

FINEL

Generated by Doxygen 1.8.12



# Contents

<b>1</b>	<b>Modules Index</b>	<b>1</b>
1.1	Modules List . . . . .	1
<b>2</b>	<b>Data Type Index</b>	<b>3</b>
2.1	Data Types List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	meshstructure Module Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Function/Subroutine Documentation . . . . .	7
4.1.2.1	mallocelelem() . . . . .	7
4.1.2.2	mallocnodes() . . . . .	8
4.2	minputreader Module Reference . . . . .	8
4.2.1	Detailed Description . . . . .	9
4.2.2	Function/Subroutine Documentation . . . . .	10
4.2.2.1	analyzefile() . . . . .	10
4.2.2.2	analyzefileinput() . . . . .	11
4.2.2.3	createsimpleinputfile() . . . . .	11
4.2.2.4	findinclude() . . . . .	12
4.2.2.5	findkeyword() . . . . .	12
4.2.2.6	mergeincludecontents() . . . . .	13
4.2.2.7	preparefilelines() . . . . .	13

4.2.2.8	<a href="#">readboundaryconditions()</a>	14
4.2.2.9	<a href="#">readinputfiles()</a>	15
4.2.2.10	<a href="#">readintarraykeywordvalue()</a>	16
4.2.2.11	<a href="#">readintegerkeywordvalue()</a>	17
4.2.2.12	<a href="#">readrealkeywordvalue()</a>	18
4.2.2.13	<a href="#">readrealmatrixvalues()</a>	18
4.2.2.14	<a href="#">readstringkeywordvalue()</a>	19
4.2.3	<a href="#">Variable Documentation</a>	20
4.2.3.1	<a href="#">file_lines</a>	20
4.2.3.2	<a href="#">number_of_lines</a>	20
4.3	<a href="#">mio Module Reference</a>	20
4.3.1	<a href="#">Function/Subroutine Documentation</a>	21
4.3.1.1	<a href="#">closefiles()</a>	21
4.3.1.2	<a href="#">openfiles()</a>	21
4.3.1.3	<a href="#">print_sol1d()</a>	22
4.3.1.4	<a href="#">read_elems()</a>	22
4.3.1.5	<a href="#">read_nodes()</a>	23
4.3.2	<a href="#">Variable Documentation</a>	24
4.3.2.1	<a href="#">iin</a>	24
4.3.2.2	<a href="#">infile</a>	24
4.3.2.3	<a href="#">iout</a>	24
4.3.2.4	<a href="#">isol</a>	24
4.3.2.5	<a href="#">outfile</a>	24
4.3.2.6	<a href="#">solfile</a>	24
4.3.2.7	<a href="#">title</a>	24
4.4	<a href="#">mprocessor Module Reference</a>	24
4.4.1	<a href="#">Detailed Description</a>	25
4.4.2	<a href="#">Function/Subroutine Documentation</a>	25
4.4.2.1	<a href="#">applybc()</a>	25
4.4.2.2	<a href="#">assmb()</a>	26

4.4.2.3	<a href="#">drchlt()</a>	27
4.4.2.4	<a href="#">formkf()</a>	27
4.4.2.5	<a href="#">neumann1d()</a>	28
4.4.2.6	<a href="#">processor()</a>	29
4.4.2.7	<a href="#">solver()</a>	30
4.5	<a href="#">mscalar Module Reference</a>	30
4.5.1	<a href="#">Detailed Description</a>	31
4.5.2	<a href="#">Function/Subroutine Documentation</a>	31
4.5.2.1	<a href="#">localelem()</a>	31
4.6	<a href="#">msetup Module Reference</a>	32
4.6.1	<a href="#">Detailed Description</a>	32
4.6.2	<a href="#">Function/Subroutine Documentation</a>	32
4.6.2.1	<a href="#">preprocessor()</a>	32
4.6.2.2	<a href="#">setupphase()</a>	33
4.7	<a href="#">mshapefunctions Module Reference</a>	34
4.7.1	<a href="#">Detailed Description</a>	35
4.7.2	<a href="#">Function/Subroutine Documentation</a>	35
4.7.2.1	<a href="#">setint()</a>	35
4.7.2.2	<a href="#">shpf1d()</a>	35
4.7.3	<a href="#">Variable Documentation</a>	36
4.7.3.1	<a href="#">w</a>	36
4.7.3.2	<a href="#">xi</a>	36
4.8	<a href="#">msolver Module Reference</a>	36
4.8.1	<a href="#">Detailed Description</a>	36
4.8.2	<a href="#">Function/Subroutine Documentation</a>	36
4.8.2.1	<a href="#">rsub()</a>	36
4.8.2.2	<a href="#">tri()</a>	37
4.9	<a href="#">mutilities Module Reference</a>	37
4.9.1	<a href="#">Detailed Description</a>	38
4.9.2	<a href="#">Function/Subroutine Documentation</a>	38
4.9.2.1	<a href="#">f1()</a>	38
4.9.2.2	<a href="#">linspace()</a>	38
4.9.2.3	<a href="#">print_matrix()</a>	39
4.9.2.4	<a href="#">quad1()</a>	39
4.9.2.5	<a href="#">test_shpf1d()</a>	40
4.10	<a href="#">scalarstructure Module Reference</a>	40
4.10.1	<a href="#">Detailed Description</a>	41
4.10.2	<a href="#">Function/Subroutine Documentation</a>	41
4.10.2.1	<a href="#">mallocelemkf()</a>	41
4.10.2.2	<a href="#">mallocglobalkf()</a>	41

<b>5</b>	<b>Data Type Documentation</b>	<b>43</b>
5.1	meshstructure::mesh Type Reference	43
5.1.1	Detailed Description	44
5.1.2	Member Data Documentation	44
5.1.2.1	ei	44
5.1.2.2	ej	45
5.1.2.3	ek	45
5.1.2.4	flagnode	45
5.1.2.5	gnode	45
5.1.2.6	mat	45
5.1.2.7	nelem	45
5.1.2.8	nelems	45
5.1.2.9	nen	45
5.1.2.10	nintp	46
5.1.2.11	nnodes	46
5.1.2.12	nsd	46
5.1.2.13	numat	46
5.1.2.14	si	46
5.1.2.15	sj	46
5.1.2.16	sk	46
5.1.2.17	x	46
5.1.2.18	xv	47
5.1.2.19	y	47
5.1.2.20	yv	47
5.2	scalarstructure::scalarstructuresystem Type Reference	47
5.2.1	Detailed Description	48
5.2.2	Member Data Documentation	48
5.2.2.1	kbc	48
5.2.2.2	lhelem	48
5.2.2.3	lhsys	48
5.2.2.4	mat	48
5.2.2.5	rhelem	49
5.2.2.6	rhsys	49
5.2.2.7	u	49
5.2.2.8	vbc	49

<b>6 File Documentation</b>	<b>51</b>
6.1 src/driver.F90 File Reference	51
6.1.1 Function/Subroutine Documentation	51
6.1.1.1 finel()	51
6.2 src/io.F90 File Reference	52
6.3 src/meshStructure.F90 File Reference	53
6.4 src/mInputReader.F90 File Reference	53
6.5 src/processor.F90 File Reference	54
6.6 src/scalar.F90 File Reference	55
6.7 src/scalarStructure.F90 File Reference	55
6.8 src/setup.F90 File Reference	56
6.9 src/shapeFunctions.F90 File Reference	56
6.10 src/solver.F90 File Reference	56
6.11 src/utilities.F90 File Reference	57
<b>Index</b>	<b>59</b>





# Chapter 1

## Modules Index

### 1.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">meshstructure</a>	Module that contains the data structure of a mesh associate to a problem . . . . .	7
<a href="#">minputreader</a>	Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada . . . . .	8
<a href="#">mio</a>	. . . . .	20
<a href="#">mprocessor</a>	Processor module to compute, assemble and solve the system . . . . .	24
<a href="#">mscalar</a>	Contains variables and subroutine related to a general scalar problem . . . . .	30
<a href="#">msetup</a>	Module for setup phase by IO procedures . . . . .	32
<a href="#">mshapefunctions</a>	Module for shape functions computations and relate operations . . . . .	34
<a href="#">msolver</a>	Contains subroutine to compute numerical solution of linear systems $Ax=b$ . . . . .	36
<a href="#">mutilities</a>	Module for auxiliar routines . . . . .	37
<a href="#">scalarstructure</a>	Module that contains the data structure of a general scalar problem . . . . .	40



## Chapter 2

# Data Type Index

### 2.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">meshstructure::mesh</a>	
Data type for a mesh . . . . .	43
<a href="#">scalarstructure::scalarstructuresystem</a>	
Variables and characteristic data for a scalar problem . . . . .	47



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/driver.F90 . . . . .	51
src/io.F90 . . . . .	52
src/meshStructure.F90 . . . . .	53
src/mInputReader.F90 . . . . .	53
src/processor.F90 . . . . .	54
src/scalar.F90 . . . . .	55
src/scalarStructure.F90 . . . . .	55
src/setup.F90 . . . . .	56
src/shapeFunctions.F90 . . . . .	56
src/solver.F90 . . . . .	56
src/utilities.F90 . . . . .	57



## Chapter 4

# Module Documentation

### 4.1 meshstructure Module Reference

Module that contains the data structure of a mesh associate to a problem.

