

PowerManager

1.0

Generated by Doxygen 1.8.1.2

Mon Jan 14 2013 18:25:43

Contents

1	Data Type Index	1
1.1	Data Types List	1
2	File Index	3
2.1	File List	3
3	Data Type Documentation	5
3.1	interfaces::abortExecution Interface Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function/Subroutine Documentation	5
3.1.2.1	abortexecution	5
3.2	filetools::cFindEntry Interface Reference	6
3.2.1	Detailed Description	6
3.2.2	Member Function/Subroutine Documentation	6
3.2.2.1	cfindentry	6
3.3	filetools::cMatrixRead Interface Reference	6
3.3.1	Detailed Description	7
3.3.2	Member Function/Subroutine Documentation	7
3.3.2.1	cmatrixread	7
3.4	cmdvar Module Reference	7
3.4.1	Detailed Description	7
3.4.2	Member Data Documentation	8
3.4.2.1	silent	8
3.4.2.2	verb	8
3.4.2.3	vsilent	8
3.5	graphtools::constraints Interface Reference	8
3.5.1	Detailed Description	8
3.5.2	Constructor & Destructor Documentation	9
3.5.2.1	constraints	9
3.6	filetools::dFindEntry Interface Reference	9
3.6.1	Detailed Description	9
3.6.2	Member Function/Subroutine Documentation	9

3.6.2.1	dfindentry	9
3.7	filetools::dMatrixRead Interface Reference	10
3.7.1	Detailed Description	10
3.7.2	Member Function/Subroutine Documentation	10
3.7.2.1	dmatrixread	10
3.8	economy Module Reference	11
3.8.1	Detailed Description	11
3.8.2	Member Function/Subroutine Documentation	12
3.8.2.1	chrev	12
3.8.2.2	currcost	12
3.8.2.3	elrev	13
3.8.2.4	firecost	13
3.8.2.5	fuelcost	14
3.8.2.6	maintenancecost	14
3.8.2.7	threv	15
3.9	energy Module Reference	15
3.9.1	Detailed Description	16
3.9.2	Member Function/Subroutine Documentation	16
3.9.2.1	chprod	16
3.9.2.2	elprod	16
3.9.2.3	elselfcons	17
3.9.2.4	fuelcons	17
3.9.2.5	thprod	18
3.9.2.6	thselfcons	18
3.10	filetools Module Reference	18
3.10.1	Detailed Description	19
3.11	filetools::findEntry Interface Reference	20
3.11.1	Detailed Description	20
3.11.2	Member Function/Subroutine Documentation	20
3.11.2.1	findentry	20
3.12	graphtools Module Reference	21
3.12.1	Detailed Description	21
3.12.2	Member Function/Subroutine Documentation	21
3.12.2.1	allcombin	21
3.12.2.2	grapharcs	22
3.12.2.3	graphpoints	22
3.12.2.4	minpathtopo	22
3.13	filetools::hCount Interface Reference	23
3.13.1	Detailed Description	23
3.13.2	Member Function/Subroutine Documentation	23

3.13.2.1	hcount	23
3.14	filetools::iFindEntry Interface Reference	23
3.14.1	Detailed Description	24
3.14.2	Member Function/Subroutine Documentation	24
3.14.2.1	ifindentry	24
3.15	filetools::iMatrixRead Interface Reference	24
3.15.1	Detailed Description	24
3.15.2	Member Function/Subroutine Documentation	24
3.15.2.1	imatrixread	24
3.16	inputvar Module Reference	25
3.16.1	Detailed Description	27
3.16.2	Member Data Documentation	28
3.16.2.1	cch	28
3.16.2.2	cel	28
3.16.2.3	cth	28
3.16.2.4	degrateb	28
3.16.2.5	degratec	28
3.16.2.6	degratet	28
3.16.2.7	dt1	28
3.16.2.8	etab	28
3.16.2.9	etac	28
3.16.2.10	etacht	28
3.16.2.11	etaelt	28
3.16.2.12	etatht	28
3.16.2.13	firecostb	29
3.16.2.14	firecostc	29
3.16.2.15	firecostt	29
3.16.2.16	fuelcostb	29
3.16.2.17	fuelcostt	29
3.16.2.18	fuellhvb	29
3.16.2.19	fuellhvt	29
3.16.2.20	gridbuycost	29
3.16.2.21	gridconnection	29
3.16.2.22	gridsellcost	29
3.16.2.23	ich	29
3.16.2.24	ichp	29
3.16.2.25	ideg	30
3.16.2.26	iel	30
3.16.2.27	ielp	30
3.16.2.28	invb	30

3.16.2.29 invc	30
3.16.2.30 invt	30
3.16.2.31 ith	30
3.16.2.32 ithp	30
3.16.2.33 itime	30
3.16.2.34 ksizeb	30
3.16.2.35 ksizec	30
3.16.2.36 ksizeet	30
3.16.2.37 lifeb	31
3.16.2.38 lifec	31
3.16.2.39 lifet	31
3.16.2.40 maintcostb	31
3.16.2.41 maintcostc	31
3.16.2.42 maintcostt	31
3.16.2.43 nboi	31
3.16.2.44 nchi	31
3.16.2.45 netab	31
3.16.2.46 netac	31
3.16.2.47 netacht	31
3.16.2.48 netaelt	31
3.16.2.49 netatht	32
3.16.2.50 nld	32
3.16.2.51 nload	32
3.16.2.52 nlp	32
3.16.2.53 nsizeb	32
3.16.2.54 nsizec	32
3.16.2.55 nsizeet	32
3.16.2.56 nspb	32
3.16.2.57 nspc	32
3.16.2.58 nspt	32
3.16.2.59 ntime	32
3.16.2.60 ntimes	32
3.16.2.61 ntrig	33
3.16.2.62 obj	33
3.16.2.63 pmaxb	33
3.16.2.64 pmaxc	33
3.16.2.65 pmaxt	33
3.16.2.66 spb	33
3.16.2.67 spc	33
3.16.2.68 spt	33

