::gridconnection

Definition at line 43 of file inputVar.f90.

3.16.2.22 real(kind(1.d0)), dimension(:), allocatable inputvar::gridsellcost

Definition at line 79 of file inputVar.f90.

3.16.2.23 integer inputvar::ich

Definition at line 77 of file inputVar.f90.

3.16.2.24 integer inputvar::ichp

Definition at line 77 of file inputVar.f90.

3.16.2.25 logical inputvar::ideg

Definition at line 45 of file inputVar.f90.

3.16.2.26 integer inputvar::iel

Definition at line 77 of file inputVar.f90.

3.16.2.27 integer inputvar::ielp

Definition at line 77 of file inputVar.f90.

3.16.2.28 real(kind(1.d0)), dimension(:), allocatable inputvar::invb

Definition at line 62 of file inputVar.f90.

3.16.2.29 real(kind(1.d0)), dimension(:), allocatable inputvar::invc

Definition at line 71 of file inputVar.f90.

3.16.2.30 real(kind(1.d0)), dimension(:), allocatable inputvar::invt

Definition at line 53 of file inputVar.f90.

3.16.2.31 integer inputvar::ith

Definition at line 77 of file inputVar.f90.

3.16.2.32 integer inputvar::ithp

Definition at line 77 of file inputVar.f90.

3.16.2.33 integer inputvar::itime

Definition at line 77 of file inputVar.f90.

3.16.2.34 real(kind(1.d0)), dimension(:,:), allocatable inputvar::ksizeb

Definition at line 64 of file inputVar.f90.

3.16.2.35 real(kind(1.d0)), dimension(:,:), allocatable inputvar::ksizec

Definition at line 72 of file inputVar.f90.

3.16.2.36 real(kind(1.d0)), dimension(:,:), allocatable inputvar::ksizet

Definition at line 55 of file inputVar.f90.

3.16.2.37 real(kind(1.d0)), dimension(:), allocatable inputvar::lifeb

Definition at line 62 of file inputVar.f90.

3.16.2.38 real(kind(1.d0)), dimension(:), allocatable inputvar::lifec

Definition at line 71 of file inputVar.f90.

3.16.2.39 real(kind(1.d0)), dimension(:), allocatable inputvar::lifet

Definition at line 53 of file inputVar.f90.

3.16.2.40 real(kind(1.d0)), dimension(:), allocatable inputvar::maintcostb

Definition at line 62 of file inputVar.f90.

3.16.2.41 real(kind(1.d0)), dimension(:), allocatable inputvar::maintcostc

Definition at line 71 of file inputVar.f90.

3.16.2.42 real(kind(1.d0)), dimension(:), allocatable inputvar::maintcostt

Definition at line 53 of file inputVar.f90.

3.16.2.43 integer inputvar::nboi

Definition at line 60 of file inputVar.f90.

3.16.2.44 integer inputvar::nchi

Definition at line 69 of file inputVar.f90.

3.16.2.45 integer, dimension(:), allocatable inputvar::netab

Definition at line 61 of file inputVar.f90.

3.16.2.46 integer, dimension(:), allocatable inputvar::netac

Definition at line 70 of file inputVar.f90.

3.16.2.47 integer, dimension(:), allocatable inputvar::netacht

Definition at line 52 of file inputVar.f90.

3.16.2.48 integer, dimension(:), allocatable inputvar::netaelt

Definition at line 52 of file inputVar.f90.

3.16.2.49 integer, dimension(:), allocatable inputvar::netatht

Definition at line 52 of file inputVar.f90.

3.16.2.50 integer, dimension(:), allocatable inputvar::nld

Definition at line 80 of file inputVar.f90.

3.16.2.51 integer inputvar::nload

Definition at line 77 of file inputVar.f90.

3.16.2.52 integer, dimension(:), allocatable inputvar::nlp

Definition at line 80 of file inputVar.f90.

3.16.2.53 integer, dimension(:), allocatable inputvar::nsizeb

Definition at line 61 of file inputVar.f90.

3.16.2.54 integer, dimension(:), allocatable inputvar::nsizec

Definition at line 70 of file inputVar.f90.

3.16.2.55 integer, dimension(:), allocatable inputvar::nsizet

Definition at line 52 of file inputVar.f90.

3.16.2.56 integer, dimension(:), allocatable inputvar::nspb

Definition at line 61 of file inputVar.f90.

3.16.2.57 integer, dimension(:), allocatable inputvar::nspc

Definition at line 70 of file inputVar.f90.

3.16.2.58 integer, dimension(:), allocatable inputvar::nspt

Definition at line 52 of file inputVar.f90.

3.16.2.59 integer inputvar::ntime

Definition at line 77 of file inputVar.f90.

3.16.2.60 integer inputvar::ntimes

Definition at line 44 of file inputVar.f90.

3.16.2.61 integer inputvar::ntrig

Definition at line 51 of file inputVar.f90.

3.16.2.62 character(len=100) inputvar::obj

Definition at line 48 of file inputVar.f90.

3.16.2.63 real(kind(1.d0)), dimension(:), allocatable inputvar::pmaxb

Definition at line 62 of file inputVar.f90.

3.16.2.64 real(kind(1.d0)), dimension(:), allocatable inputvar::pmaxc

Definition at line 71 of file inputVar.f90.

3.16.2.65 real(kind(1.d0)), dimension(:), allocatable inputvar::pmaxt

Definition at line 53 of file inputVar.f90.

3.16.2.66 real(kind(1.d0)), dimension(:,:), allocatable inputvar::spb

Definition at line 64 of file inputVar.f90.

3.16.2.67 real(kind(1.d0)), dimension(:,:), allocatable inputvar::spc

Definition at line 72 of file inputVar.f90.

3.16.2.68 real(kind(1.d0)), dimension(:,:), allocatable inputvar::spt

Definition at line 55 of file inputVar.f90.

3.16.2.69 real(kind(1.d0)), dimension(:), allocatable inputvar::startpoint

Definition at line 46 of file inputVar.f90.

3.16.2.70 character(len=50), dimension(:), allocatable inputvar::tecb

Definition at line 66 of file inputVar.f90.

3.16.2.71 character(len=50), dimension(:), allocatable inputvar::tecc

Definition at line 74 of file inputVar.f90.

3.16.2.72 character(len=50), dimension(:), allocatable inputvar::tect

Definition at line 57 of file inputVar.f90.

3.16.2.73 real(kind(1.d0)), dimension(:), allocatable inputvar::time

Definition at line 79 of file inputVar.f90.