#### Data Types

- type `mesh`  
*Data type for a mesh.*

#### Functions/Subroutines

- subroutine `mallocnodes` (meshStrct, n)  
*Routine that allocate memory to node data.*
- subroutine `mallocelem` (meshStrct, n)  
*Routine that allocate memory to element data.*

#### 4.1.1 Detailed Description

Module that contains the data structure of a mesh associate to a problem.

#### Author

Diego T. Volpatto

#### 4.1.2 Function/Subroutine Documentation

##### 4.1.2.1 mallocelem()

```
subroutine meshstructure::mallocelem (  
    type(mesh) meshStrct,  
    integer n )
```

Routine that allocate memory to element data.

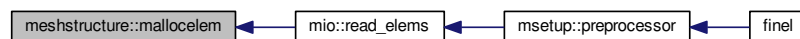
**Parameters**

<i>meshStrct</i>	[in/out] mesh structure to allocate
<i>n</i>	[in] number of elements

**Author**

Diego T. Volpatto

Here is the caller graph for this function:

**4.1.2.2 mallocnodes()**

```

subroutine meshstructure::mallocnodes (
    type(mesh) meshStrct,
    integer n )
  
```

Routine that allocate memory to node data.

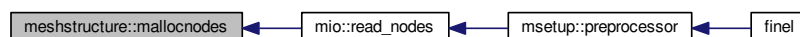
**Parameters**

<i>meshStrct</i>	[in/out] mesh structure to allocate
<i>n</i>	[in] number of nodes

**Author**

Diego T. Volpatto

Here is the caller graph for this function:

**4.2 minputreader Module Reference**

Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.



## Functions/Subroutines

- subroutine [readinputfiles](#) ()  
*Le arquivo de input e armazena seu conteudo em um array.*
- subroutine [createsimpleinputfile](#) ()  
*Cria a estrutura de input usando um arquivo de entrada sem includes.*
- subroutine [mergeincludecontents](#) (include\_file, include\_line)  
*Le o conteudo do arquivo de include e armazena no array principal.*
- subroutine [preparefilelines](#) (include\_indexes, include\_number\_of\_lines, number\_of\_includes, original\_file\_lines)  
*Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.*
- subroutine [analyzefileinput](#) (number\_of\_lines, number\_of\_includes)  
*Efetua algumas análises no arquivo recebido.*
- subroutine [analyzefile](#) (file\_name, number\_of\_lines, number\_of\_includes)  
*Efetua algumas análises no arquivo recebido.*
- integer \*4 function [findinclude](#) (position, file\_lines, number\_of\_lines)  
*Procura a n-esima palavra-chave include.*
- integer \*4 function [findkeyword](#) (keyword)  
*Procura uma palavra-chave.*
- subroutine [readintegerkeywordvalue](#) (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.*
- subroutine [readintarraykeywordvalue](#) (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se nao encontrado, associa o valor default fornecido.  
Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".*
- subroutine [readstringkeywordvalue](#) (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.*
- subroutine [readrealkeywordvalue](#) (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.*
- subroutine [readrealmatrixvalues](#) (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.*
- subroutine [readboundaryconditions](#) (keyword, kbc, vbc, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.*

## Variables

- character(len=200), dimension(:), allocatable [file\\_lines](#)  
*Armazena as linhas do arquivo de input.*
- integer \*4 [number\\_of\\_lines](#)  
*Armazena o numero de linhas no arquivo.*

### 4.2.1 Detailed Description

Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.

## 4.2.2 Function/Subroutine Documentation

### 4.2.2.1 analyzefile()

```
subroutine minputreader::analyzefile (  
    character(len=200) file_name,  
    integer*4 number_of_lines,  
    integer*4 number_of_includes )
```

Efetua algumas análises no arquivo recebido.

## Parameters

<i>file_name</i>	O nome do arquivo.
<i>number_of_lines</i>	Numero de linhas.
<i>number_of_include</i>	Numero de ocorrencias da palavra include.

Here is the caller graph for this function:



## 4.2.2.2 analyzefileinput()

```
subroutine minputreader::analyzefileinput (  
    integer*4 number_of_lines,  
    integer*4 number_of_includes )
```

Efetua algumas analises no arquivo recebido.

## Parameters

<i>number_of_lines</i>	Numero de linhas.
<i>number_of_include</i>	Numero de ocorrencias da palavra include.

Here is the caller graph for this function:



## 4.2.2.3 createsimpleinputfile()

```
subroutine minputreader::createsimpleinputfile ( )
```

Cria a estrutura de input usando um arquivo de entrada sem includes.

## Parameters

<i>file_name</i>	Nome do arquivo a ser lido.
------------------	-----------------------------

Here is the caller graph for this function:



#### 4.2.2.4 findinclude()

```
integer*4 function minputreader::findinclude (
    integer*4 position,
    character(len=200), dimension(:) file_lines,
    integer*4 number_of_lines )
```

Procura a n-esima palavra-chave include.

##### Parameters

<i>position</i>	Corresponde a posicao desejada.
<i>file_lines</i>	Linhas do arquivo.
<i>number_of_lines</i>	Numero de linhas atuais.

##### Returns

O indice da palavra-chave no array que contem as linhas do arquivo de entrada.

Here is the caller graph for this function:



#### 4.2.2.5 findkeyword()

```
integer*4 function minputreader::findkeyword (
    character(50) keyword )
```

Procura uma palavra-chave.

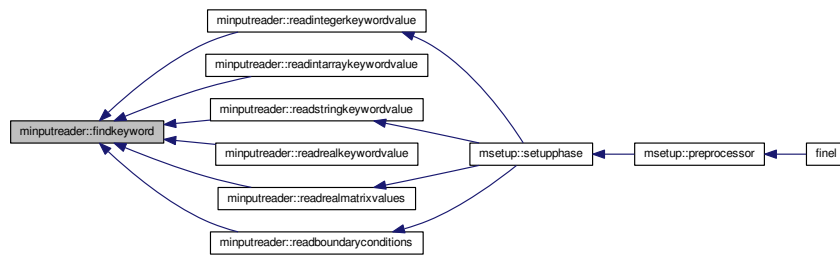
##### Parameters

<i>keyword</i>	A palavra-chave.
----------------	------------------

##### Returns

O indice da palavra-chave no array que contem as linhas do arquivo de entrada.

Here is the caller graph for this function:



#### 4.2.2.6 mergeincludecontents()

```

subroutine minputreader::mergeincludecontents (
    character(len=200) include_file,
    integer*4 include_line )

```

Le o conteúdo do arquivo de include e armazena no array principal.

##### Parameters

<i>include_index</i>	O index do include.
<i>include_files</i>	Array com includes.
<i>include_line</i>	A linha do include.

Here is the caller graph for this function:



#### 4.2.2.7 preparefilelines()

```

subroutine minputreader::preparefilelines (
    integer*4, dimension(:) include_indexes,
    integer*4, dimension(:) include_number_of_lines,
    integer*4 number_of_includes,
    character(len=200), dimension(:) original_file_lines )

```

Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.

##### Parameters

<i>include_indexes</i>	Array os indices de ocorrencias dos includes.
<i>include_number_of_lines</i>	Array com o numero de linhas de cada include
<i>number_of_includes</i>	Numero de includes.
<i>original_file_lines</i>	Linhas do arquivo de entrada original.