3.16.2.69 startpoint	33
3.16.2.70 tecb	33
3.16.2.71 tecc	33
3.16.2.72 tect	33
3.16.2.73 time	34
3.16.2.74 uch	34
3.16.2.75 uel	34
3.16.2.76 uth	34
3.17 interfaces Module Reference	34
3.17.1 Detailed Description	34
3.18 mathtools::interpolation Interface Reference	35
3.18.1 Detailed Description	35
3.18.2 Constructor & Destructor Documentation	35
3.18.2.1 interpolation	35
3.19 mathtools Module Reference	35
3.19.1 Detailed Description	36
3.20 filetools::matrixRead Interface Reference	36
3.20.1 Detailed Description	37
3.20.2 Member Function/Subroutine Documentation	37
3.20.2.1 matrixread	37
3.21 myarithmetic Module Reference	37
3.21.1 Detailed Description	37
3.21.2 Member Function/Subroutine Documentation	38
3.21.2.1 inan	38
3.21.2.2 isnan	38
3.21.2.3 rnan	38
3.22 graphtools::objFunction Interface Reference	39
3.22.1 Detailed Description	39
3.22.2 Member Function/Subroutine Documentation	39
3.22.2.1 objfunction	39
3.23 plantvar Module Reference	39
3.23.1 Detailed Description	41
3.23.2 Member Data Documentation	41
3.23.2.1 cf	41
3.23.2.2 cr	41
3.23.2.3 cref	41
3.23.2.4 dt	41
3.23.2.5 etab_	41
3.23.2.6 etac_	41
3.23.2.7 etach	42

3.23.2.8	etacht_	42
3.23.2.9	etael	42
3.23.2.10	etaelt_	42
3.23.2.11	etath	42
3.23.2.12	etatht_	42
3.23.2.13	ib	42
3.23.2.14	ic	42
3.23.2.15	ie	42
3.23.2.16	is	42
3.23.2.17	it	42
3.23.2.18	lhv	42
3.23.2.19	nm	43
3.23.2.20	nsp	43
3.23.2.21	nsptot	43
3.23.2.22	oemcost	43
3.23.2.23	onoffcost	43
3.23.2.24	pes	43
3.23.2.25	pmax	43
3.23.2.26	sp	43
3.24	filetools::readKeyword Interface Reference	44
3.24.1	Detailed Description	44
3.24.2	Member Function/Subroutine Documentation	44
3.24.2.1	readkeyword	44
3.25	filetools::rewUnit Interface Reference	44
3.25.1	Detailed Description	45
3.25.2	Member Function/Subroutine Documentation	45
3.25.2.1	rewunit	45
3.26	filetools::vCount Interface Reference	45
3.26.1	Detailed Description	45
3.26.2	Member Function/Subroutine Documentation	45
3.26.2.1	vcoun	45
3.27	interfaces::warning Interface Reference	46
3.27.1	Detailed Description	46
3.27.2	Constructor & Destructor Documentation	46
3.27.2.1	warning	46
4	File Documentation	47
4.1	/home/Codici/Blink/FortranCode/src/abortExecution.f90 File Reference	47
4.1.1	Detailed Description	47
4.1.2	Function/Subroutine Documentation	47

4.1.2.1	abortexecution	47
4.2	/home/Codici/Blink/FortranCode/src/aiuto.f90 File Reference	48
4.2.1	Detailed Description	48
4.2.2	Function/Subroutine Documentation	48
4.2.2.1	aiuto	48
4.3	/home/Codici/Blink/FortranCode/src/allocateVar.f90 File Reference	48
4.3.1	Detailed Description	48
4.3.2	Function/Subroutine Documentation	49
4.3.2.1	allocatevar	49
4.4	/home/Codici/Blink/FortranCode/src/buildPlant.f90 File Reference	49
4.4.1	Detailed Description	49
4.4.2	Function/Subroutine Documentation	49
4.4.2.1	buildplant	49
4.5	/home/Codici/Blink/FortranCode/src/checkPlant.f90 File Reference	50
4.5.1	Detailed Description	50
4.5.2	Function/Subroutine Documentation	50
4.5.2.1	checkplant	50
4.6	/home/Codici/Blink/FortranCode/src/cmdVar.f90 File Reference	50
4.6.1	Detailed Description	50
4.7	/home/Codici/Blink/FortranCode/src/commandline.f90 File Reference	51
4.7.1	Detailed Description	51
4.7.2	Function/Subroutine Documentation	51
4.7.2.1	commandline	51
4.8	/home/Codici/Blink/FortranCode/src/constraints.f90 File Reference	51
4.8.1	Detailed Description	51
4.8.2	Function/Subroutine Documentation	51
4.8.2.1	constraints	51
4.9	/home/Codici/Blink/FortranCode/src/economy.f90 File Reference	52
4.9.1	Detailed Description	52
4.10	/home/Codici/Blink/FortranCode/src/endExecution.f90 File Reference	52
4.10.1	Detailed Description	52
4.10.2	Function/Subroutine Documentation	53
4.10.2.1	endexecution	53
4.11	/home/Codici/Blink/FortranCode/src/energy.f90 File Reference	53
4.11.1	Detailed Description	53
4.12	/home/Codici/Blink/FortranCode/src/fileTools.f90 File Reference	53
4.12.1	Detailed Description	54
4.13	/home/Codici/Blink/FortranCode/src/findEntry2.f90 File Reference	54
4.13.1	Detailed Description	54
4.13.2	Function/Subroutine Documentation	54

