

文部科学省次世代IT基盤構築のための研究開発
「革新的シミュレーションソフトウェアの研究開発」

RSS21 フリーソフトウェア

HEC ミドルウェア (HEC-MW)

PC クラスタ用ライブラリ型 HEC-MW

(hecmw-PC-cluster) バージョン 2.01

IO, 全体制御 マニュアル

本ソフトウェアは文部科学省次世代IT基盤構築のための研究開発「革新的シミュレーションソフトウェアの研究開発」プロジェクトによる成果物です。本ソフトウェアを無償でご使用になる場合「RSS21 フリーソフトウェア使用許諾条件」をご了承頂くことが前提となります。営利目的の場合には別途契約の締結が必要です。これらの契約で明示されていない事項に関して、或いは、これらの契約が存在しない状況においては、本ソフトウェアは著作権法など、関係法令により、保護されています。

お問い合わせ先

(公開／契約窓口) (財)生産技術研究奨励会

〒153-8505 東京都目黒区駒場4-6-1

(ソフトウェア管理元) 東京大学生産技術研究所 計算科学技術連携研究センター

〒153-8505 東京都目黒区駒場4-6-1

Fax : 03-5452-6662

目 次

1. 概要	1
2. 全体制御機能	2
2.1. 初期化処理	2
2.2. 終了処理	2
3. 全体制御ファイル	3
3.1. 全体制御ファイル概要	3
3.2. 入力規則	3
(1) ヘッダ	3
(2) ヘッダ行	3
(3) データ行	4
(4) コメント行	4
(5) 区切り文字	4
(6) 空白の扱い	4
(7) 名前	4
(8) ファイル名	4
(9) 浮動小数点データ	5
3.3. ヘッダー一覧	5
!!, #	6
!CONTROL	7
!MESH	8
!MESH GROUP	10
!RESTART	12
!RESULT	14
4. HEC-MW 単一領域メッシュデータ	16
4.1. HEC-MW 単一領域メッシュデータ概要	16
4.2. 入力規則	16
(1) ヘッダ	16
(2) ヘッダ行	16
(3) データ行	17
(4) コメント行	17
(5) 区切り文字	17
(6) 空白の扱い	17
(7) 名前	17
(8) ファイル名	18

(9) 浮動小数点データ	18
4.3. ヘッダー一覧	18
!!, #	20
!AMPLITUDE	21
!EGROUP	23
!ELEMENT	25
!END	28
!EQUATION	29
!HEADER	31
!INITIAL CONDITION	32
!MATERIAL	34
!NGROUP	40
!NODE	42
!SECTION	44
!SGROUP	47
!ZERO	49
4.4. 特記事項	50
(1) !ELEMENT における材料物性値	50
(2) !EQUATION に関する処理	50
5. 分散メッシュデータ構造体の概要	51
5.1. 一般パラメータ設定	52
5.2. 分散メッシュ情報	53
(1) 全体情報	53
(2) 節点情報	54
(3) 要素情報	55
(4) PE および通信情報	57
(5) 階層構造	58
(6) 下部構造	58
5.3. セクション情報	59
5.4. 材料物性情報	60
5.5. 拘束グループ情報	61
5.6. AMPLITUDE 情報	61
5.7. グループ情報 (節点)	62
5.8. グループ情報 (要素)	63
5.9. グループ情報 (面)	64
6. 結果データ	65

6.1.	結果データ構造体.....	65
6.2.	結果ファイルフォーマット	66
7.	Fortan API リファレンス.....	67
8.	C API リファレンス.....	95
8.1.	データ構造一覧	95
8.2.	HEC-MW データ構造	97
8.3.	HEC-MW ファイル一覧.....	204
8.4.	HEC-MW ファイル	206
9.	HEC-MW 要素ライブラリ	485
10.	HEC-MW を用いたプログラム作成方法	498
10.1.	はじめに	498
10.2.	コンパイル・リンク方法	498
10.3.	最小の HEC-MW 利用プログラム	499
10.4.	メッシュデータ入力方法	500
10.5.	メッシュデータ出力方法	501
10.6.	結果データ出力方法.....	502
10.7.	結果データ入力方法.....	504
10.8.	リスタートデータ出力方法.....	505
10.9.	リスタートデータ入力方法.....	506
10.10.	可視化方法（メモリ渡し）	507