Here is the caller graph for this function:



#### 4.2.2.8 readboundaryconditions()

```

subroutine minputreader::readboundaryconditions (
  character(50) keyword,
  integer*4, dimension(:), allocatable kbc,
  real*8, dimension(:), allocatable vbc,
  real(8) default_value )
  
```

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

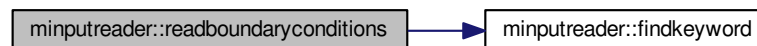
##### Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>kbc</i>	Tipo da CC (1 = Dirichlet, 2 = Neumann).
<i>vbc</i>	Valor prescrito na CC.
<i>default_value</i>	Valor default.

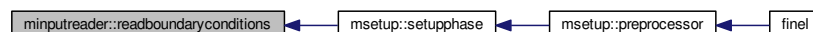
##### Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.2.2.9 readinputfileds()

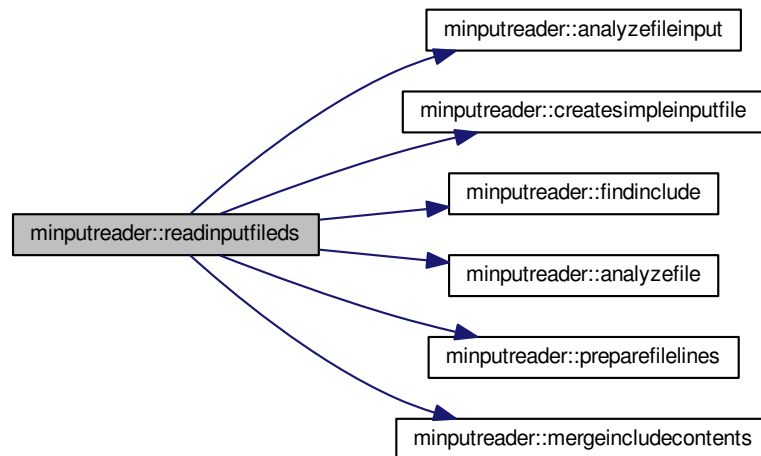
```
subroutine minputreader::readinputfileds ( )
```

Le arquivo de input e armazena seu conteudo em um array.

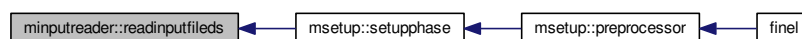
## Parameters

<i>file_name</i>	Nome do arquivo a ser lido.
------------------	-----------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.2.2.10 readintarraykeywordvalue()

```

subroutine minputreader::readintarraykeywordvalue (
    character(50) keyword,
    integer*4, dimension(:), allocatable target,
    integer*4 default_value )

```

Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se não encontrado, associa o valor default fornecido.  
 Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".

## Parameters

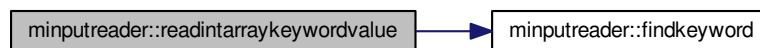
<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variável onde o valor inteiro será atribuído.
<i>default_value</i>	Valor default.



## Author

Diego Volpatto

Here is the call graph for this function:



## 4.2.2.11 readintegerkeywordvalue()

```

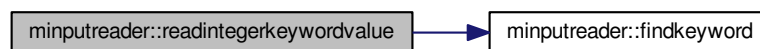
subroutine minputreader::readintegerkeywordvalue (
    character(50) keyword,
    target,
    integer*4, target default_value )
  
```

Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.

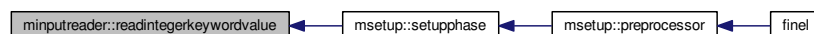
## Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde o valor inteiro sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.2.2.12 readrealkeywordvalue()

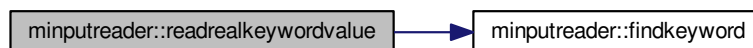
```
subroutine minputreader::readrealkeywordvalue (
    character(50) keyword,
    target,
    real(8), target default_value )
```

Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.

##### Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde o real sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



#### 4.2.2.13 readrealmatrixvalues()

```
subroutine minputreader::readrealmatrixvalues (
    character(50) keyword,
    real*8, dimension(:,:), allocatable target,
    real(8) default_value )
```

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

##### Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde os valores serao atribuido.
<i>default_value</i>	Valor default.

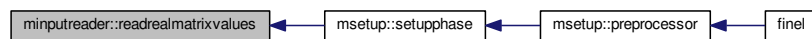
**Author**

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:

**4.2.2.14 readstringkeywordvalue()**

```

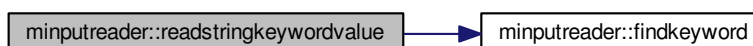
subroutine minputreader::readstringkeywordvalue (
    character(50) keyword,
    character(len=*) target,
    character(len=*) default_value )
  
```

Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.

**Parameters**

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde a string sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.2.3 Variable Documentation

### 4.2.3.1 file\_lines

```
character(len=200), dimension(:), allocatable minputreader::file_lines
```

Armazena as linhas do arquivo de input.

### 4.2.3.2 number\_of\_lines

```
integer*4 minputreader::number_of_lines
```

Armazena o numero de linhas no arquivo.

## 4.3 mio Module Reference

### Functions/Subroutines

- subroutine [openfiles](#) ()  
*Open IO files.*
- subroutine [closefiles](#) ()  
*Close IO files.*
- subroutine [read\\_nodes](#) (mesh\_)  
*Subroutine to read node data file generated by EasyMesh.*
- subroutine [read\\_elems](#) (mesh\_)  
*Subroutine to read element data file generated by EasyMesh.*
- subroutine [print\\_sol1d](#) (mesh\_, scalar\_)

### Variables

- integer, parameter [iin](#) = 111  
*Input file id.*
- integer, parameter [iout](#) = 112  
*Output file id.*
- integer, parameter [isol](#) = 113  
*Solution file id.*
- character(len=20), parameter [infile](#) ='input.dat'  
*Input file name.*
- character(len=20), parameter [outfile](#) ='output.dat'  
*Output file name.*
- character(len=20), parameter [solfile](#) ='solution.dat'  
*Solution file name.*
- character(len=50) [title](#)

### 4.3.1 Function/Subroutine Documentation

#### 4.3.1.1 closefiles()

```
subroutine mio::closefiles ( )
```

Close IO files.

**Author**

Diego T. Volpatto

Here is the caller graph for this function:



#### 4.3.1.2 openfiles()

```
subroutine mio::openfiles ( )
```

Open IO files.

**Author**

Diego T. Volpatto

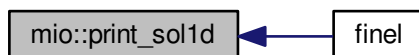
Here is the caller graph for this function:



#### 4.3.1.3 print\_sol1d()

```
subroutine mio::print_sol1d (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Here is the caller graph for this function:



#### 4.3.1.4 read\_elems()

```
subroutine mio::read_elems (
    type(mesh) mesh_ )
```

Subroutine to read element data file generated by EasyMesh.

##### Parameters

<i>mesh</i> ↔	[in/out] a mesh structure
—	

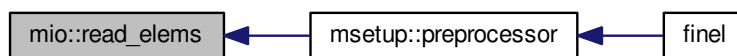
##### Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.1.5 read\_nodes()

```

subroutine mio::read_nodes (
    type(mesh) mesh_ )
  
```

Subroutine to read node data file generated by EasyMesh.

##### Parameters

<i>mesh</i> ↔	[in/out] a mesh structure
_	

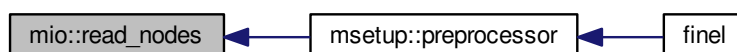
##### Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.3.2 Variable Documentation

### 4.3.2.1 iin

```
integer, parameter mio::iin = 111
```

Input file id.

### 4.3.2.2 infile

```
character(len=20), parameter mio::infile = 'input.dat'
```

Input file name.

### 4.3.2.3 iout

```
integer, parameter mio::iout = 112
```

Output file id.

### 4.3.2.4 isol

```
integer, parameter mio::isol = 113
```

Solution file id.

### 4.3.2.5 outfile

```
character(len=20), parameter mio::outfile = 'output.dat'
```

Output file name.

### 4.3.2.6 solfile

```
character(len=20), parameter mio::solfile = 'solution.dat'
```

Solution file name.

### 4.3.2.7 title

```
character(len=50) mio::title
```

## 4.4 mprocessor Module Reference

Processor module to compute, assemble and solve the system.



## Functions/Subroutines

- subroutine `formkf` (mesh\_, scalar\_)  
*Form and assemble  $Ku = F$  system.*
- subroutine `assmb` (mesh\_, scalar\_, nel)  
*Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.*
- subroutine `drchlt` (mesh\_, scalar\_, n)  
*Apply Dirichlet Boundary Condition.*
- subroutine `neumann1d` (mesh\_, scalar\_, n)  
*Apply Neumann Boundary Condition – 1D. Prescribe  $-k(x)u = vbc$ .*
- subroutine `applybc` (mesh\_, scalar\_)  
*Modify  $Ku=F$  system to incorporate BC data.*
- subroutine `solver` (mesh\_, scalar\_)
- subroutine `processor` (mesh\_, scalar\_)  
*Processor routine phase.*

### 4.4.1 Detailed Description

Processor module to compute, assemble and solve the system.

#### Author

Diego T. Volpatto

### 4.4.2 Function/Subroutine Documentation

#### 4.4.2.1 `applybc()`

```
subroutine mprocessor::applybc (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Modify  $Ku=F$  system to incorporate BC data.