3.16.2.74 real(kind(1.d0)), dimension(:,:), allocatable inputvar::uch

Definition at line 78 of file inputVar.f90.

3.16.2.75 real(kind(1.d0)), dimension(:,:), allocatable inputvar::uel

Definition at line 78 of file inputVar.f90.

3.16.2.76 real(kind(1.d0)), dimension(:,:), allocatable inputvar::uth

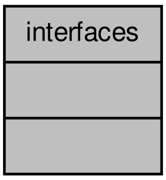
Definition at line 78 of file inputVar.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/inputVar.f90

3.17 interfaces Module Reference

Collaboration diagram for interfaces:



Data Types

- interface abortExecution
- interface warning

3.17.1 Detailed Description

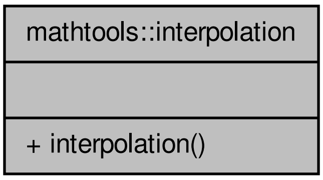
Definition at line 35 of file interfaces.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/interfaces.f90

3.18 mathtools::interpolation Interface Reference

Collaboration diagram for mathtools::interpolation:



Public Member Functions

- real(kind(1.d0)) function,
  dimension(m) interpolation (xIn, yIn, n, xOut, m, warn)

3.18.1 Detailed Description

Definition at line 39 of file mathTools.f90.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 real(kind(1.d0)) function, dimension(m) mathtools::interpolation::interpolation (real(kind(1.d0)), dimension(n), intent(in) *xIn,* real(kind(1.d0)), dimension(n), intent(in) *yIn,* integer, intent(in) *n,* real(kind(1.d0)), dimension(m), intent(in) *xOut,* integer, intent(in) *m,* integer, dimension(2), intent(in), optional *warn*)

Definition at line 39 of file mathTools.f90.

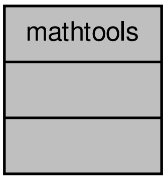
The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/mathTools.f90

3.19 mathtools Module Reference

Collection of interfaces for basic mathematical tools.

Collaboration diagram for mathtools:



Data Types

- interface interpolation

3.19.1 Detailed Description

Collection of interfaces for basic mathematical tools.

Author

Andrea Facci.

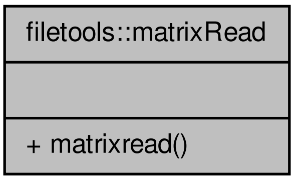
Definition at line 36 of file mathTools.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/mathTools.f90

3.20 filetools::matrixRead Interface Reference

Collaboration diagram for filetools::matrixRead:



Public Member Functions

- character(len=100) function,
  dimension(nline) matrixread (theUnit, nline, first_, last_)

3.20.1 Detailed Description

Definition at line 93 of file fileTools.f90.

3.20.2 Member Function/Subroutine Documentation

3.20.2.1 character(len=100) function, dimension(nline) filetools::matrixRead::matrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* character(len=1), optional *first_,* character(len=1), optional *last_*)

Definition at line 93 of file fileTools.f90.

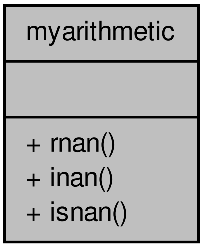
The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.21 myarithmetic Module Reference

Creates and detects NaN.

Collaboration diagram for myarithmetic:



Public Member Functions

- real(kind(1.d0)) function rnan (x)

  Creates NaN.
- integer function inan (x)

  Creates NaN.
- logical function isnan (x)

  Detects NaNs.

3.21.1 Detailed Description

This module collects some subroutine useful to create and detect NaNs. It trys to imitate the IEEE_ARITHMETIC module that unfortunately is still not available for the gfortran compiler.

Author

Definition at line 36 of file myArithmetic.f90.

3.21.2 Member Function/Subroutine Documentation

3.21.2.1 integer function myarithmetic::inan (integer, intent(in) *x*)

Creates a NaN to be associated to an integer, variable.

Parameters

in	*x*	random integer number.

Author

Andrea Facci.

Definition at line 59 of file myArithmetic.f90.

Referenced by graphtools::grapharcs().

3.21.2.2 logical function myarithmetic::isnan (*x*)

This subroutine determine if a value is NaN.

Parameters

in	*x*	the value to be tested.

Author

Andrea Facci

Definition at line 70 of file myArithmetic.f90.

3.21.2.3 real(kind(1.d0)) function myarithmetic::rnan (real(kind(1.d0)), intent(in) *x*)

Creates a NaN to be associated to a real, double precition variable.

Parameters

in	*x*	random double precision number.

Author

Andrea Facci.

Definition at line 45 of file myArithmetic.f90.

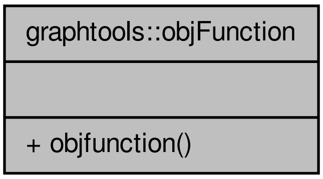
Referenced by graphtools::grapharcs().

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/myArithmetic.f90

3.22 graphtools::objFunction Interface Reference

Collaboration diagram for graphtools::objFunction:



Public Member Functions

- real(kind(1.d0)) function objfunction (c, t, obj)

3.22.1 Detailed Description

Definition at line 50 of file graphTools.f90.

3.22.2 Member Function/Subroutine Documentation

3.22.2.1 real(kind(1.d0)) function graphtools::objFunction::objfunction (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*, character(len=100), intent(in) *obj*)

Definition at line 50 of file graphTools.f90.

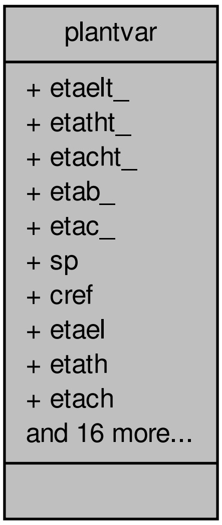
The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/graphTools.f90

3.23 plantvar Module Reference

collection of variables relative to the power plant structure.