1. 概要

並列計算を実施する場合、大規模かつ分散したデータを効率的に扱うことが必要である。特に並列 I/O については、利用者が容易に大規模分散データの処理を実施できるような工夫が求められる。

HEC-MW では、大規模分散データの処理に必要と考えられる以下の並列 I/O について、容易に操作可能なインタフェースを提供する。

- 全体メッシュデータ入力
- 分散メッシュデータ入出力
- リスタートデータ入出力
- 結果データ入出力

また、全体メッシュデータ入力においては、以下のメッシュデータが読み込み可能である。

- HEC-MW 単一領域メッシュデータ
- GeoFEM メッシュデータ
- ABAQUS メッシュデータ（現版では未実装）
- NASTRAN メッシュデータ（現版では未実装）
- FEMAP メッシュデータ（現版では未実装）

それらの入力はいずれ一つのインタフェースによって行うことができる。さらに、異なるタイプのメッシュデータを同時に入力し、一つのメッシュデータとして扱う事も可能である。

本マニュアルは、HEC-MW の機能のうち、全体制御機能と並列 I/O についてその利用方法を述べたものである。

2. 全体制御機能

2.1. 初期化处理

初期化处理は、各機能が正常に動作するために必要な処理を行う。

HEC-MW を利用するにあたり、ユーザは任意の機能呼び出して利用することができるが、その前に必ず初期化处理を行わなければならない。

MPI の初期化はここで行われる。

また、全体制御ファイルは HEC-MW の利用に必須であるが、その読み込みは初期化处理によって自動的に行われる。

2.2. 終了処理

HEC-MW の利用を終えるには、終了処理を行う必要がある。

MPI の終了処理はここで行われる。

3. 全体制御ファイル

3.1. 全体制御ファイル概要

全体制御ファイルは、HEC-MW における制御情報を記述するためのファイルである。
全体制御ファイルの特徴は以下のとおりである。

- 自由書式に基づく ASCII 形式のファイルである。
- 「！」で始まるヘッダとそれに続くデータから構成されている。
- ヘッダの記述の順番は基本的に自由である。
- データの区切り記号には「,」を使用する。

3.2. 入力規則

全体制御ファイルは、ヘッダ行、データ行、コメント行から構成される。
ヘッダ行には必ず一つのヘッダが含まれる。

(1) ヘッダ

全体制御ファイル内で、データの意味とデータブロックを特定する。
行頭が「！」で始まる場合、ヘッダであるとみなされる。

(2) ヘッダ行

ヘッダとそれに伴うパラメータを記述する。

ヘッダ行はヘッダで始まっていなければならない。パラメータが必要な場合は、「,」を用いてその後に続けなければならない。パラメータが値をとる場合は、パラメータの後に「=」が続き、その後に値を記述する。

ヘッダ行を複数行にわたって記述することはできない。

(3) データ行

ヘッダ行の次の行から開始され、必要なデータを記述する。

データ行は複数行にわたる可能性があるが、それは各ヘッダで定義されるデータ記述の規則により決定される。

データ行は必要ない場合もある。

(4) コメント行

行頭が「!!」または「#」で始まる行はコメント行とみなされ、無視される。

コメント行はファイル中の任意の位置に挿入でき、その数に制限はない。

(5) 区切り文字

データの区切り文字にはカンマ「,」を用いる。

(6) 空白の扱い

空白は無視される。

(7) 名前

名前に使用可能な文字は、アンダースコア「_」、ハイフン「-」、英数字「a-zA-Z0-9」であるが、最初の一文字は「_」または英字「a-zA-Z」で始まっていなければならない。大文字小文字の区別はなく、内部的には全て大文字として扱われる。

また、名前の最大長は 63 文字である。

(8) ファイル名

ファイル名に使用可能な文字は、アンダースコア「_」、ハイフン「-」、ピリオド「.」、ス

ラッシュ「/」，英数字「a-zA-Z0-9」である．

ファイル名は，特に記述がない限りパスを含んでもよい．相対パス，絶対パスのいずれも指定可能である．

また，ファイル名の最大長は 1023 文字である．

(9) 浮動小数点データ

指数はあってもなくてもよい．指数の前には，「E」または「e」の記号をつけなければならない．

3.3. ヘッダー一覧

現在，HEC-MW 全体制御ファイルは以下のヘッダによって構成されている．

!CONTROL	制御ファイル定義
!MESH	メッシュファイル定義
!MESH GROUP	メッシュグループ定義
!RESTART	リスタートファイル定義
!RESULT	結果ファイル定義
!COUPLE MESH	連成メッシュ定義
!COUPLE	連成定義