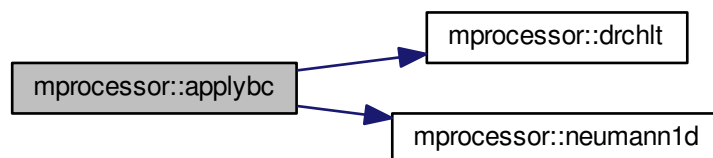
#### Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure

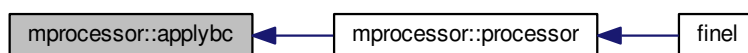
## Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.4.2.2 assmb()

```

subroutine mprocessor::assmb (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel )
  
```

Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.

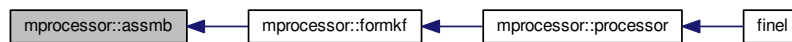
## Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>nel</i>	Index of current element

## Author

Diego Volpatto

Here is the caller graph for this function:

4.4.2.3 `drchlt()`

```

subroutine mprocessor::drchlt (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer n )
  
```

Apply Dirichlet Boundary Condition.

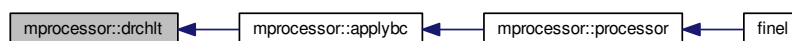
## Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure
<i>n</i>	Node index of BC

## Author

Diego Volpatto

Here is the caller graph for this function:

4.4.2.4 `formkf()`

```

subroutine mprocessor::formkf (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Form and assemble  $Ku = F$  system.

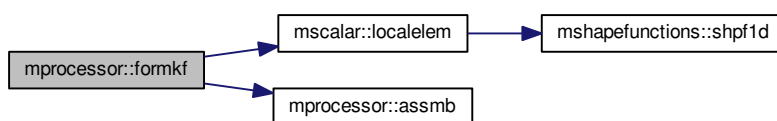
## Parameters

<i>mesh</i> ↔	[in/out] A mesh structure
—	
<i>scalar</i> ↔	[in/out] A scalar structure
—	

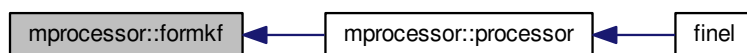
## Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.4.2.5 neumann1d()

```

subroutine mprocessor::neumann1d (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer n )

```

Apply Neumann Boundary Condition – 1D. Prescribe  $-k(x)u = vbc$ .

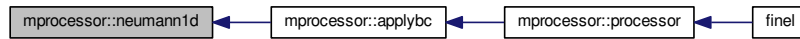
## Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>n</i>	Node index of BC

## Author

Diego Volpatto

Here is the caller graph for this function:



## 4.4.2.6 processor()

```

subroutine mprocessor::processor (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Processor routine phase.

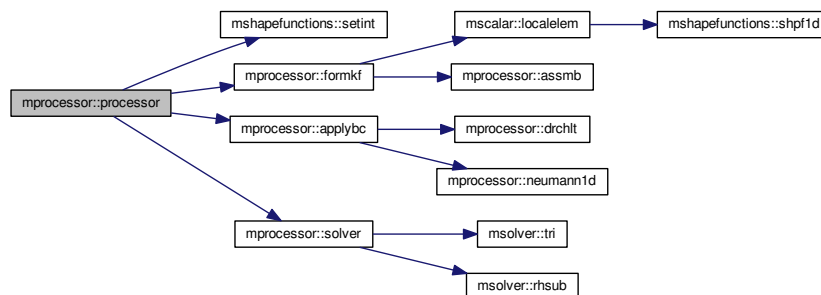
## Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	

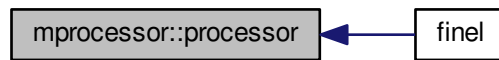
## Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:

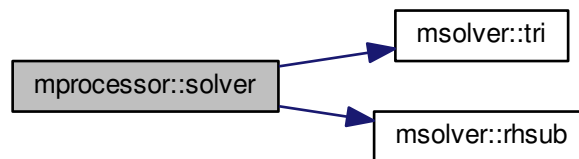


#### 4.4.2.7 solver()

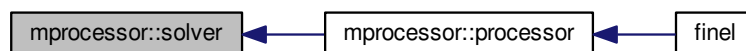
```

subroutine mprocessor::solver (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.5 mscalar Module Reference

Contains variables and subroutine related to a general scalar problem.

### Functions/Subroutines

- subroutine [localelem](#) (mesh\_, scalar\_, nel)  
*Computes a master element contribution – 1D.*

### 4.5.1 Detailed Description

Contains variables and subroutine related to a general scalar problem.

#### Author

Diego T. Volpatto

### 4.5.2 Function/Subroutine Documentation

#### 4.5.2.1 localelem()

```
subroutine mscalar::localelem (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel )
```

Computes a master element contribution – 1D.

#### Parameters

<i>mesh</i> ↔ —	[in/out] A mesh structure
<i>scalar</i> ↔ —	[in/out] A scalar structure
<i>nel</i>	[in] Index of current element

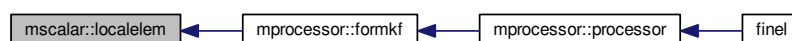
#### Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.6 msetup Module Reference

Module for setup phase by IO procedures.

### Functions/Subroutines

- subroutine [setupphase](#) (mesh\_, scalar\_)
  - subroutine [preprocessor](#) (mesh\_, scalar\_)
- Realizes preprocessor routines.*

### 4.6.1 Detailed Description

Module for setup phase by IO procedures.

#### Author

Diego T. Volpatto

### 4.6.2 Function/Subroutine Documentation

#### 4.6.2.1 preprocessor()

```
subroutine msetup::preprocessor (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Realizes preprocessor routines.

#### Parameters

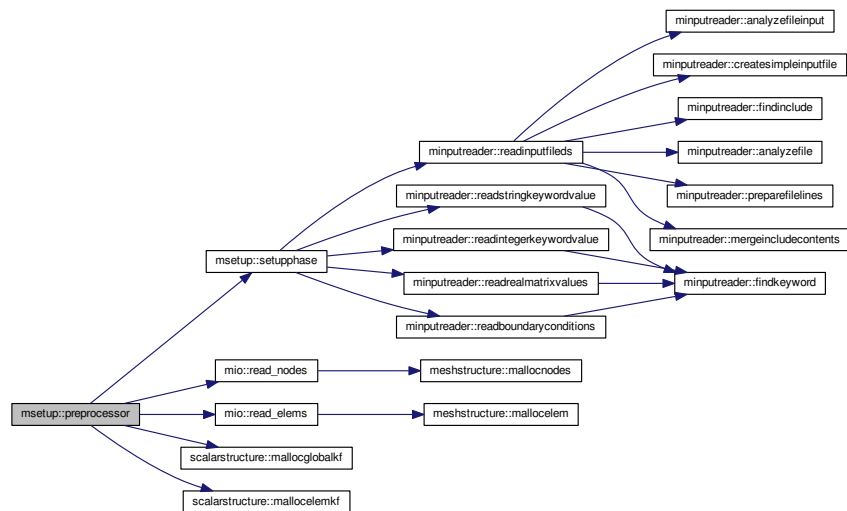
<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure



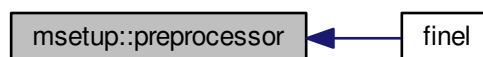
## Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:

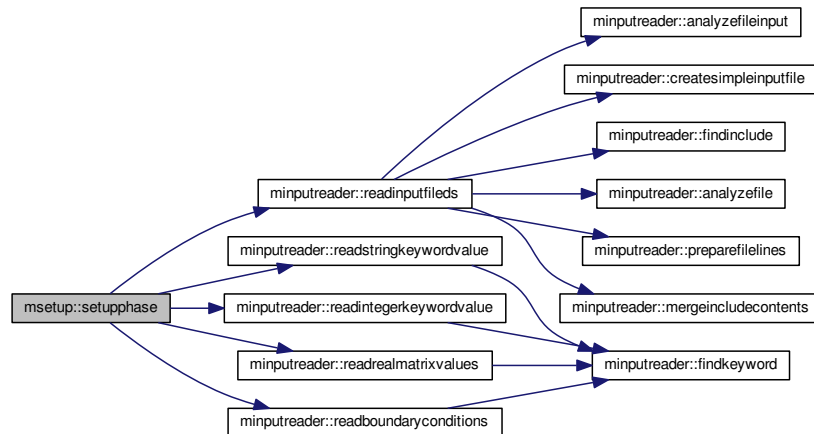


## 4.6.2.2 setupphase()

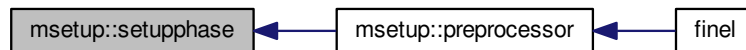
```

subroutine msetup::setupphase (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.7 mshapefunctions Module Reference

Module for shape functions computations and relate operations.

### Functions/Subroutines

- subroutine [setint](#)  
*Gauss quadrature data set routine.*
- subroutine [shpf1d](#) (xl, n, psi, dpsl)  
*Calculates the values of the shape functions and their derivatives.*

### Variables

- real \*8, dimension(4, 4) [xi](#)  
*Gauss point integration.*
- real \*8, dimension(4, 4) [w](#)  
*Gauss weights.*

### 4.7.1 Detailed Description

Module for shape functions computations and relate operations.

Author

Diego T. Volpatto

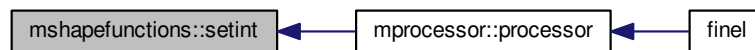
### 4.7.2 Function/Subroutine Documentation

#### 4.7.2.1 setint()

```
subroutine mshapefunctions::setint ( )
```

Gauss quadrature data set routine.

Here is the caller graph for this function:



#### 4.7.2.2 shpf1d()

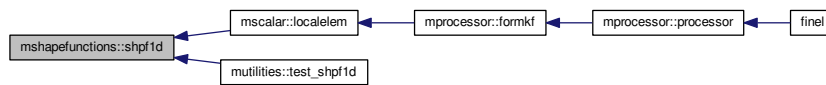
```
subroutine mshapefunctions::shpf1d (
    real*8 xl,
    integer n,
    real*8, dimension(n) psi,
    real*8, dimension(n) dpsi )
```

Calculates the values of the shape functions and their derivatives.

Parameters

<i>xl</i>	[in] specified value of master element coord
<i>n</i>	[in] number of element nodes
<i>psi</i>	[out] shape function values
<i>dpsi</i>	[out] derivatives shape functions values

Here is the caller graph for this function:



### 4.7.3 Variable Documentation

#### 4.7.3.1 w

```
real*8, dimension(4,4) mshapefunctions::w
```

Gauss weights.

#### 4.7.3.2 xi

```
real*8, dimension(4,4) mshapefunctions::xi
```

Gauss point integration.

## 4.8 msolver Module Reference

Contains subroutine to compute numerical solution of linear systems  $Ax=b$ .

### Functions/Subroutines

- subroutine [tri](#) (A, n)  
*Applies Gauss reduction in  $A(n,n)$  to obtain a superior triangular equivalent form.*
- subroutine [rsub](#) (A, x, b, n)  
*Does the forward substitution on the right-side-hand.*

#### 4.8.1 Detailed Description

Contains subroutine to compute numerical solution of linear systems  $Ax=b$ .

The present module has a general purpose such that the routines here intend to be independent of others module.

#### Author

Diego T. Volpatto

### 4.8.2 Function/Subroutine Documentation

#### 4.8.2.1 rsub()

```

subroutine msolver::rsub (
    real*8, dimension(n,n) A,
    real*8, dimension(n) x,
    real*8, dimension(n) b,
    integer n )

```

Does the forward substitution on the right-side-hand.

## Parameters

$A$	[in]A matrix A(n,n)
$x$	[out]Solution vector
$b$	[in/out]RHS-vector
$n$	[in]Number of solution points

## Author

Diego T. Volpatto

Here is the caller graph for this function:



## 4.8.2.2 tri()

```

subroutine msolver::tri (
    real*8, dimension(n,n) A,
    integer n )
  
```

Applies Gauss reduction in A(n,n) to obtain a superior triangular equivalent form.

## Parameters

$A$	[in/out]A matrix A(n,n)
$n$	[in]Number of rows/columns of matrix A

## Author

Diego T. Volpatto

Here is the caller graph for this function:



## 4.9 mutilities Module Reference

Module for auxiliar routines.

## Functions/Subroutines

- subroutine `linspace` (`x1`, `x2`, `nintv`, `x`)  
*Generate points between `x1` and `x2` equally spaced in `x(i)`. Same idea of numpy subroutine.*
- real \*8 function `f1` (`x`)  
*A function to test purpose.*
- subroutine `quad1` (`n`, `x1`, `x2`)  
*Subroutine that computes gaussian quadrature of `f1`.*
- subroutine `test_shpf1d` (`n`, `nelem`, `x`)  
*Check if `shpf1d` works properly.*
- subroutine `print_matrix` (`A`, `n`, `m`)  
*Prints in the screen a matrix `A(n,m)`*

### 4.9.1 Detailed Description

Module for auxiliar routines.

#### Author

Diego T. Volpatto

### 4.9.2 Function/Subroutine Documentation

#### 4.9.2.1 `f1()`

```
real*8 function mutilities::f1 (  
    real*8 x )
```

A function to test purpose.

#### Parameters

<code>x</code>	input coordinate
----------------	------------------

Here is the caller graph for this function:



#### 4.9.2.2 `linspace()`

```
subroutine mutilities::linspace (
```

```

real*8 x1,
real*8 x2,
integer nintv,
real*8, dimension(:), allocatable x )

```

Generate points between x1 and x2 equally spaced in x(i). Same idea of numpy subroutine.

#### Parameters

<i>x1</i>	interval lower bound
<i>x2</i>	interval upper bound
<i>nintv</i>	num of intervals
<i>x</i>	vector to assemble the values

#### 4.9.2.3 print\_matrix()

```

subroutine mutilities::print_matrix (
    real*8, dimension(n,m) A,
    integer n,
    integer m )

```

Prints in the screen a matrix A(n,m)

#### Parameters

<i>A</i>	A matrix
<i>n</i>	Number of lines of A
<i>m</i>	Number of columns of A

#### Author

Diego Volpatto

#### 4.9.2.4 quad1()

```

subroutine mutilities::quad1 (
    integer n,
    real*8 x1,
    real*8 x2 )

```

Subroutine that computes gaussian quadrature of f1.

#### Parameters

<i>n</i>	quadrature order
<i>x1</i>	integral lower bound
<i>x2</i>	integral upper bound

Here is the call graph for this function:



#### 4.9.2.5 test\_shpf1d()

```

subroutine mutilities::test_shpf1d (
    integer n,
    integer nelem,
    real*8, dimension(nelem+1) x )
  
```

Check if shpf1d works properly.

##### Parameters

<i>n</i>	element node numbers
<i>nelem</i>	num of discrete intervals
<i>x</i>	master element's coordinates

Here is the call graph for this function:



## 4.10 scalarstructure Module Reference

Module that contains the data structure of a general scalar problem.

### Data Types

- type [scalarstructuresystem](#)

*Variables and characteristic data for a scalar problem.*



## Functions/Subroutines

- subroutine `mallocglobalkf` (`scalar_`, `n`)  
*Routine to allocate and clear the  $Ku = F$  system.*
- subroutine `mallocelemkf` (`scalar_`, `n`)  
*Routine to allocate and clear the element  $KF$ .*

### 4.10.1 Detailed Description

Module that contains the data structure of a general scalar problem.

#### Author

Diego T. Volpatto

### 4.10.2 Function/Subroutine Documentation

#### 4.10.2.1 mallocelemkf()

```
subroutine scalarstructure::mallocelemkf (
    type(scalarstructuresystem) scalar_,
    integer n )
```

Routine to allocate and clear the element  $KF$ .

#### Parameters

<code>scalar_</code> ↔	[in/out] A general scalar structure
<code>n</code>	[in] Number of element nodes

Here is the caller graph for this function:



#### 4.10.2.2 mallocglobalkf()