Collaboration diagram for plantvar:



Public Attributes

- real(kind(1.d0)), dimension(:,:), allocatable etaelt_
- real(kind(1.d0)), dimension(:,:), allocatable etatht_
- real(kind(1.d0)), dimension(:,:), allocatable etacht_
- real(kind(1.d0)), dimension(:,:), allocatable etab_
- real(kind(1.d0)), dimension(:,:), allocatable etac_
- real(kind(1.d0)), dimension(:,:), allocatable sp
- real(kind(1.d0)), dimension(:,:), allocatable cref
- real(kind(1.d0)), dimension(:,:), allocatable etael
- real(kind(1.d0)), dimension(:,:), allocatable etath
- real(kind(1.d0)), dimension(:,:), allocatable etach
- integer, dimension(:,:), allocatable cr
- real(kind(1.d0)), dimension(:), allocatable pmax
- real(kind(1.d0)), dimension(:), allocatable dt
- real(kind(1.d0)), dimension(:), allocatable cf

- real(kind(1.d0)), dimension(:), allocatable lhv
- real(kind(1.d0)), dimension(:), allocatable onoffcost
- real(kind(1.d0)), dimension(:), allocatable oemcost
- character(len=4), dimension(:), allocatable pes
- integer nsptot
- integer nm
- integer it
- integer ib
- integer ic
- integer, dimension(3) is
- integer, dimension(3) ie
- integer, dimension(:), allocatable nsp

3.23.1 Detailed Description

collection of variables relative to the power plant structure.

Author

Definition at line 35 of file plantVar.f90.

3.23.2 Member Data Documentation

3.23.2.1 real(kind(1.d0)), dimension(:), allocatable plantvar::cf

Definition at line 41 of file plantVar.f90.

3.23.2.2 integer, dimension(:,:), allocatable plantvar::cr

Definition at line 40 of file plantVar.f90.

3.23.2.3 real(kind(1.d0)), dimension(:,:), allocatable plantvar::cref

Definition at line 37 of file plantVar.f90.

3.23.2.4 real(kind(1.d0)), dimension(:), allocatable plantvar::dt

Definition at line 41 of file plantVar.f90.

3.23.2.5 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etab_

Definition at line 37 of file plantVar.f90.

3.23.2.6 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etac_

Definition at line 37 of file plantVar.f90.

3.23.2.7 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etach

Definition at line 39 of file plantVar.f90.

3.23.2.8 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etacht_

Definition at line 37 of file plantVar.f90.

3.23.2.9 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etael

Definition at line 39 of file plantVar.f90.

3.23.2.10 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etaelt_

Definition at line 37 of file plantVar.f90.

3.23.2.11 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etath

Definition at line 39 of file plantVar.f90.

3.23.2.12 real(kind(1.d0)), dimension(:,:), allocatable plantvar::etatht_

Definition at line 37 of file plantVar.f90.

3.23.2.13 integer plantvar::ib

Definition at line 43 of file plantVar.f90.

3.23.2.14 integer plantvar::ic

Definition at line 43 of file plantVar.f90.

3.23.2.15 integer, dimension(3) plantvar::ie

Definition at line 44 of file plantVar.f90.

3.23.2.16 integer, dimension(3) plantvar::is

Definition at line 44 of file plantVar.f90.

3.23.2.17 integer plantvar::it

Definition at line 43 of file plantVar.f90.

3.23.2.18 real(kind(1.d0)), dimension(:), allocatable plantvar::lhv

Definition at line 41 of file plantVar.f90.

3.23.2.19 integer plantvar::nm

Definition at line 43 of file plantVar.f90.

3.23.2.20 integer, dimension(:), allocatable plantvar::nsp

Definition at line 45 of file plantVar.f90.

3.23.2.21 integer plantvar::nsptot

Definition at line 43 of file plantVar.f90.

3.23.2.22 real(kind(1.d0)), dimension(:), allocatable plantvar::oemcost

Definition at line 41 of file plantVar.f90.

3.23.2.23 real(kind(1.d0)), dimension(:), allocatable plantvar::onoffcost

Definition at line 41 of file plantVar.f90.

3.23.2.24 character(len=4), dimension(:), allocatable plantvar::pes

Definition at line 42 of file plantVar.f90.

3.23.2.25 real(kind(1.d0)), dimension(:), allocatable plantvar::pmax

Definition at line 41 of file plantVar.f90.

3.23.2.26 real(kind(1.d0)), dimension(:,:), allocatable plantvar::sp

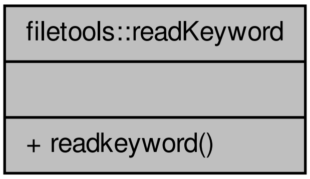
Definition at line 37 of file plantVar.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/FortranCode/src/plantVar.f90

3.24 filetools::readKeyword Interface Reference

Collaboration diagram for filetools::readKeyword:



Public Member Functions

- subroutine readkeyword (theUnit, rew, keyword, value, error, nRow)

3.24.1 Detailed Description

Definition at line 153 of file fileTools.f90.

3.24.2 Member Function/Subroutine Documentation

3.24.2.1 subroutine filetools::readKeyword::readkeyword (integer, intent(in) *theUnit,* logical, intent(in), optional *rew,* character(len=100), intent(out) *keyword,* character(len=100), intent(out) *value,* integer, intent(out), optional *error,* integer, intent(out), optional *nRow*)

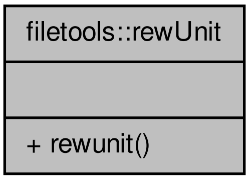
Definition at line 153 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.25 filetools::rewUnit Interface Reference

Collaboration diagram for filetools::rewUnit:



Public Member Functions

- subroutine rewunit (theUnit, n)

3.25.1 Detailed Description

Definition at line 42 of file fileTools.f90.

3.25.2 Member Function/Subroutine Documentation

3.25.2.1 subroutine filetools::rewUnit::rewunit (integer, intent(in) *theUnit,* integer, intent(in) *n*)

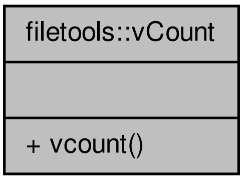
Definition at line 42 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.26 filetools::vCount Interface Reference

Collaboration diagram for filetools::vCount:



Public Member Functions

- integer function vcount (theUnit, rew_, first_, last_)

3.26.1 Detailed Description

Definition at line 49 of file fileTools.f90.

3.26.2 Member Function/Subroutine Documentation

3.26.2.1 integer function filetools::vCount::vcount (integer, intent(in) *theUnit,* logical, optional *rew_,* character(len=1), intent(in), optional *first_,* character(len=1), intent(in), optional *last_*)

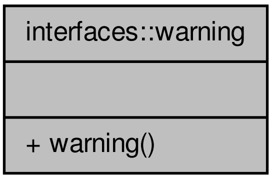
Definition at line 49 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/fileTools.f90

3.27 interfaces::warning Interface Reference

Collaboration diagram for interfaces::warning:



Public Member Functions

- subroutine warning (i, j, k, line, word, r1, r2)

3.27.1 Detailed Description

Definition at line 48 of file interfaces.f90.