4.13.2.1	cfindentry	54
4.13.2.2	dfindentry	55
4.13.2.3	findentry	55
4.13.2.4	ifindentry	56
4.14	/home/Codici/Blink/FortranCode/src/graphTools.f90 File Reference	56
4.14.1	Detailed Description	56
4.15	/home/Codici/Blink/FortranCode/src/hCount.f90 File Reference	57
4.15.1	Detailed Description	57
4.15.2	Function/Subroutine Documentation	57
4.15.2.1	hcount	57
4.16	/home/Codici/Blink/FortranCode/src/inputVar.f90 File Reference	57
4.16.1	Detailed Description	58
4.17	/home/Codici/Blink/FortranCode/src/interfaces.f90 File Reference	58
4.17.1	Detailed Description	58
4.18	/home/Codici/Blink/FortranCode/src/interpolation.f90 File Reference	58
4.18.1	Detailed Description	58
4.18.2	Function/Subroutine Documentation	59
4.18.2.1	interpolation	59
4.19	/home/Codici/Blink/FortranCode/src/main.f90 File Reference	59
4.19.1	Detailed Description	59
4.19.2	Function/Subroutine Documentation	59
4.19.2.1	main	59
4.20	/home/Codici/Blink/FortranCode/src/mathTools.f90 File Reference	60
4.20.1	Detailed Description	60
4.21	/home/Codici/Blink/FortranCode/src/matrixRead2.f90 File Reference	60
4.21.1	Detailed Description	60
4.21.2	Function/Subroutine Documentation	61
4.21.2.1	cmatrixread	61
4.21.2.2	dmatrixread	62
4.21.2.3	imatrixread	63
4.21.2.4	matrixread	64
4.22	/home/Codici/Blink/FortranCode/src/myArithmetic.f90 File Reference	64
4.22.1	Detailed Description	64
4.23	/home/Codici/Blink/FortranCode/src/objFunction.f90 File Reference	65
4.23.1	Detailed Description	65
4.23.2	Function/Subroutine Documentation	65
4.23.2.1	objfunction	65
4.24	/home/Codici/Blink/FortranCode/src/openUnit.f90 File Reference	65
4.24.1	Detailed Description	66
4.24.2	Function/Subroutine Documentation	66

4.24.2.1	openunit	66
4.25	/home/Codici/Blink/FortranCode/src/plantVar.f90 File Reference	66
4.25.1	Detailed Description	66
4.26	/home/Codici/Blink/FortranCode/src/prototipo.f90 File Reference	66
4.26.1	Detailed Description	67
4.26.2	Function/Subroutine Documentation	67
4.26.2.1	implicit	67
4.27	/home/Codici/Blink/FortranCode/src/readBoiler.f90 File Reference	67
4.27.1	Detailed Description	67
4.27.2	Function/Subroutine Documentation	67
4.27.2.1	readboiler	67
4.28	/home/Codici/Blink/FortranCode/src/readChiller.f90 File Reference	68
4.28.1	Detailed Description	68
4.28.2	Function/Subroutine Documentation	68
4.28.2.1	readchillers	68
4.29	/home/Codici/Blink/FortranCode/src/readGeneral.f90 File Reference	69
4.29.1	Detailed Description	69
4.29.2	Function/Subroutine Documentation	69
4.29.2.1	readgeneral	69
4.30	/home/Codici/Blink/FortranCode/src/readKeyword.f90 File Reference	70
4.30.1	Detailed Description	70
4.30.2	Function/Subroutine Documentation	70
4.30.2.1	readkeyword	70
4.31	/home/Codici/Blink/FortranCode/src/readLoad.f90 File Reference	71
4.31.1	Detailed Description	71
4.31.2	Function/Subroutine Documentation	71
4.31.2.1	readloads	71
4.32	/home/Codici/Blink/FortranCode/src/readTrigen.f90 File Reference	72
4.32.1	Detailed Description	72
4.32.2	Function/Subroutine Documentation	72
4.32.2.1	readtrigen	72
4.33	/home/Codici/Blink/FortranCode/src/rewUnit.f90 File Reference	73
4.33.1	Detailed Description	73
4.33.2	Function/Subroutine Documentation	73
4.33.2.1	rewunit	73
4.34	/home/Codici/Blink/FortranCode/src/vCount.f90 File Reference	73
4.34.1	Detailed Description	74
4.34.2	Function/Subroutine Documentation	74
4.34.2.1	vcount	74
4.35	/home/Codici/Blink/FortranCode/src/warning.f90 File Reference	75

4.35.1 Detailed Description	75
4.35.2 Function/Subroutine Documentation	75
4.35.2.1 warning	75

Chapter 1

Data Type Index

1.1 Data Types List

Here are the data types with brief descriptions:

interfaces::abortExecution	5
filetools::cFindEntry	6
filetools::cMatrixRead	6
cmdvar	
Collects the variable read from command line	7
graphtools::constraints	8
filetools::dFindEntry	9
filetools::dMatrixRead	10
economy	11
energy	
Module for energy calculations	15
filetools	
Interfaces of procedures to read from file	18
filetools::findEntry	20
graphtools	21
filetools::hCount	23
filetools::iFindEntry	23
filetools::iMatrixRead	24
inputvar	
Input variables collection	25
interfaces	34
mathtools::interpolation	35
mathtools	
Collection of interfaces for basic mathematical tools	35
filetools::matrixRead	36
myarithmetic	
Creates and detects NaN	37
graphtools::objFunction	39
plantvar	
Collection of variables relative to the power plant structure	39
filetools::readKeyword	44
filetools::rewUnit	44
filetools::vCount	45
interfaces::warning	46