```
subroutine scalarstructure::mallocglobalkf (
    type(scalarstructuresystem) scalar_,
    integer n )
```

Routine to allocate and clear the  $Ku = F$  system.

## Parameters

<i>scalar</i> ↔	[in/out] A general scalar structure
—	
<i>n</i>	[in] Number of global nodes

Here is the caller graph for this function:



## Chapter 5

# Data Type Documentation

### 5.1 meshstructure::mesh Type Reference

Data type for a mesh.

Collaboration diagram for meshstructure::mesh:

meshstructure::mesh
<div><div><div><div><div><div></div><div>+ numat</div></div></div><div><div><div></div><div>+ nsd</div></div></div><div><div><div></div><div>+ nintp</div></div></div><div><div><div></div><div>+ nnodes</div></div></div><div><div><div></div><div>+ nelems</div></div></div><div><div><div></div><div>+ nen</div></div></div><div><div><div></div><div>+ x</div></div></div><div><div><div></div><div>+ y</div></div></div><div><div><div></div><div>+ flagnode</div></div></div><div><div><div></div><div>+ nelem</div></div></div><div><div><div></div><div>and 10 more...</div></div></div></div></div></div>

#### Public Attributes

- integer [numat](#)  
*Number of materials.*
- integer [nsd](#)  
*Number of spatial.*
- integer [nintp](#)  
*Number of integration points.*

- integer `nnodes`  
*Number of nodes.*
- integer `nelems`  
*Number of elements.*
- integer `nen`  
*Number of element's nodes.*
- real \*8, dimension(:), allocatable `x`  
*x coordinates nodes*
- real \*8, dimension(:), allocatable `y`  
*y coordinates nodes*
- integer \*4, dimension(:), allocatable `flagnode`  
*boundary flag*
- integer `nelem`  
*Number of elements.*
- real \*8, dimension(:), allocatable `xv`  
*Circumcenter Elem xcoord.*
- real \*8, dimension(:), allocatable `yv`  
*Circumcenter Elem ycoord.*
- integer \*4, dimension(:, :), allocatable `gnode`  
*Global node.*
- integer \*4, dimension(:), allocatable `mat`  
*Element material kind.*
- integer \*4, dimension(:), allocatable `ei`  
*i-opposite element*
- integer \*4, dimension(:), allocatable `ej`  
*j-opposite element*
- integer \*4, dimension(:), allocatable `ek`  
*k-opposite element*
- integer \*4, dimension(:), allocatable `si`  
*Opposite i-side.*
- integer \*4, dimension(:), allocatable `sj`  
*Opposite j-side.*
- integer \*4, dimension(:), allocatable `sk`  
*Opposite k-side.*

### 5.1.1 Detailed Description

Data type for a mesh.

### 5.1.2 Member Data Documentation

#### 5.1.2.1 `ei`

```
integer*4, dimension(:), allocatable meshstructure::mesh::ei
```

i-opposite element

### 5.1.2.2 ej

`integer*4, dimension(:), allocatable meshstructure::mesh::ej`

j-opposite element

### 5.1.2.3 ek

`integer*4, dimension(:), allocatable meshstructure::mesh::ek`

k-opposite element

### 5.1.2.4 flagnode

`integer*4, dimension(:), allocatable meshstructure::mesh::flagnode`

boundary flag

### 5.1.2.5 gnode

`integer*4, dimension(:,:), allocatable meshstructure::mesh::gnode`

Global node.

### 5.1.2.6 mat

`integer*4, dimension(:), allocatable meshstructure::mesh::mat`

Element material kind.

### 5.1.2.7 nelem

`integer meshstructure::mesh::nelem`

Number of elements.

### 5.1.2.8 nelems

`integer meshstructure::mesh::nelems`

Number of elements.

### 5.1.2.9 nen

`integer meshstructure::mesh::nen`

Number of element's nodes.

#### 5.1.2.10 nintp

```
integer meshstructure::mesh::nintp
```

Number of integration points.

#### 5.1.2.11 nnodes

```
integer meshstructure::mesh::nnodes
```

Number of nodes.

#### 5.1.2.12 nsd

```
integer meshstructure::mesh::nsd
```

Number of spatial.

#### 5.1.2.13 numat

```
integer meshstructure::mesh::numat
```

Number of materials.

#### 5.1.2.14 si

```
integer*4, dimension(:), allocatable meshstructure::mesh::si
```

Opposite i-side.

#### 5.1.2.15 sj

```
integer*4, dimension(:), allocatable meshstructure::mesh::sj
```

Opposite j-side.

#### 5.1.2.16 sk

```
integer*4, dimension(:), allocatable meshstructure::mesh::sk
```

Opposite k-side.

#### 5.1.2.17 x

```
real*8, dimension(:), allocatable meshstructure::mesh::x
```

x coordinates nodes

## 5.1.2.18 xv

```
real*8, dimension(:), allocatable meshstructure::mesh::xv
```

Circumcenter Elem xcoor.

## 5.1.2.19 y

```
real*8, dimension(:), allocatable meshstructure::mesh::y
```

y coordinates nodes

## 5.1.2.20 yv

```
real*8, dimension(:), allocatable meshstructure::mesh::yv
```

Circumcenter Elem ycoor.

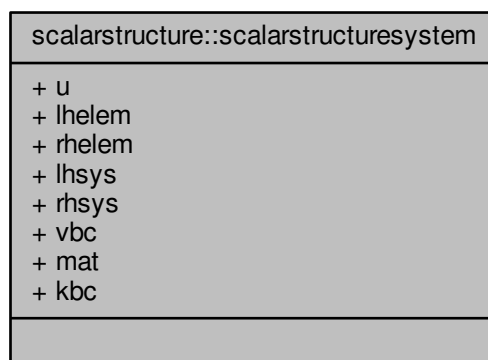
The documentation for this type was generated from the following file:

- [src/meshStructure.F90](#)

## 5.2 scalarstructure::scalarstructuresystem Type Reference

Variables and characteristic data for a scalar problem.

Collaboration diagram for scalarstructure::scalarstructuresystem:



## Public Attributes

- real \*8, dimension(:), allocatable [u](#)  
*Solution vector.*
- real \*8, dimension(:, :), allocatable [lhelem](#)  
*Element left-hand system.*
- real \*8, dimension(:), allocatable [rhelem](#)  
*Element right-hand system.*
- real \*8, dimension(:, :), allocatable [lhsys](#)  
*Global left-hand system.*
- real \*8, dimension(:), allocatable [rhsys](#)  
*Global right-hand system.*
- real \*8, dimension(:), allocatable [vbc](#)  
*BC values vector.*
- real \*8, dimension(:, :), allocatable [mat](#)  
*Material properties values.*
- integer \*4, dimension(:), allocatable [kbc](#)  
*BC kind.*

### 5.2.1 Detailed Description

Variables and characteristic data for a scalar problem.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 kbc

```
integer*4, dimension(:), allocatable scalarstructure::scalarstructuresystem::kbc
```

BC kind.

#### 5.2.2.2 lhelem

```
real*8, dimension(:, :), allocatable scalarstructure::scalarstructuresystem::lhelem
```

Element left-hand system.

#### 5.2.2.3 lhsys

```
real*8, dimension(:, :), allocatable scalarstructure::scalarstructuresystem::lhsys
```

Global left-hand system.

#### 5.2.2.4 mat

```
real*8, dimension(:, :), allocatable scalarstructure::scalarstructuresystem::mat
```

Material properties values.



### 5.2.2.5 rhelem

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::rhelem
```

Element right-hand system.

### 5.2.2.6 rhsys

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::rhsys
```

Global right-hand system.

### 5.2.2.7 u

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::u
```

Solution vector.