3.27.2 Constructor & Destructor Documentation

3.27.2.1 subroutine interfaces::warning::warning (integer, intent(in), optional *i,* integer, intent(in), optional *j,* integer, intent(in), optional *k,* integer, intent(in), optional *line,* character(len=∗), intent(in), optional *word,* real(kind(1.d0)), intent(in), optional *r1,* real(kind(1.d0)), intent(in), optional *r2*)

Definition at line 48 of file interfaces.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/FortranCode/src/interfaces.f90

Chapter 4

File Documentation

4.1 /home/Codici/Blink/FortranCode/src/abortExecution.f90 File Reference

terminates the execution in case of error.

Functions/Subroutines

- subroutine abortexecution (i, j, line, word, r1, r2)

  abort the program execution.

4.1.1 Detailed Description

```
aborts the program exectuion in case of error and print the error message
according to the error code given as input.
```

Author

Andrea Facci.

Definition in file abortExecution.f90.

4.1.2 Function/Subroutine Documentation

4.1.2.1 subroutine abortexecution (integer, intent(in), optional *i,* integer, intent(in), optional *j,* integer, intent(in), optional *line,* character(len=∗), intent(in), optional *word,* real(kind(1.d0)), intent(in), optional *r1,* real(kind(1.d0)), intent(in), optional *r2*)

aborts the program exectuion in case of error and print the error message according to the error code given as input.

Parameters

`in`	*i,j*	integers that identify the error.
`in`	*line*	integer that identifies the line affected by the mistake in an input file
`in`	*word*	character input to report the mispelled or unexpected sentence
`in`	*r1,r2*	double precision reals. Useful to report incoherence between plant parameters

Definition at line 42 of file abortExecution.f90.

Referenced by checkplant(), readboiler(), readchillers(), readgeneral(), readloads(), and readtrigen().

4.2 /home/Codici/Blink/FortranCode/src/aiuto.f90 File Reference

prints a very short help.

Functions/Subroutines

- subroutine aiuto

  prints a very short help.

4.2.1 Detailed Description

prints a very short help.

Author

Andrea Facci.

Definition in file aiuto.f90.

4.2.2 Function/Subroutine Documentation

4.2.2.1 subroutine aiuto ( )

prints a very short help. This is called if -help option is given from the command line.

Author

Andrea Facci.

Definition at line 39 of file aiuto.f90.

References endexecution().

Referenced by commandline().

4.3 /home/Codici/Blink/FortranCode/src/allocateVar.f90 File Reference

variable allocation.

Functions/Subroutines

- subroutine allocatevar (what, num)

  variable allocation

4.3.1 Detailed Description

Author

Definition in file allocateVar.f90.

4.3.2 Function/Subroutine Documentation

4.3.2.1 subroutine allocatevar (integer, intent(in) *what*, integer, intent(in), optional *num*)

Allocates the veraibles according to the inputs. All the allocation of "global" variables should be done here.

Parameters

`in`	*what*	defines the variables to be allocated
`in`	*num*	defines the dimnesion of the array

Author

Andrea Facci

Definition at line 36 of file allocateVar.f90.

Referenced by buildplant(), readboiler(), readchillers(), readgeneral(), readloads(), and readtrigen().

4.4 /home/Codici/Blink/FortranCode/src/buildPlant.f90 File Reference

Collects all the informations relative to the power plant.

Functions/Subroutines

- subroutine buildplant

  Collects all the informations relative to the power plant.

4.4.1 Detailed Description

Author

Definition in file buildPlant.f90.

4.4.2 Function/Subroutine Documentation

4.4.2.1 subroutine buildplant ( )

Collects all the informations relative to the power plant starting from the Inputs of the single machinery. Calculates the efficiencies for each given set-point and store them in a single array. Perform units of measure conversions when necessary.

Author

Definition at line 40 of file buildPlant.f90.

References allocatevar(), and checkplant().

Referenced by main().

4.5 /home/Codici/Blink/FortranCode/src/checkPlant.f90 File Reference

Check the power plant coherence with the energy demand.

Functions/Subroutines

- subroutine checkplant

  Check the power plant coherence with the energy demand.

4.5.1 Detailed Description

Author

Definition in file checkPlant.f90.

4.5.2 Function/Subroutine Documentation

4.5.2.1 subroutine checkplant ( )

Check the power plant coherence with the energy demand. Specifically the maximum thermal and chilling power must be higher than the relative energy demand. Thermal load includes also themal energy needed by absorption chillers. If the plant is not grid connected also rated electrical power must be grater than maximum electrical demand.

Author

Definition at line 41 of file checkPlant.f90.

References abortexecution().

Referenced by buildplant().

4.6 /home/Codici/Blink/FortranCode/src/cmdVar.f90 File Reference

Collects the variable read from command line.

Data Types

- module cmdvar

  Collects the variable read from command line.

4.6.1 Detailed Description

Collects the variable read from command line.

Author

Definition in file cmdVar.f90.

4.7 /home/Codici/Blink/FortranCode/src/commandline.f90 File Reference

Reads the commad line.

Functions/Subroutines

- subroutine commandline

4.7.1 Detailed Description

subroutine to read the execution options from command line.

Author

Andrea Facci

Definition in file commandline.f90.

4.7.2 Function/Subroutine Documentation

4.7.2.1 subroutine commandline ( )

Definition at line 35 of file commandline.f90.

References aiuto().

Referenced by main().

4.8 /home/Codici/Blink/FortranCode/src/constraints.f90 File Reference

static constraints.

Functions/Subroutines

- logical function constraints (c, t)

  static constraints

4.8.1 Detailed Description

This file contains a subroutine that checks if the plant respects the energy constraints for a given set-point and time-step.

Definition in file constraints.f90.

4.8.2 Function/Subroutine Documentation

4.8.2.1 logical function constraints (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

This function checks if the plant respects the energy constraints for a given set-point and time-step. The constraints are:

- Thermal Power: $\sum U_{th} + U_{th}^{self} \leq \sum P_{th}$

- Chilling Power: $\sum U_{ch} \leq \sum P_{ch}$
- Electrical Power : $\sum U_{el} + U_{el}^{self} \leq \sum P_{el}$ only if the pant is bnot grid connected

Parameters

in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the

Author

Andrea Facci

Definition at line 46 of file constraints.f90.

References energy::chprod(), energy::thprod(), and energy::thselfcons().

4.9 /home/Codici/Blink/FortranCode/src/economy.f90 File Reference

costs and revenues calulation prodedures.