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/Codici/Blink/FortranCode/src/ abortExecution.f90	
Terminates the execution in case of error	47
/home/Codici/Blink/FortranCode/src/ aiuto.f90	
Prints a very short help	48
/home/Codici/Blink/FortranCode/src/ allocateVar.f90	
Variable allocation	48
/home/Codici/Blink/FortranCode/src/ buildPlant.f90	
Collects all the informations relative to the power plant	49
/home/Codici/Blink/FortranCode/src/ checkPlant.f90	
Check the power plant coherence with the energy demand	50
/home/Codici/Blink/FortranCode/src/ cmdVar.f90	
Collects the variable read from command line	50
/home/Codici/Blink/FortranCode/src/ commandline.f90	
Reads the commad line	51
/home/Codici/Blink/FortranCode/src/ constraints.f90	
Static constraints	51
/home/Codici/Blink/FortranCode/src/ economy.f90	
Costs and revenues calulation prodedures	52
/home/Codici/Blink/FortranCode/src/ endExecution.f90	
Normal termination of the execution	52
/home/Codici/Blink/FortranCode/src/ energy.f90	
Collection of function for energy flow calculation	53
/home/Codici/Blink/FortranCode/src/ fileTools.f90	
Collection of proceture interfaces useful to read from files	53
/home/Codici/Blink/FortranCode/src/ findEntry2.f90	
Collection of procetures to find a specific entry inside a file	54
/home/Codici/Blink/FortranCode/src/ graphTools.f90	
Graph construction and minumum path	56
/home/Codici/Blink/FortranCode/src/ hCount.f90	
Count the number of elements of a vector in a text file	57
/home/Codici/Blink/FortranCode/src/ inputVar.f90	
Input variables collection	57
/home/Codici/Blink/FortranCode/src/ interfaces.f90	
Collection of general pourpose interfaces	58
/home/Codici/Blink/FortranCode/src/ interpolation.f90	
Remaps a discrete scalar field on a given 1d grid	58
/home/Codici/Blink/FortranCode/src/ main.f90	
This is the main driver	59

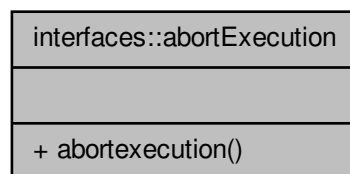
/home/Codici/Blink/FortranCode/src/mathTools.f90	
Collection of interfaces for basic mathematical tools	60
/home/Codici/Blink/FortranCode/src/matrixRead2.f90	
Reads 2D array of values from a file	60
/home/Codici/Blink/FortranCode/src/myArithmetic.f90	
Creates and detects NaN	64
/home/Codici/Blink/FortranCode/src/objFunction.f90	
Objective function	65
/home/Codici/Blink/FortranCode/src/openUnit.f90	
Checks file presence and opens the unit	65
/home/Codici/Blink/FortranCode/src/plantVar.f90	
Collection of variables relative to the power plant structure	66
/home/Codici/Blink/FortranCode/src/prototipo.f90	
File prototype	66
/home/Codici/Blink/FortranCode/src/readBoiler.f90	
Reads Boiler.inp file	67
/home/Codici/Blink/FortranCode/src/readChiller.f90	
Reads Chiller.inp file	68
/home/Codici/Blink/FortranCode/src/readGeneral.f90	
Reads General.inp file	69
/home/Codici/Blink/FortranCode/src/readKeyword.f90	
Reads a line of a text file	70
/home/Codici/Blink/FortranCode/src/readLoad.f90	
Reads Load.inp file	71
/home/Codici/Blink/FortranCode/src/readTrigen.f90	
Reads Trigeneration.inp file	72
/home/Codici/Blink/FortranCode/src/rewUnit.f90	
Rewind a unit	73
/home/Codici/Blink/FortranCode/src/vCount.f90	
Counts the lines of a text matrix	73
/home/Codici/Blink/FortranCode/src/warning.f90	
Prints warnings to standard output	75

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*)

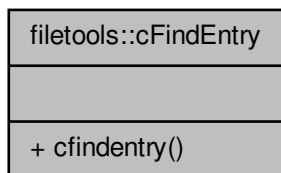
Definition at line 38 of file `interfaces.f90`.

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*)

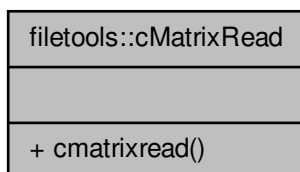
Definition at line 128 of file fileTools.f90.

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`.

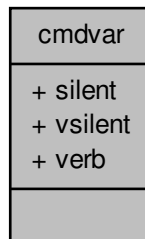
The documentation for this interface was generated from the following file:

- `/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

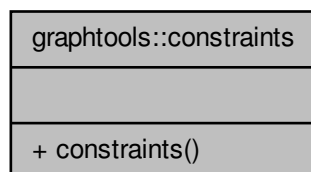
Definition at line 42 of file cmdVar.f90.

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*)

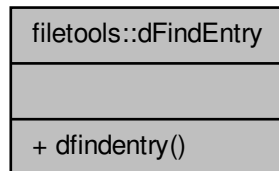
Definition at line 60 of file graphTools.f90.

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_*)

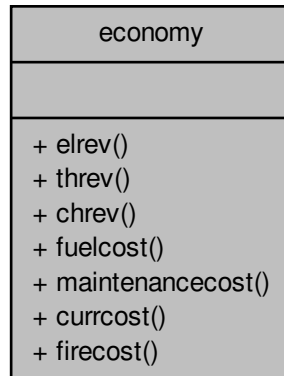
Definition at line 66 of file fileTools.f90.

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.

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

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)

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

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

<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	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()`.

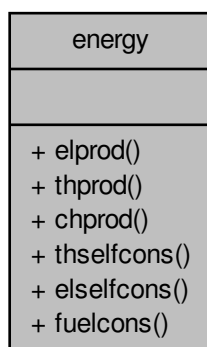
The documentation for this module was generated from the following file:

- `/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 to calculate the energy fluxes inside the power plant and between 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

<code>in</code>	<code>c_</code>	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::elfcons (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 number 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 primary 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

<code>in</code>	<code>c_</code>	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, the 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

<code>in</code>	<code>c_</code>	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*)

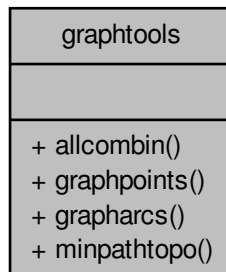
Definition at line 141 of file fileTools.f90.