各ヘッダには，パラメータとそれぞれのヘッダに対応したデータの項目がある．

以下，上記各ヘッダについてデータ作成例とともに簡単に説明する．

!!,

コメント行先頭フラグ「!!」または「#」が先頭にある行はコメント行とみなされ、無視される。

1 行目

!!<comment>

又は

#<comment>

パラメータ

なし

2 行目以降

なし

使用例

```
!! mesh file for input
!MESH, NAME=in-mesh, TYPE=HECMW-ENTIRE
mesh.in

# mesh file for output
!MESH, NAME=out-mesh, TYPE=HECMW-DIST
mesh.out
```

!CONTROL

制御ファイルを指定する.

1 行目

```
!CONTROL, NAME=<name>
```

パラメータ	
NAME	識別子 (必須)

パラメータ名	パラメータ値	内 容
NAME	<name>	識別子

2 行目以降

(2 行目) file

変数名	内容
File	制御ファイル名 (相対パス, 絶対パス共に指定可能. 相対パスの場合はカレントディレクトリからのパスとなる)

使用例

```
!CONTROL, NAME=myctrl  
myctrl.dat
```

!MESH

メッシュファイルを指定する.

1 行目

!MESH, NAME=<name>, TYPE=<type>[,optional parameter]

パラメータ	
NAME	識別子 (必須)
TYPE	メッシュタイプ (必須)
IO	入出力指定 (省略可)

パラメータ名	パラメータ値	内 容
NAME	<name>	識別子
TYPE	HECMW-DIST	HEC-MW 分散メッシュデータ
	HECMW-ENTIRE	HEC-MW 全体メッシュデータ
	GeoFEM (現版では使用不可)	GeoFEM メッシュデータ
	ABAQUS (現版では使用不可)	ABAQUS メッシュデータ
	NASTRAN (現版では使用不可)	NASTRAN バルクデータ
	FEMAP (現版では使用不可)	FEMAP メッシュデータ
IO	IN	入力用 (デフォルト)
	OUT	出力用

2 行目以降

(2 行目) fileheader

変数名	内容
Fileheader	制御ファイル名のヘッダ (相対パス, 絶対パス共に指定可能. 相対パスの場合はカレントディレクトリからのパスとなる)

注意

IO パラメータはユーザが自由に使用してよい. IO パラメータの有無, パラメータ値は他に何も影響を与えない.

TYPE が HECMW-DIST の場合, データ行に指定する fileheader は, ファイル名末尾の「.<rank>」を除いたものである.

使用例

```
!MESH, NAME=input-mesh, TYPE=HECMW-ENTIRE  
mesh.in
```

!MESH GROUP

メッシュグループの定義

!MESH で定義されたメッシュファイルをグループ化する.

1 行目

!MESH, GROUP, NAME=<name>

パラメータ	
NAME	識別子 (必須)

パラメータ名	パラメータ値	内容
NAME	<name>	識別子

2 行目以降

(2 行目) mesh1, mesh2, ...

(以下同様)

変数名	内容
meshX	メッシュファイル識別子 (!MESH の NAME)

注意

NAME は, !MESH のそれと共有される. したがって, !MESH と同じ NAME を指定することはできない.

メッシュファイル識別子で識別されるメッシュファイルが!MESH で定義されていない
ければエラーとなる.

使用例

```
!MESH GROUP, NAME=input-mesh-grp  
input-mesh, input-mesh-mat
```

!RESTART

リスタートファイルを指定する.

1 行目

!RESTART, NAME=<name>, IO=<io>

パラメータ	
NAME	識別子 (必須)
IO	入出力指定 (必須)

パラメータ名	パラメータ値	内容
NAME	<name>	識別子
IO	IN	入力用
	OUT	出力用
	INOUT	入出力両用

2 行目以降

(2 行目) fileheader

変数名	内容
fileheader	制御ファイル名 (相対パス, 絶対パス共に指定可能. 相対パスの場合はカレントディレクトリからのパスとなる)

注意

この定義によって生成されるファイル名は, fileheader+.<rank>となる.