Data Types

- module economy

4.9.1 Detailed Description

This module contains the definition of all the procedures that perform economic calculations for a give set-point and time-step, that are, fuel costs, O&M costs, and the revenues from thermal, electric, and chilling, energy selling.

Definition in file economy.f90.

4.10 /home/Codici/Blink/FortranCode/src/endExecution.f90 File Reference

normal termination of the execution.

Functions/Subroutines

- subroutine endexecution

  normal termination of the execution.

4.10.1 Detailed Description

Author

Definition in file endExecution.f90.

4.10.2 Function/Subroutine Documentation

4.10.2.1 subroutine endexecution ( )

This procedure is called for the normal termination of the program execution.

Author

Definition at line 35 of file endExecution.f90.

Referenced by aiuto(), and main().

4.11 /home/Codici/Blink/FortranCode/src/energy.f90 File Reference

Collection of function for energy flow calculation.

Data Types

- module energy

  module for energy calculations.

4.11.1 Detailed Description

This file contains a module that collects all the procedure for energy calculations

Author

Andrea Facci.

Definition in file energy.f90.

4.12 /home/Codici/Blink/FortranCode/src/fileTools.f90 File Reference

collection of proceture interfaces useful to read from files.

Data Types

- module filetools

  Interfaces of procedures to read from file.
- interface filetools::rewUnit
- interface filetools::vCount
- interface filetools::hCount
- interface filetools::dMatrixRead
- interface filetools::iMatrixRead
- interface filetools::cMatrixRead
- interface filetools::matrixRead
- interface filetools::iFindEntry
- interface filetools::dFindEntry
- interface filetools::cFindEntry
- interface filetools::findEntry
- interface filetools::readKeyword

4.12.1 Detailed Description

Author

Definition in file fileTools.f90.

4.13 /home/Codici/Blink/FortranCode/src/findEntry2.f90 File Reference

Collection of procetures to find a specific entry inside a file.

Functions/Subroutines

- subroutine ifindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

  Finds an entry with integer value.
- subroutine dfindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

  Finds an entry with double precision value.
- subroutine cfindentry (entry, n, theUnit, rew, valore, isPresent, nRow)

  Finds an entry with character value.
- subroutine findentry (entry, theUnit, rew, valore, isPresent, nRow)

  Finds an entry and returns the vaule as it is.

4.13.1 Detailed Description

Author

Definition in file findEntry2.f90.

4.13.2 Function/Subroutine Documentation

4.13.2.1 subroutine cfindentry (character(len=100), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

Use this subroutine to find a specific entry in a file, when an character value is expected. in the value field, that is between the two "|". The string in the value field is associated to a character array of dimension "n" Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file. Note that the maximum length of each element of the "value" vector is 100.

Parameters

in	*entry*	the keyword to be located
in	*n*	the number of elements expected for the value field
in	*theUnit*	the unit corresponding to the file to be searched
in	*rew*	wether the unit is to be rewinded or not
out	*valore*	the vaule correspondind to keyword that is searched for
out	*isPresent*	logical that tells if the entry is present or not
out	*nRow*	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 185 of file findEntry2.f90.

References findentry().

Referenced by readchillers().

4.13.2.2 subroutine dfindentry (character(len=100), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, real(kind(1.d0)), dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

Use this subroutine to find a specific entry in a file, when a double precision value is expected. in the value field, that is between the two "|". The string in the value field is associated to double precision array of dimension "n" Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	*entry*	the keyword to be located
in	*n*	the number of elements expected for the value field
in	*theUnit*	the unit corresponding to the file to be searched
in	*rew*	wether the unit is to be rewinded or not
out	*valore*	the vaule correspondind to keyword that is searched for
out	*isPresent*	logical that tells if the entry is present or not
out	*nRow*	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 115 of file findEntry2.f90.

References findentry().

4.13.2.3 subroutine findentry (character(len=100), intent(in) *entry*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

Use this subroutine to find a specific entry in a file. If given as parameter returns the "value" field, that is the value comprised between the two "|" in the input file. This subroutine does not associate the vaule fied to any array but returns a single strig axactly as it is in the input file. Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	*entry*	the keyword to be located
in	*n*	the number of elements expected for the value field
in	*theUnit*	the unit corresponding to the file to be searched
in	*rew*	wether the unit is to be rewinded or not
out	*valore*	the vaule correspondind to keyword that is searched for
out	*isPresent*	logical that tells if the entry is present or not
out	*nRow*	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 253 of file findEntry2.f90.

References rewunit().

Referenced by cfindentry(), dfindentry(), ifindentry(), and readloads().

4.13.2.4 subroutine ifindentry (character(len=100), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, integer, dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

Use this subroutine to find a specific entry in a file, when an integer value is expected in the value field, that is between the two "|". The string in the value field is associated to an integer array of dimension "n" Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	*entry*	the keyword to be located
in	*n*	the number of elements expected for the value field
in	*theUnit*	the unit corresponding to the file to be searched
in	*rew*	wether the unit is to be rewinded or not
out	*valore*	the vaule correspondind to keyword that is searched for
out	*isPresent*	logical that tells if the entry is present or not
out	*nRow*	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 47 of file findEntry2.f90.

References findentry().

Referenced by readboiler(), readchillers(), and readtrigen().

4.14 /home/Codici/Blink/FortranCode/src/graphTools.f90 File Reference

graph construction and minumum path.

Data Types

- module graphtools
- interface graphtools::objFunction
- interface graphtools::constraints

4.14.1 Detailed Description

this file implements a module (graphTools) that contains all the routines necessary to build the graph representing the problem and find the minumum path across the graph. The graph is acyclic (no closed paths) and represented in topological ordering using a predecessor list.

Author

Andrea Facci.

Definition in file graphTools.f90.

4.15 /home/Codici/Blink/FortranCode/src/hCount.f90 File Reference

Count the number of elements of a vector in a text file.

Functions/Subroutines

- integer function hcount (value_, first_, last_)

  Count the number of elements of a vector in a text file.

4.15.1 Detailed Description

Author

Andrea Facci.

Definition in file hCount.f90.

4.15.2 Function/Subroutine Documentation

4.15.2.1 integer function hcount (character(len=100), intent(in) *value_*, character(len=1), optional *first_*, character(len=1), optional *last_*)

Use This subroutine to determine the length of a vector, that is a series of values encloesd between two delimiters, when reading from text file. The vector must be specified as a single string input (max length=100), while delimiters may be specified as single character input or left to default that is "(" for opening and ")" for closing.

Parameters

in	*value_*	the string containing teh vector whose lenght is to be determined
in	*first_,last_*	sigle character delimiters of the vector (open and close respectively). If not provided "(" ad ")" will be assumed as defaults

Author

Andrea Facci.