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, starting 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 graph 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 associated 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(), myarithmetric::inan(), and myarithmetric::nran().

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 veteces 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 opimizazion the algorithm is tailored to sort acyclic graphs with topolgical ordering.

Author

Andrea Facci.

Definition at line 283 of file graphTools.f90.

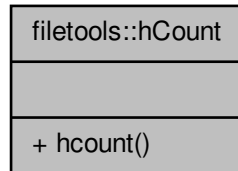
Referenced by main().

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_`)

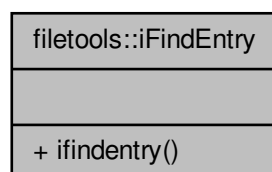
Definition at line 58 of file `fileTools.f90`.

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*)

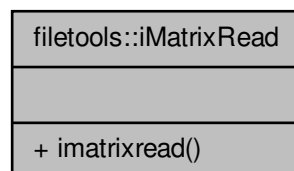
Definition at line 102 of file fileTools.f90.

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` [inv](#)
- `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` [etath](#)
- `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::nsizb

Definition at line 61 of file inputVar.f90.

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

Definition at line 70 of file inputVar.f90.

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

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`

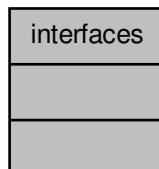
Definition at line 78 of file `inputVar.f90`.

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

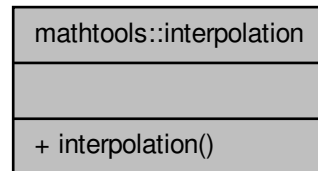
Definition at line 35 of file `interfaces.f90`.

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`.

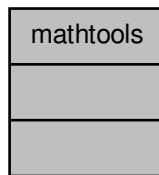
The documentation for this interface was generated from the following file:

- `/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.

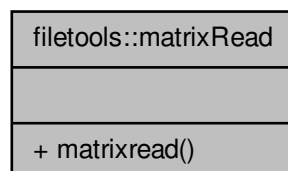
Definition at line 36 of file mathTools.f90.

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.

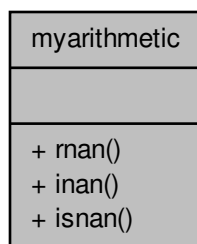
The documentation for this interface was generated from the following file:

- [/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 tries 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.

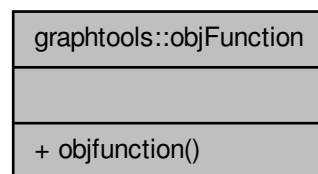
Referenced by graphtools::grapharcs().

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.

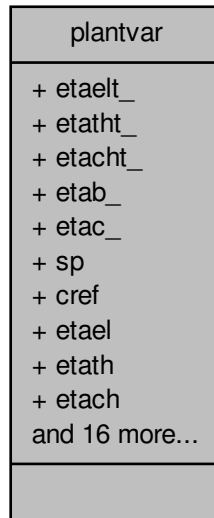
The documentation for this interface was generated from the following file:

- [/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

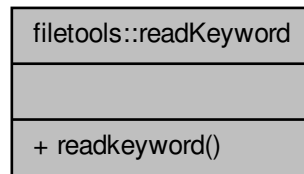
Definition at line 37 of file plantVar.f90.

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*)

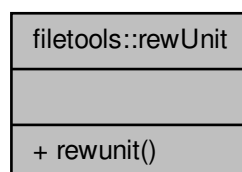
Definition at line 153 of file fileTools.f90.

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*)

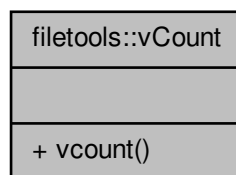
Definition at line 42 of file fileTools.f90.

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_*)

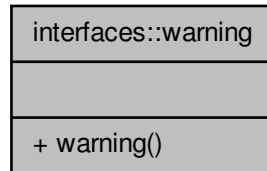
Definition at line 49 of file fileTools.f90.

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 execution 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 execution in case of error and print the error message according to the error code given as input.

Parameters

<i>in</i>	<i>i,j</i>	integers that identify the error.
<i>in</i>	<i>line</i>	integer that identifies the line affected by the mistake in an input file
<i>in</i>	<i>word</i>	character input to report the misspelled or unexpected sentence
<i>in</i>	<i>r1,r2</i>	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 variables according to the inputs. All the allocation of "global" variables should be done here.

Parameters

<i>in</i>	<i>what</i>	defines the variables to be allocated
<i>in</i>	<i>num</i>	defines the dimension 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 thermal energy needed by absorption chillers. If the plant is not grid connected also rated electrical power must be greater 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 command 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

<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	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	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	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	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	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	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	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	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	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 enclosed 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

<i>in</i>	<i>value_</i>	the string containing the vector whose length is to be determined
<i>in</i>	<i>first_</i> , <i>last_</i>	single character delimiters of the vector (open and close respectively). If not provided "(" and ")" 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 original grid, values are extrapolated and an optional warning code is returned.

Parameters

in	<i>xIn,yIn</i>	grid and values of the 1d field to be mapped
in	<i>n</i>	number of elements of the discrete field (size(<i>xIn</i>))
in	<i>xOut</i>	1d grid where the field is sampled
in	<i>m</i>	number of elements of the interpolation grid (<i>xOut</i>)
out	<i>warn</i>	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 useless 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 automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", 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 represents a row of the vector output and columns are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	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 useless 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 automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

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

$$exArr = \begin{bmatrix} 1.1 & 1.2 \\ 2.1 & 2.2 \end{bmatrix}$$

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 represents a row of the vector output and coluns are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	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 useless 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 automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", 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 represents a row of the vector output and coluns are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	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 useless 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 automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	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

<code>in</code>	<code>obj</code>	the optimization criterion.
<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	time step index. Note <code>t=x</code> meas the <code>x</code> '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=*) *fl*, integer, intent(in) *unt*, logical, intent(out) *prsnt*)

checks file presence and opens the unit.