### 5.2.2.8 vbc

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::vbc
```

BC values vector.

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

- [src/scalarStructure.F90](#)



## Chapter 6

# File Documentation

### 6.1 src/driver.F90 File Reference

#### Functions/Subroutines

- program [finel](#)

*A FINite ELeMent program for general purpose problems. The present is based in the book "Finite Elements: An Introduction" wrote by Eric Becker, Graham Carey and Tinsley Oden.*

#### 6.1.1 Function/Subroutine Documentation

##### 6.1.1.1 finel()

```
program finel ( )
```

A FINite ELeMent program for general purpose problems. The present is based in the book "Finite Elements: An Introduction" wrote by Eric Becker, Graham Carey and Tinsley Oden.

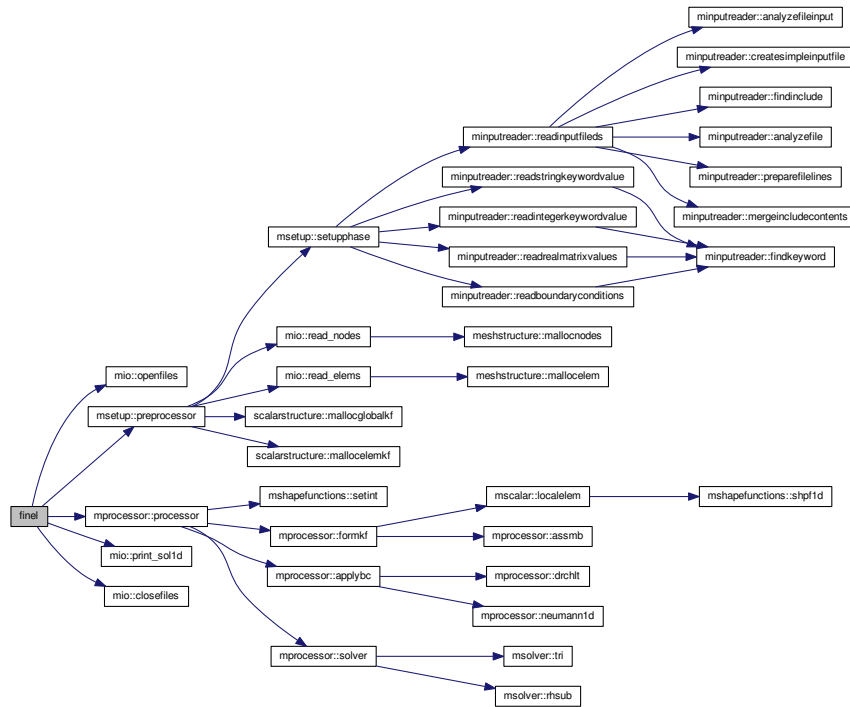
Due to the evolution of Fortran programming language, the code developed here incorporate several changes comparing to the original given in the book cited before. Modular paradigm was employed, as well a little of derived data structure.

Implementations by Diego T. Volpatto. email: [volpatto@lncc.br](mailto:volpatto@lncc.br) or [dtvolpatto@gmail.com](mailto:dtvolpatto@gmail.com)

## Author

Diego Tavares Volpatto

Here is the call graph for this function:



## 6.2 src/io.F90 File Reference

### Modules

- module [mio](#)

### Functions/Subroutines

- subroutine [mio::openfiles](#) ()  
*Open IO files.*
- subroutine [mio::closefiles](#) ()  
*Close IO files.*
- subroutine [mio::read\\_nodes](#) (mesh\_)  
*Subroutine to read node data file generated by EasyMesh.*
- subroutine [mio::read\\_elems](#) (mesh\_)  
*Subroutine to read element data file generated by EasyMesh.*
- subroutine [mio::print\\_sol1d](#) (mesh\_, scalar\_)

## Variables

- integer, parameter `mio::iin` = 111  
*Input file id.*
- integer, parameter `mio::iout` = 112  
*Output file id.*
- integer, parameter `mio::isol` = 113  
*Solution file id.*
- character(len=20), parameter `mio::infile` = 'input.dat'  
*Input file name.*
- character(len=20), parameter `mio::outfile` = 'output.dat'  
*Output file name.*
- character(len=20), parameter `mio::solfile` = 'solution.dat'  
*Solution file name.*
- character(len=50) `mio::title`

## 6.3 src/meshStructure.F90 File Reference

### Data Types

- type `meshstructure::mesh`  
*Data type for a mesh.*

### Modules

- module `meshstructure`  
*Module that contains the data structure of a mesh associate to a problem.*

### Functions/Subroutines

- subroutine `meshstructure::mallocnodes` (meshStrct, n)  
*Routine that allocate memory to node data.*
- subroutine `meshstructure::mallocelem` (meshStrct, n)  
*Routine that allocate memory to element data.*

## 6.4 src/mInputReader.F90 File Reference

### Modules

- module `minputreader`  
*Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.*

## Functions/Subroutines

- subroutine `minputreader::readinputfileds` ()  
*Le arquivo de input e armazena seu conteudo em um array.*
- subroutine `minputreader::createsimpleinputfile` ()  
*Cria a estrutura de input usando um arquivo de entrada sem includes.*
- subroutine `minputreader::mergeincludecontents` (include\_file, include\_line)  
*Le o conteudo do arquivo de include e armazena no array principal.*
- subroutine `minputreader::preparefilelines` (include\_indexes, include\_number\_of\_lines, number\_of\_includes, original\_file\_lines)  
*Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.*
- subroutine `minputreader::analyzefileinput` (number\_of\_lines, number\_of\_includes)  
*Efetua algumas análises no arquivo recebido.*
- subroutine `minputreader::analyzefile` (file\_name, number\_of\_lines, number\_of\_includes)  
*Efetua algumas análises no arquivo recebido.*
- integer \*4 function `minputreader::findinclude` (position, file\_lines, number\_of\_lines)  
*Procura a n-esima palavra-chave include.*
- integer \*4 function `minputreader::findkeyword` (keyword)  
*Procura uma palavra-chave.*
- subroutine `minputreader::readintegerkeywordvalue` (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.*
- subroutine `minputreader::readintarraykeywordvalue` (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se nao encontrado, associa o valor default fornecido. Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".*
- subroutine `minputreader::readstringkeywordvalue` (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.*
- subroutine `minputreader::readrealkeywordvalue` (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.*
- subroutine `minputreader::readrealmatrixvalues` (keyword, target, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.*
- subroutine `minputreader::readboundaryconditions` (keyword, kbc, vbc, default\_value)  
*Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.*

## Variables

- character(len=200), dimension(:), allocatable `minputreader::file_lines`  
*Armazena as linhas do arquivo de input.*
- integer \*4 `minputreader::number_of_lines`  
*Armazena o numero de linhas no arquivo.*

## 6.5 src/processor.F90 File Reference

### Modules

- module `mprocessor`  
*Processor module to compute, assemble and solve the system.*

## Functions/Subroutines

- subroutine `mprocessor::formkf` (mesh\_, scalar\_)  
*Form and assemble  $Ku = F$  system.*
- subroutine `mprocessor::assmb` (mesh\_, scalar\_, nel)  
*Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.*
- subroutine `mprocessor::drchlt` (mesh\_, scalar\_, n)  
*Apply Dirichlet Boundary Condition.*
- subroutine `mprocessor::neumann1d` (mesh\_, scalar\_, n)  
*Apply Neumann Boundary Condition – 1D. Prescribe  $-k(x)u = vbc$ .*
- subroutine `mprocessor::applybc` (mesh\_, scalar\_)  
*Modify  $Ku=F$  system to incorporate BC data.*
- subroutine `mprocessor::solver` (mesh\_, scalar\_)
- subroutine `mprocessor::processor` (mesh\_, scalar\_)  
*Processor routine phase.*

## 6.6 src/scalar.F90 File Reference

### Modules

- module `mscalar`  
*Contains variables and subroutine related to a general scalar problem.*

### Functions/Subroutines

- subroutine `mscalar::localelem` (mesh\_, scalar\_, nel)  
*Computes a master element contribution – 1D.*

## 6.7 src/scalarStructure.F90 File Reference

### Data Types

- type `scalarstructure::scalarstructuresystem`  
*Variables and characteristic data for a scalar problem.*

### Modules

- module `scalarstructure`  
*Module that contains the data structure of a general scalar problem.*

### Functions/Subroutines

- subroutine `scalarstructure::mallocglobalkf` (scalar\_, n)  
*Routine to allocate and clear the  $Ku = F$  system.*
- subroutine `scalarstructure::mallocelemkf` (scalar\_, n)  
*Routine to allocate and clear the element KF.*

## 6.8 src/setup.F90 File Reference

### Modules

- module [msetup](#)  