Definition at line 43 of file hCount.f90.

Referenced by readboiler(), readchillers(), readloads(), and readtrigen().

4.16 /home/Codici/Blink/FortranCode/src/inputVar.f90 File Reference

Input variables collection.

Data Types

- module inputvar

Input variables collection.

4.16.1 Detailed Description

Author

Definition in file inputVar.f90.

4.17 /home/Codici/Blink/FortranCode/src/interfaces.f90 File Reference

Collection of general pourpose interfaces.

Data Types

- module interfaces
- interface interfaces::abortExecution
- interface interfaces::warning

4.17.1 Detailed Description

Author

Definition in file interfaces.f90.

4.18 /home/Codici/Blink/FortranCode/src/interpolation.f90 File Reference

Remaps a discrete scalar field on a given 1d grid.

Functions/Subroutines

- real(kind(1.d0)) function,
  dimension(m) interpolation (xIn, yIn, n, xOut, m, warn)

  Remaps a discrete scalar field on a given 1d grid.

4.18.1 Detailed Description

Author

Andrea Facci.

Definition in file interpolation.f90.

4.18.2 Function/Subroutine Documentation

4.18.2.1 real(kind(1.d0)) function, dimension(m) interpolation (real(kind(1.d0)), dimension(n), intent(in) *xIn,* real(kind(1.d0)), dimension(n), intent(in) *yIn,* integer, intent(in) *n,* real(kind(1.d0)), dimension(m), intent(in) *xOut,* integer, intent(in) *m,* integer, dimension(2), intent(out), optional *warn*)

This subroutine takes a discrete scalar field defined over a 1d grid and remaps it on another 1d mesh given as input. If any of the values in the new mesh is outside the range defined by the oiginal grid, values are extrapolated and an optional warning code is returned.

Parameters

in	*xIn,yIn*	grid and values of the 1d field to be mapped
in	*n*	number of elements of the discrete field (size(xIn))
in	*xOut*	1d grid where the field is sampled
in	*m*	number of elements of the interpolation grid (xOut)
out	*warn*	warning code.

Author

Andrea Facci.

Definition at line 46 of file interpolation.f90.

4.19 /home/Codici/Blink/FortranCode/src/main.f90 File Reference

this is the main driver.

Functions/Subroutines

- program main

  this is the main driver.

4.19.1 Detailed Description

```
main driver
```

Author

Andrea Facci.

Definition in file main.f90.

4.19.2 Function/Subroutine Documentation

4.19.2.1 program main ( )

```
main driver
```

Author

Andrea Facci.

Definition at line 37 of file main.f90.

References graphtools::allcombin(), buildplant(), commandline(), endexecution(), graphtools::grapharcs(), graphtools::graphpoints(), graphtools::minpathtopo(), readboiler(), readchillers(), readgeneral(), readloads(), and readtrigen().

4.20 /home/Codici/Blink/FortranCode/src/mathTools.f90 File Reference

Collection of interfaces for basic mathematical tools.

Data Types

- module mathtools

  Collection of interfaces for basic mathematical tools.
- interface mathtools::interpolation

4.20.1 Detailed Description

Author

Andrea Facci.

Definition in file mathTools.f90.

4.21 /home/Codici/Blink/FortranCode/src/matrixRead2.f90 File Reference

reads 2D array of values from a file

Functions/Subroutines

- real(kind(1.d0)) function,
  dimension(nline, ncol) dmatrixread (theUnit, nline, ncol, first_, last_)

  reads 2D array of double precision values from a file.
- integer function, dimension(nline,
  ncol) imatrixread (theUnit, nline, ncol, first_, last_)

  reads 2D array of integer values from a file.
- character(len=100) function,
  dimension(nline, ncol) cmatrixread (theUnit, nline, ncol, first_, last_)

  reads 2D array of character values from a file.
- character(len=100) function,
  dimension(nline) matrixread (theUnit, nline, first_, last_)

  reads 2D array from a file.

4.21.1 Detailed Description

Collection of procedures to read 2D arrays from files. Each function associates the values to a different data type (integer, double, character). the first letter of the function name indicates the data type.

Author

Andrea Facci.

Definition in file matrixRead2.f90.

4.21.2 Function/Subroutine Documentation

4.21.2.1 character(len=100) function, dimension(nline,ncol) cmatrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* integer, intent(in) *ncol,* character(len=1), optional *first_,* character(len=1), optional *last_*)

This subroutine reads a 2D array of character values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two
   end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the defult values that are "(" ande ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} Scrudge & DonaldDuck \\ Goofy & MickeyMouse \end{bmatrix}$$

is:

```
   begin
      !Comment one
      exArr  | (Scrudge DonadDuck | !Comment two
             |  Goofy MikeyMouse) |
   end
```

or equivalently:

```
   begin
      !Comment one
      exArr  |        | !Comment two
             | (Scrudge DonaldDuck |
             |  Goofy   MickeyMouse)|
   end
```

so that each line in the text represnts a row of the vector output and coluns are space separated.

Parameters

in	*TheUnit*	the unit to be read.
in	*nline*	The number of rows of the 2D array
in	*ncol*	The number of columns of the 2D array
in	*first_*	the opening delimiter of the array. Default is "("
in	*last_*	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 286 of file matrixRead2.f90.

References matrixread().

Referenced by readloads().

4.21.2.2 real(kind(1.d0)) function, dimension(nline,ncol) dmatrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* integer, intent(in) *ncol,* character(len=1), optional *first_,* character(len=1), optional *last_*)

This subroutine reads a 2D array of double precision values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two
   end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the deful values that are "(" ande ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \left[\begin{array}{cc} 1.1 & 1.2 \\ 2.1 & 2.2 \end{array} \right]$$

is:

```
   begin
      !Comment one
      exArr  | (1.1 1.2 | !Comment two
             |  2.1 2.2)|
   end
```

or equivalently:

```
   begin
      !Comment one
      exArr  |         | !Comment two
             | (1.1 1.2 |
             |  2.1 2.2)|
   end
```

so that each line in the text represnts a row of the vector output and coluns are space separated.

Parameters

in	*TheUnit*	the unit to be read.
in	*nline*	The number of rows of the 2D array
in	*ncol*	The number of columns of the 2D array
in	*first_*	the opening delimiter of the array. Default is "("
in	*last_*	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 86 of file matrixRead2.f90.

References matrixread().

Referenced by readboiler(), readchillers(), readloads(), and readtrigen().