Parameters

in	<i>fl</i>	The file name
in	<i>unt</i>	The unit number
out	<i>prsnt</i>	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 useless 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 useless 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 useless 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 lot really useless 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 useless 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 useless 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" fields as single value characters of maximum length = 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 the 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 useless 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 lot of really useless 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 misspelled, no value 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	<i>Keyword,value</i>	keyword and value fields
out	<i>error</i>	Error code = 1 in case of misspelled lines
out	<i>nRow</i>	The number of lines that were read before the first valid line
in	<i>theUnit</i>	Unit number associated to the desired file.
in	<i>rew</i>	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 useless 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 useless 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 Trigenation.inp file.

Functions/Subroutines

- subroutine [readtrigen](#)
Reads Trigenation.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 "Trigenation.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 useless 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 lot really useless 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

<code>in</code>	<i>theUnit</i>	The Unit to be rewinded.
<code>in</code>	<i>n</i>	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 useless 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 automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

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

$$exArr = \begin{bmatrix} 1.1 & 1.2 \\ 2.1 & 2.2 \end{bmatrix}$$

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 represents a row of the vector output and columns are space separated.

Parameters

in	<i>theUnit</i>	The unit associated to the file to be read
in	<i>rew_</i>	Weather to rewind or not the unit at the end. Default is false.
in	<i>first_</i> , <i>last_</i>	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

<i>in</i>	<i>i,j,k</i>	error codes
<i>in</i>	<i>line</i>	line to locate the error position
<i>in</i>	<i>word</i>	misspelled or unrecognized word
<i>in</i>	<i>r1,r</i>	real numbers for unexpected values.

Definition at line 38 of file `warning.f90`.

Index

/home/Codici/Blink/FortranCode/src/abortExecution.f90, 47
/home/Codici/Blink/FortranCode/src/aiuto.f90, 48
/home/Codici/Blink/FortranCode/src/allocateVar.f90, 48
/home/Codici/Blink/FortranCode/src/buildPlant.f90, 49
/home/Codici/Blink/FortranCode/src/checkPlant.f90, 50
/home/Codici/Blink/FortranCode/src/cmdVar.f90, 50
/home/Codici/Blink/FortranCode/src/commandline.f90, 51
/home/Codici/Blink/FortranCode/src/constraints.f90, 51
/home/Codici/Blink/FortranCode/src/economy.f90, 52
/home/Codici/Blink/FortranCode/src/endExecution.f90, 52
/home/Codici/Blink/FortranCode/src/energy.f90, 53
/home/Codici/Blink/FortranCode/src/fileTools.f90, 53
/home/Codici/Blink/FortranCode/src/findEntry2.f90, 54
/home/Codici/Blink/FortranCode/src/graphTools.f90, 56
/home/Codici/Blink/FortranCode/src/hCount.f90, 57
/home/Codici/Blink/FortranCode/src/inputVar.f90, 57
/home/Codici/Blink/FortranCode/src/interfaces.f90, 58
/home/Codici/Blink/FortranCode/src/interpolation.f90, 58
/home/Codici/Blink/FortranCode/src/main.f90, 59
/home/Codici/Blink/FortranCode/src/mathTools.f90, 60
/home/Codici/Blink/FortranCode/src/matrixRead2.f90, 60
/home/Codici/Blink/FortranCode/src/myArithmetic.f90, 64
/home/Codici/Blink/FortranCode/src/objFunction.f90, 65
/home/Codici/Blink/FortranCode/src/openUnit.f90, 65
/home/Codici/Blink/FortranCode/src/plantVar.f90, 66
/home/Codici/Blink/FortranCode/src/prototipo.f90, 66
/home/Codici/Blink/FortranCode/src/readBoiler.f90, 67
/home/Codici/Blink/FortranCode/src/readChiller.f90, 68
/home/Codici/Blink/FortranCode/src/readGeneral.f90, 69
/home/Codici/Blink/FortranCode/src/readKeyword.f90, 70
/home/Codici/Blink/FortranCode/src/readLoad.f90, 71
/home/Codici/Blink/FortranCode/src/readTrigen.f90, 72
/home/Codici/Blink/FortranCode/src/rewUnit.f90, 73
/home/Codici/Blink/FortranCode/src/vCount.f90, 73
/home/Codici/Blink/FortranCode/src/warning.f90, 75

abortExecution.f90
 abortexecution, 47
abortexecution
 abortExecution.f90, 47
 interfaces::abortExecution, 5
aiuto
 aiuto.f90, 48
aiuto.f90
 aiuto, 48
allcombin
 graphtools, 21
allocateVar.f90
 allocatevar, 49
allocatevar
 allocateVar.f90, 49
buildPlant.f90
 buildplant, 49
buildplant
 buildPlant.f90, 49
cch
 inputvar, 28
cel
 inputvar, 28
cf
 plantvar, 41
cfindentry
 filetools::cFindEntry, 6
 findEntry2.f90, 54
checkPlant.f90
 checkplant, 50
checkplant
 checkPlant.f90, 50
chprod
 energy, 16
chrev
 economy, 12
cmatrixread
 filetools::cMatrixRead, 7
 matrixRead2.f90, 61
cmdvar, 7
 silent, 8
 verb, 8
 vsilent, 8
commandline
 commandline.f90, 51
commandline.f90
 commandline, 51
constraints
 constraints.f90, 51
 graphtools::constraints, 9
constraints.f90
 constraints, 51
cr
 plantvar, 41

cref
 plantvar, 41

cth
 inputvar, 28

currcost
 economy, 12

degrateb
 inputvar, 28

degratec
 inputvar, 28

degratet
 inputvar, 28

dfindentry
 filetools::dFindEntry, 9
 findEntry2.f90, 55

dmatrixread
 filetools::dMatrixRead, 10
 matrixRead2.f90, 62

dt
 plantvar, 41

dt1
 inputvar, 28

economy, 11
 chrev, 12
 currcost, 12
 elrev, 13
 firecost, 13
 fuelcost, 14
 maintenancecost, 14
 threv, 14