*Module for setup phase by IO procedures.*

### Functions/Subroutines

- subroutine [msetup::setupphase](#) (mesh\_, scalar\_)
- subroutine [msetup::preprocessor](#) (mesh\_, scalar\_)  
*Realizes preprocessor routines.*

## 6.9 src/shapeFunctions.F90 File Reference

### Modules

- module [mshapefunctions](#)  
*Module for shape functions computations and relate operations.*

### Functions/Subroutines

- subroutine [mshapefunctions::setint](#)  
*Gauss quadrature data set routine.*
- subroutine [mshapefunctions::shpf1d](#) (xl, n, psi, dpsi)  
*Calculates the values of the shape functions and their derivatives.*

### Variables

- real \*8, dimension(4, 4) [mshapefunctions::xi](#)  
*Gauss point integration.*
- real \*8, dimension(4, 4) [mshapefunctions::w](#)  
*Gauss weights.*

## 6.10 src/solver.F90 File Reference

### Modules

- module [msolver](#)  
*Contains subroutine to compute numerical solution of linear systems  $Ax=b$ .*



## Functions/Subroutines

- subroutine `msolver::tri` (A, n)  
*Applies Gauss reduction in  $A(n,n)$  to obtain a superior triangular equivalent form.*
- subroutine `msolver::rsub` (A, x, b, n)  
*Does the forward substitution on the right-side-hand.*

## 6.11 src/utilities.F90 File Reference

### Modules

- module `mutilities`  
*Module for auxiliar routines.*

### Functions/Subroutines

- subroutine `mutilities::linspace` (x1, x2, nintv, x)  
*Generate points between x1 and x2 equally spaced in  $x(i)$ . Same idea of numpy subroutine.*
- real \*8 function `mutilities::f1` (x)  
*A function to test purpose.*
- subroutine `mutilities::quad1` (n, x1, x2)  
*Subroutine that computes gaussian quadrature of f1.*
- subroutine `mutilities::test_shpf1d` (n, nelem, x)  
*Check if shpf1d works properly.*
- subroutine `mutilities::print_matrix` (A, n, m)  
*Prints in the screen a matrix  $A(n,m)$*



# Index

analyzefile  
    minputreader, 10  
analyzefileinput  
    minputreader, 11  
applybc  
    mprocessor, 25  
assmb  
    mprocessor, 26  
  
closefiles  
    mio, 21  
createsimpleinputfile  
    minputreader, 11  
  
drchlt  
    mprocessor, 27  
driver.F90  
    finel, 51  
  
ei  
    meshstructure::mesh, 44  
ej  
    meshstructure::mesh, 44  
ek  
    meshstructure::mesh, 45  
  
f1  
    mutilities, 38  
file\_lines  
    minputreader, 20  
findinclude  
    minputreader, 12  
findkeyword  
    minputreader, 12  
finel  
    driver.F90, 51  
flagnode  
    meshstructure::mesh, 45  
formkf  
    mprocessor, 27  
  
gnode  
    meshstructure::mesh, 45  
  
iin  
    mio, 24  
infile  
    mio, 24  
iout  
    mio, 24  
isol  
    mio, 24  
  
kbc  
    scalarstructure::scalarstructuresystem, 48  
  
lhelem  
    scalarstructure::scalarstructuresystem, 48  
lhsys  
    scalarstructure::scalarstructuresystem, 48  
linspace  
    mutilities, 38  
localelem  
    mscalar, 31  
  
mallocelem  
    meshstructure, 7  
mallocelemkf  
    scalarstructure, 41  
mallocglobalkf  
    scalarstructure, 41  
mallocnodes  
    meshstructure, 8  
mat  
    meshstructure::mesh, 45  
    scalarstructure::scalarstructuresystem, 48  
mergeincludecontents  
    minputreader, 13  
meshstructure, 7  
    mallocelem, 7  
    mallocnodes, 8  
meshstructure::mesh, 43  
    ei, 44  
    ej, 44  
    ek, 45  
    flagnode, 45  
    gnode, 45  
    mat, 45  
    nelem, 45  
    nelems, 45  
    nen, 45  
    nintp, 45  
    nnodes, 46  
    nsd, 46  
    numat, 46  
    si, 46  
    sj, 46  
    sk, 46  
    x, 46  
    xv, 46  
    y, 47

- yv, 47
- minputreader, 8
  - analyzefile, 10
  - analyzefileinput, 11
  - createsimpleinputfile, 11
  - file\_lines, 20
  - findinclude, 12
  - findkeyword, 12
  - mergeincludecontents, 13
  - number\_of\_lines, 20
  - preparefilelines, 13
  - readboundaryconditions, 14
  - readinputfiles, 14
  - readintarraykeywordvalue, 16
  - readintegerkeywordvalue, 17
  - readrealkeywordvalue, 17
  - readrealmatrixvalues, 18
  - readstringkeywordvalue, 19
- mio, 20
  - closefiles, 21
  - iin, 24
  - infile, 24
  - iout, 24
  - isol, 24
  - openfiles, 21
  - outfile, 24
  - print\_sol1d, 21
  - read\_elems, 22
  - read\_nodes, 23
  - solfile, 24
  - title, 24
- mprocessor, 24
  - applybc, 25
  - assmb, 26
  - drchlt, 27
  - formkf, 27
  - neumann1d, 28
  - processor, 29
  - solver, 30
- mscalar, 30
  - localelem, 31
- msetup, 32
  - preprocessor, 32
  - setupphase, 33
- mshapefunctions, 34
  - setint, 35
  - shpf1d, 35
  - w, 36
  - xi, 36
- msolver, 36
  - rhsb, 36
  - tri, 37
- mutilities, 37
  - f1, 38
  - linspace, 38
  - print\_matrix, 39
  - quad1, 39
  - test\_shpf1d, 40
- nelem
  - meshstructure::mesh, 45
- nelems
  - meshstructure::mesh, 45
- nen
  - meshstructure::mesh, 45
- neumann1d
  - mprocessor, 28
- nintp
  - meshstructure::mesh, 45
- nnodes
  - meshstructure::mesh, 46
- nsd
  - meshstructure::mesh, 46
- numat
  - meshstructure::mesh, 46
- number\_of\_lines
  - minputreader, 20
- openfiles
  - mio, 21
- outfile
  - mio, 24
- preparefilelines
  - minputreader, 13
- preprocessor
  - msetup, 32
- print\_matrix
  - mutilities, 39
- print\_sol1d
  - mio, 21
- processor
  - mprocessor, 29
- quad1
  - mutilities, 39
- read\_elems
  - mio, 22
- read\_nodes
  - mio, 23
- readboundaryconditions
  - minputreader, 14
- readinputfiles
  - minputreader, 14
- readintarraykeywordvalue
  - minputreader, 16
- readintegerkeywordvalue
  - minputreader, 17
- readrealkeywordvalue
  - minputreader, 17
- readrealmatrixvalues
  - minputreader, 18
- readstringkeywordvalue
  - minputreader, 19
- rhelem
  - scalarstructure::scalarstructuresystem, 48
- rhsb

- msolver, 36
- rhsys
  - scalarstructure::scalarstructuresystem, 49
- scalarstructure, 40
  - mallocelemkf, 41
  - mallocglobalkf, 41
- scalarstructure::scalarstructuresystem, 47
  - kbc, 48
  - lhelem, 48
  - lhsys, 48
  - mat, 48
  - rhelem, 48
  - rhsys, 49
  - u, 49
  - vbc, 49
- setint
  - mshapefunctions, 35
- setupphase
  - msetup, 33
- shpf1d
  - mshapefunctions, 35
- si
  - meshstructure::mesh, 46
- sj
  - meshstructure::mesh, 46
- sk
  - meshstructure::mesh, 46
- solfile
  - mio, 24
- solver
  - mprocessor, 30
- src/driver.F90, 51
- src/io.F90, 52
- src/mInputReader.F90, 53
- src/meshStructure.F90, 53
- src/processor.F90, 54
- src/scalar.F90, 55
- src/scalarStructure.F90, 55
- src/setup.F90, 56
- src/shapeFunctions.F90, 56
- src/solver.F90, 56
- src/utilities.F90, 57
- test\_shpf1d
  - mutilities, 40
- title
  - mio, 24
- tri
  - msolver, 37
- u
  - scalarstructure::scalarstructuresystem, 49
- vbc
  - scalarstructure::scalarstructuresystem, 49
- w
  - mshapefunctions, 36
- x
  - meshstructure::mesh, 46
- xi
  - mshapefunctions, 36
- xv
  - meshstructure::mesh, 46
- y
  - meshstructure::mesh, 47
- yv
  - meshstructure::mesh, 47