4.21.2.3 integer function, dimension(nline,ncol) imatrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* integer, intent(in) *ncol,* character(len=1), optional *first_,* character(len=1), optional *last_*)

This subroutine reads a 2D array of integer values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two
   end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the defult values that are "(" ande ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} 11 & 12 \\ 21 & 22 \end{bmatrix}$$

is:

```
   begin
      !Comment one
      exArr  | (11 12 | !Comment two
             |  21 22)|
   end
```

or equivalently:

```
   begin
      !Comment one
      exArr  |        | !Comment two
             | (11 12 |
             |  21 22)|
   end
```

so that each line in the text represnts a row of the vector output and coluns are space separated.

Parameters

in	*TheUnit*	the unit to be read.
in	*nline*	The number of rows of the 2D array
in	*ncol*	The number of columns of the 2D array
in	*first_*	the opening delimiter of the array. Default is "("
in	*last_*	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 187 of file matrixRead2.f90.

References matrixread().

4.21.2.4 character(len=100) function, dimension(nline) matrixread (integer, intent(in) *theUnit,* integer, intent(in) *nline,* character(len=1), optional *first_,* character(len=1), optional *last_*)

This subroutine reads a 2D array of directly from a specified unit. This procedure does not associate the elements of the text array to any data type. On the contrary each line in the text array is associated to a row of a character type row vector, as it is. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two
   end
```

The unit must be already opened and associated to the desired file. Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the defult values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

Parameters

in	*TheUnit*	the unit to be read.
in	*nline*	The number of rows of the 2D array
in	*first_*	the opening delimiter of the array. Default is "("
in	*last_*	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 359 of file matrixRead2.f90.

References rewunit().

Referenced by cmatrixread(), dmatrixread(), imatrixread(), and readloads().

4.22 /home/Codici/Blink/FortranCode/src/myArithmetic.f90 File Reference

creates and detects NaN

Data Types

- module myarithmetic

  Creates and detects NaN.

4.22.1 Detailed Description

Author

Definition in file myArithmetic.f90.

4.23 /home/Codici/Blink/FortranCode/src/objFunction.f90 File Reference

Objective function.

Functions/Subroutines

- real(kind(1.d0)) function objfunction (c, t, obj)

  Objective function.

4.23.1 Detailed Description

Author

Definition in file objFunction.f90.

4.23.2 Function/Subroutine Documentation

4.23.2.1 real(kind(1.d0)) function objfunction (integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*, character(len=100), intent(in) *obj*)

Returns the value of the objective function for a given plant state, time step and optimization criterion. This procedure accounts only for functions that are local in time, that is, that are function only of the plant state at time t and not at time $>$ t, neither at time $<$ t.

Parameters

in	*obj*	the optimization criterion.
in	*c*	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c\_(i))$
in	*t*	time step index. Note t=x meas the x'th time step from the

Definition at line 39 of file objFunction.f90.

References economy::currcost().

Referenced by graphtools::graphpoints().

4.24 /home/Codici/Blink/FortranCode/src/openUnit.f90 File Reference

checks file presence and opens the unit.

Functions/Subroutines

- subroutine openunit (fl, unt, prsnt)

  checks file presence and opens the unit.

4.24.1 Detailed Description

Author

Andrea Facci.

Definition in file openUnit.f90.

4.24.2 Function/Subroutine Documentation

4.24.2.1 subroutine openunit (character(len=∗), intent(in) *fl*, integer, intent(in) *unt*, logical, intent(out) *prsnt*)

checks file presence and opens the unit.

Parameters

`in`	*fl*	The file name
`in`	*unt*	The unit number
`out`	*prsnt*	Logical for file presence.

Author

Andrea Facci.

Definition at line 37 of file openUnit.f90.

Referenced by readloads().

4.25 /home/Codici/Blink/FortranCode/src/plantVar.f90 File Reference

collection of variables relative to the power plant structure.

Data Types

- module plantvar

  collection of variables relative to the power plant structure.

4.25.1 Detailed Description

Author

Definition in file plantVar.f90.

4.26 /home/Codici/Blink/FortranCode/src/prototipo.f90 File Reference

File prototype.

Functions/Subroutines

- subroutine implicit none

4.26.1 Detailed Description

this is the prototype for all the files of the PowerManger project. Copy, rename, and modify this this file to create a new procedure or module. Andrea Facci.

Definition in file prototipo.f90.

4.26.2 Function/Subroutine Documentation

4.26.2.1 subroutine implicit (*none*)

Definition at line 32 of file prototipo.f90.

4.27 /home/Codici/Blink/FortranCode/src/readBoiler.f90 File Reference

Reads Boiler.inp file.

Functions/Subroutines

- subroutine readboiler

  Reads Boiler.inp file.

4.27.1 Detailed Description

Author

Andrea Facci.

Definition in file readBoiler.f90.

4.27.2 Function/Subroutine Documentation

4.27.2.1 subroutine readboiler ( )

This subroutine reads the file "Boilers.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readBoiler.f90.

References abortexecution(), allocatevar(), dmatrixread(), hcount(), ifindentry(), readkeyword(), rewunit(), and vcount().

Referenced by main().

4.28 /home/Codici/Blink/FortranCode/src/readChiller.f90 File Reference

Reads Chiller.inp file.

Functions/Subroutines

- subroutine readchillers

  Reads Chiller.inp file.

4.28.1 Detailed Description

Author

Andrea Facci.

Definition in file readChiller.f90.

4.28.2 Function/Subroutine Documentation

4.28.2.1 subroutine readchillers ( )

This subroutine reads the file "Chiller.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readChiller.f90.

References abortexecution(), allocatevar(), cfindentry(), dmatrixread(), hcount(), ifindentry(), readkeyword(), rewunit(), and vcount().

Referenced by main().

4.29 /home/Codici/Blink/FortranCode/src/readGeneral.f90 File Reference

Reads General.inp file.

Functions/Subroutines

- subroutine readgeneral

  Reads Genearl.inp file.

4.29.1 Detailed Description

Author

Andrea Facci.

Definition in file readGeneral.f90.

4.29.2 Function/Subroutine Documentation

4.29.2.1 subroutine readgeneral ( )

This subroutine reads the file "General.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 60 of file readGeneral.f90.

References abortexecution(), allocatevar(), and readkeyword().

Referenced by main().

4.30 /home/Codici/Blink/FortranCode/src/readKeyword.f90 File Reference

Reads a line of a text file.

Functions/Subroutines

- subroutine readkeyword (theUnit, rew, keyword, value, error, nRow)

  Reads a line of a text file.

4.30.1 Detailed Description

Author

Andrea Facci.

Definition in file readKeyword.f90.

4.30.2 Function/Subroutine Documentation

4.30.2.1 subroutine readkeyword (integer, intent(in) *theUnit,* logical, intent(in), optional *rew,* character(len=100), intent(out) *keyword,* character(len=100), intent(out) *value,* integer, intent(out), optional *error,* integer, intent(out), optional *nRow*)

This procedure reads a line of a unit and returns the "keyword" and "value" fileds as single value charaters of maximum lenght = 100. The Unit needs to be already opened and associated to the desired file. This subroutine reads the line corresponding to the actual position of te cursor. The structure expected for the input text file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, as well as lines beginning with "!" that are considered comments. If only one delimiter "|" is present the line is considered mispelled, no value are is associated to keyword nor to value, and an optional error code is returned. The number of lines that were read before the first valid line can be returned with the optional argument "nRow". If the optional argument "rew" is set to true the unit is rewinded exactly of nRow lines at the end of the procedure.

Parameters

out	*Keyword,value*	keyword and value fields
out	*error*	Error code = 1 in case of mispelled lines
out	*nRow*	The number of lines that were read before the first valid line
in	*theUnit*	Unit number associated to the desired file.
in	*rew*	Wether to rewind or not the unit. Default is false.

Author

Andrea Facci.

Definition at line 71 of file readKeyword.f90.

References rewunit().

Referenced by readboiler(), readchillers(), readgeneral(), and readtrigen().

4.31 /home/Codici/Blink/FortranCode/src/readLoad.f90 File Reference

Reads Load.inp file.

Functions/Subroutines

- subroutine readloads

  Reads Load.inp file.

4.31.1 Detailed Description

Author

Andrea Facci.

Definition in file readLoad.f90.

4.31.2 Function/Subroutine Documentation

4.31.2.1 subroutine readloads ( )

This subroutine reads the file "Load.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readLoad.f90.

References abortexecution(), allocatevar(), cmatrixread(), dmatrixread(), findentry(), hcount(), matrixread(), openunit(), rewunit(), and vcount().

Referenced by main().

4.32 /home/Codici/Blink/FortranCode/src/readTrigen.f90 File Reference

Reads Trigeneration.inp file.

Functions/Subroutines

- subroutine readtrigen

  Reads Trigeneration.inp file.

4.32.1 Detailed Description

Author

Andrea Facci.

Definition in file readTrigen.f90.

4.32.2 Function/Subroutine Documentation

4.32.2.1 subroutine readtrigen ( )

This subroutine reads the file "Trigeneration.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of usless stuff because only text between begin and end is read.
   begin
      !Commented line
      KeywordField  | ValueField | !Comment two

      KeywordField  | ValueField |
      ScalarValue   | 1          |
      Vector        | 1 2 n      |
      VectorSeries  |(a b c ) (d e f)|
      Matrix        |(11 12 13  |
                    | 21 22 23  |
                    | 31 32 33) |
   end
A lof really usless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 62 of file readTrigen.f90.

References abortexecution(), allocatevar(), dmatrixread(), hcount(), ifindentry(), readkeyword(), rewunit(), and vcount().

Referenced by main().

4.33 /home/Codici/Blink/FortranCode/src/rewUnit.f90 File Reference

rewind a unit.

Functions/Subroutines

- subroutine rewunit (theUnit, n)

  rewind a unit.

4.33.1 Detailed Description

Author

Definition in file rewUnit.f90.

4.33.2 Function/Subroutine Documentation

4.33.2.1 subroutine rewunit (integer, intent(in) *theUnit,* integer, intent(in) *n*)

Rewinds a unit for a specified number of lines. Note that the unit needs to be opened.

Parameters

in	*theUnit*	The Unit to be rewinded.
in	*n*	The number of lines to rewind.

Author

Definition at line 38 of file rewUnit.f90.

Referenced by findentry(), matrixread(), readboiler(), readchillers(), readkeyword(), readloads(), readtrigen(), and vcount().

4.34 /home/Codici/Blink/FortranCode/src/vCount.f90 File Reference

Counts the lines of a text matrix.

Functions/Subroutines

- integer function vcount (theUnit, rew_, first_, last_)

Counts the lines of a text matrix.

4.34.1 Detailed Description

Author

Definition in file vCount.f90.

4.34.2 Function/Subroutine Documentation

4.34.2.1 integer function vcount (integer, intent(in) *theUnit,* logical, optional *rew_,* character(len=1), intent(in), optional *first_,* character(len=1), intent(in), optional *last_*)

This subroutine determines the number of lines between the two array delimiters. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
 A Lot of usless stuff because only text between begin and end is read.
    begin
       !Commented line
       KeywordField  | ValueField | !Comment two
    end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the defult values that are "(" ande ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \left[\begin{array}{cc} 1.1 & 1.2 \\ 2.1 & 2.2 \end{array} \right]$$

is:

```
    begin
       !Comment one
       exArr  | (1.1 1.2 | !Comment two
              |  2.1 2.2)|
    end
```

or equivalently:

```
    begin
       !Comment one
       exArr  |        | !Comment two
              | (1.1 1.2 |
              |  2.1 2.2)|
    end
```

so that each line in the text represnts a row of the vector output and coluns are space separated.

Parameters

in	*theUnit*	The unit associated to the file to be read
in	*rew_*	Weather to rewind or not the unit at the end. Default is false.
in	*first_,last_*	Opening and closing characters of the array.

Author

Definition at line 80 of file vCount.f90.

References rewunit().

Referenced by readboiler(), readchillers(), readloads(), and readtrigen().

4.35 /home/Codici/Blink/FortranCode/src/warning.f90 File Reference

prints warnings to standard output

Functions/Subroutines

- subroutine warning (i, j, k, line, word, r1, r2)

  print warnings to standard output

4.35.1 Detailed Description

Author

Andrea Facci.

Definition in file warning.f90.

4.35.2 Function/Subroutine Documentation

4.35.2.1 subroutine warning (integer, intent(in), optional *i,* integer, intent(in), optional *j,* integer, intent(in), optional *k,* integer, intent(in), optional *line,* character(len=∗), intent(in), optional *word,* real(kind(1.d0)), intent(in), optional *r1,* real(kind(1.d0)), intent(in), optional *r2*)

print warnings to standard output

Author

Andrea Facci

Parameters

in	*i,j,k*	error codes
in	*line*	line to locate the error position
in	*word*	mispelled or unrecognized word
in	*r1,r*	real numbers for unexpected values.

Definition at line 38 of file warning.f90.

Index