elprod
 energy, 16

elrev
 economy, 13

elseifcons
 energy, 17

endExecution.f90
 endexecution, 53

endexecution
 endExecution.f90, 53

energy, 15
 chprod, 16
 elprod, 16
 elseifcons, 17
 fuelcons, 17
 thprod, 18
 thselfcons, 18

etab
 inputvar, 28

etab_
 plantvar, 41

etac
 inputvar, 28

etac_
 plantvar, 41

etach
 plantvar, 41

etacht
 inputvar, 28

etacht_
 plantvar, 42

etael
 plantvar, 42

etaelt
 inputvar, 28

etaelt_
 plantvar, 42

etath
 plantvar, 42

etatht
 inputvar, 28

etatht_
 plantvar, 42

filetools, 18
filetools::cFindEntry, 6
 cfindentry, 6
filetools::cMatrixRead, 6
 cmatrixread, 7
filetools::dFindEntry, 9
 dfindentry, 9
filetools::dMatrixRead, 10
 dmatrixread, 10
filetools::findEntry, 20
 findentry, 20
filetools::hCount, 23
 hcount, 23
filetools::iFindEntry, 23
 ifindentry, 24
filetools::iMatrixRead, 24
 imatrixread, 24
filetools::matrixRead, 36
 matrixread, 37
filetools::readKeyword, 44
 readkeyword, 44
filetools::rewUnit, 44
 rewunit, 45
filetools::vCount, 45
 vcount, 45
findEntry2.f90
 cfindentry, 54
 dfindentry, 55
 findentry, 55
 ifindentry, 56
findentry
 filetools::findEntry, 20
 findEntry2.f90, 55
firecost
 economy, 13
firecostb
 inputvar, 28
firecostc
 inputvar, 29
firecostt
 inputvar, 29
fuelcons

- energy, [17](#)
- fuelcost
 - economy, [14](#)
- fuelcostb
 - inputvar, [29](#)
- fuelcostt
 - inputvar, [29](#)
- fuellhvb
 - inputvar, [29](#)
- fuellhvt
 - inputvar, [29](#)
- grapharcs
 - graphtools, [22](#)
- graphpoints
 - graphtools, [22](#)
- graphtools, [21](#)
 - allcombin, [21](#)
 - grapharcs, [22](#)
 - graphpoints, [22](#)
 - minpath topo, [22](#)
- graphtools::constraints, [8](#)
 - constraints, [9](#)
- graphtools::objFunction, [39](#)
 - objfunction, [39](#)
- gridbuycost
 - inputvar, [29](#)
- gridconnection
 - inputvar, [29](#)
- gridsellcost
 - inputvar, [29](#)
- hCount.f90
 - hcount, [57](#)
- hcount
 - filetools::hCount, [23](#)
 - hCount.f90, [57](#)
- ib
 - plantvar, [42](#)
- ic
 - plantvar, [42](#)
- ich
 - inputvar, [29](#)
- ichp
 - inputvar, [29](#)
- ideg
 - inputvar, [29](#)
- ie
 - plantvar, [42](#)
- iel
 - inputvar, [30](#)
- ielp
 - inputvar, [30](#)
- ifindentry
 - filetools::iFindEntry, [24](#)
 - findEntry2.f90, [56](#)
- imatrixread
 - filetools::iMatrixRead, [24](#)
- matrixRead2.f90, [63](#)
- implicit
 - prototipo.f90, [67](#)
- inan
 - myarithmetic, [38](#)
- inputvar, [25](#)
 - cch, [28](#)
 - cel, [28](#)
 - cth, [28](#)
 - degrateb, [28](#)
 - degratec, [28](#)
 - degratet, [28](#)
 - dt1, [28](#)
 - etab, [28](#)
 - etac, [28](#)
 - etacht, [28](#)
 - etaelt, [28](#)
 - etatht, [28](#)
 - firecostb, [28](#)
 - firecostc, [29](#)
 - firecostt, [29](#)
 - fuelcostb, [29](#)
 - fuelcostt, [29](#)
 - fuellhvb, [29](#)
 - fuellhvt, [29](#)
 - gridbuycost, [29](#)
 - gridconnection, [29](#)
 - gridsellcost, [29](#)
 - ich, [29](#)
 - ichp, [29](#)
 - ideg, [29](#)
 - iel, [30](#)
 - ielp, [30](#)
 - invb, [30](#)
 - invc, [30](#)
 - invt, [30](#)
 - ith, [30](#)
 - ithp, [30](#)
 - itime, [30](#)
 - ksizeb, [30](#)
 - ksizec, [30](#)
 - ksizet, [30](#)
 - lifeb, [30](#)
 - lifec, [31](#)
 - lifet, [31](#)
 - maintcostb, [31](#)
 - maintcostc, [31](#)
 - maintcostt, [31](#)
 - nboi, [31](#)
 - nchi, [31](#)
 - netab, [31](#)
 - netac, [31](#)
 - netacht, [31](#)
 - netaelt, [31](#)
 - netatht, [31](#)
 - nld, [32](#)
 - nload, [32](#)
 - nlp, [32](#)

- nsizemb, [32](#)
- nsizec, [32](#)
- nsizet, [32](#)
- nspb, [32](#)
- nspc, [32](#)
- nspt, [32](#)
- ntime, [32](#)
- ntimes, [32](#)
- ntrig, [32](#)
- obj, [33](#)
- pmaxb, [33](#)
- pmaxc, [33](#)
- pmaxt, [33](#)
- spb, [33](#)
- spc, [33](#)
- spt, [33](#)
- startpoint, [33](#)
- tecb, [33](#)
- tecc, [33](#)
- tect, [33](#)
- time, [33](#)
- uch, [34](#)
- uel, [34](#)
- uth, [34](#)
- interfaces, [34](#)
- interfaces::abortExecution, [5](#)
 - abortexecution, [5](#)
- interfaces::warning, [46](#)
 - warning, [46](#)
- interpolation
 - interpolation.f90, [59](#)
 - mathtools::interpolation, [35](#)
- interpolation.f90
 - interpolation, [59](#)
- invb
 - inputvar, [30](#)
- invc
 - inputvar, [30](#)
- invt
 - inputvar, [30](#)
- is
 - plantvar, [42](#)
- isnan
 - myarithmetic, [38](#)
- it
 - plantvar, [42](#)
- ith
 - inputvar, [30](#)
- ithp
 - inputvar, [30](#)
- itime
 - inputvar, [30](#)
- ksizeb
 - inputvar, [30](#)
- ksizec
 - inputvar, [30](#)
- ksizet
 - inputvar, [30](#)
- lhv
 - plantvar, [42](#)
- lifeb
 - inputvar, [30](#)
- lifec
 - inputvar, [31](#)
- lifet
 - inputvar, [31](#)
- main
 - main.f90, [59](#)
- main.f90
 - main, [59](#)
- maintcostb
 - inputvar, [31](#)
- maintcostc
 - inputvar, [31](#)
- maintcostt
 - inputvar, [31](#)
- maintenancecost
 - economy, [14](#)
- mathtools, [35](#)
- mathtools::interpolation, [35](#)
 - interpolation, [35](#)
- matrixRead2.f90
 - cmatrixread, [61](#)
 - dmatrixread, [62](#)
 - imatrixread, [63](#)
 - matrixread, [64](#)
- matrixread
 - filetools::matrixRead, [37](#)
 - matrixRead2.f90, [64](#)
- minpathtopo
 - graphtools, [22](#)
- myarithmetic, [37](#)
 - inan, [38](#)
 - isnan, [38](#)
 - rnan, [38](#)
- nboi
 - inputvar, [31](#)
- nchi
 - inputvar, [31](#)
- netab
 - inputvar, [31](#)
- netac
 - inputvar, [31](#)
- netacht
 - inputvar, [31](#)
- netaelt
 - inputvar, [31](#)
- netatht
 - inputvar, [31](#)
- nld
 - inputvar, [32](#)
- nload
 - inputvar, [32](#)
- nlp
 - inputvar, [32](#)

- nm
 - plantvar, 42
- nsizéb
 - inputvar, 32
- nsizéc
 - inputvar, 32
- nsizet
 - inputvar, 32
- nsp
 - plantvar, 43
- nspb
 - inputvar, 32
- nspc
 - inputvar, 32
- nspt
 - inputvar, 32
- nsptot
 - plantvar, 43
- ntime
 - inputvar, 32
- ntimes
 - inputvar, 32
- ntrig
 - inputvar, 32
- obj
 - inputvar, 33
- objFunction.f90
 - objfunction, 65
- objfunction
 - graphtools::objFunction, 39
 - objFunction.f90, 65
- oemcost
 - plantvar, 43
- onoffcost
 - plantvar, 43
- openUnit.f90
 - openunit, 66
- openunit
 - openUnit.f90, 66
- pes
 - plantvar, 43
- plantvar, 39
 - cf, 41
 - cr, 41
 - cref, 41
 - dt, 41
 - etab_, 41
 - etac_, 41
 - etach, 41
 - etacht_, 42
 - etael, 42
 - etaelt_, 42
 - etath, 42
 - etatht_, 42
 - ib, 42
 - ic, 42
 - ie, 42
 - is, 42
 - it, 42
 - lhv, 42
 - nm, 42
 - nsp, 43
 - nsptot, 43
 - oemcost, 43
 - onoffcost, 43
 - pes, 43
 - pmax, 43
 - sp, 43
- pmax
 - plantvar, 43
- pmaxb
 - inputvar, 33
- pmaxc
 - inputvar, 33
- pmaxt
 - inputvar, 33
- prototipo.f90
 - implicit, 67
- readBoiler.f90
 - readboiler, 67
- readChiller.f90
 - readchillers, 68
- readGeneral.f90
 - readgeneral, 69
- readKeyword.f90
 - readkeyword, 70
- readLoad.f90
 - readloads, 71
- readTrigen.f90
 - readtrigen, 72
- readboiler
 - readBoiler.f90, 67
- readchillers
 - readChiller.f90, 68
- readgeneral
 - readGeneral.f90, 69
- readkeyword
 - filetools::readKeyword, 44
 - readKeyword.f90, 70
- readloads
 - readLoad.f90, 71
- readtrigen
 - readTrigen.f90, 72
- rewUnit.f90
 - rewunit, 73
- rewunit
 - filetools::rewUnit, 45
 - rewUnit.f90, 73
- rnan
 - myarithmetic, 38
- silent
 - cmdvar, 8
- sp
 - plantvar, 43

- spb
 - inputvar, [33](#)
- spc
 - inputvar, [33](#)
- spt
 - inputvar, [33](#)
- startpoint
 - inputvar, [33](#)
- tecb
 - inputvar, [33](#)
- tecc
 - inputvar, [33](#)
- tect
 - inputvar, [33](#)
- thprod
 - energy, [18](#)
- threv
 - economy, [14](#)
- thselfcons
 - energy, [18](#)
- time
 - inputvar, [33](#)
- uch
 - inputvar, [34](#)
- uel
 - inputvar, [34](#)
- uth
 - inputvar, [34](#)
- vCount.f90
 - vcount, [74](#)
- vcount
 - filetools::vCount, [45](#)
 - vCount.f90, [74](#)
- verb
 - cmdvar, [8](#)
- vsilent
 - cmdvar, [8](#)
- warning
 - interfaces::warning, [46](#)
 - warning.f90, [75](#)
- warning.f90
 - warning, [75](#)