使用例

```
!RESTART, NAME=restart-in, IO=IN  
restart.in
```

!RESULT

結果ファイルを指定する.

1 行目

!RESULT, NAME=<name>,IO=<io>

パラメータ	
NAME	識別子 (必須)
IO	入出力指定 (必須)

パラメータ名	パラメータ値	内容
NAME	<name>	識別子
IO	IN	入力用
	OUT	出力用

2 行目以降

(2 行目) file

変数名	内容
fileheader	制御ファイル名 (相対パス, 絶対パス共に指定可能. 相対パスの場合はカレントディレクトリからのパスとなる)

注意

この定義によって生成されるファイル名は, fileheader+.<rank>となる.

使用例

```
!RESULT, NAME=result-out, IO=OUT  
result.out
```

4. HEC-MW 単一領域メッシュデータ

4.1. HEC-MW 単一領域メッシュデータ概要

単一領域メッシュデータは、HEC-MW においてメッシュデータ入力を行うためのファイルフォーマットの一つである。

単一領域メッシュデータの特徴は以下のとおりである。

- 自由書式に基づく ASCII 形式のファイルである。
- 「!」で始まるヘッダとそれに続くデータから構成されている。
- ヘッダの記述の順番は基本的に自由である。
- データの区切り記号には「,」を使用する。

4.2. 入力規則

単一領域メッシュデータは、ヘッダ行、データ行、コメント行から構成される。ヘッダ行には必ず一つのヘッダが含まれる。

(1) ヘッダ

単一領域メッシュデータ内で、データの意味とデータブロックを特定する。行頭が「!」で始まる場合、ヘッダであるとみなされる。

(2) ヘッダ行

ヘッダとそれに伴うパラメータを記述する。

ヘッダ行はヘッダで始まっていなければならない。パラメータが必要な場合は、「,」を用いてその後に続けなければならない。パラメータが値をとる場合は、パラメータの後に「=」が続き、その後に値を記述する。

ヘッダ行を複数行にわたって記述することはできない。

(3) データ行

ヘッダ行の次の行から開始され、必要なデータを記述する。

データ行は複数行にわたる可能性があるが、それは各ヘッダで定義されるデータ記述の規則により決定される。

データ行は必要ない場合もある。

(4) コメント行

行頭が「!!」または「#」で始まる行はコメント行とみなされ、無視される。

コメント行はファイル中の任意の位置に挿入でき、その数に制限はない。

(5) 区切り文字

データの区切り文字にはカンマ「,」を用いる。

(6) 空白の扱い

空白は無視される。

(7) 名前

グループ名、材料名などの名前に使用可能な文字は、アンダースコア「_」、ハイフン「-」、英数字「a-zA-Z0-9」であるが、最初の一文字は「_」または英字「a-zA-Z」で始まっていなければならない。大文字小文字の区別はなく、内部的には全て大文字として扱われる。

なお、以下の名前は予約済みのため使用不可である。

- 「HECMW」で始まる名前
- 節点グループ名「ALL」, 「EQUATION_BLOCK」
- 要素グループ名「ALL」

また、名前の最大長は 63 文字である。

(8) ファイル名

ファイル名に使用可能な文字は、アンダースコア「_」、ハイフン「-」、ピリオド「.」、スラッシュ「/」、英数字「a-zA-Z0-9」である。

ファイル名は、特に記述がない限りパスを含んでもよい。相対パス、絶対パスのいずれも指定可能である。

また、ファイル名の最大長は 1023 文字である。

(9) 浮動小数点データ

指数はあってもなくてもよい。指数の前には、「E」または「e」の記号をつけなければならない。

4.3. ヘッダー一覧

HEC-MW 単一領域メッシュデータは以下のヘッダによって構成されている。

ヘッダ名	内容
!AMPLITUDE	非定常荷重
!ELEMENT	コネクティビティ
!EGROUP	要素グループ
!ELEMENT	要素情報
!EQUATION	拘束点情報
!END	読み込み終了
!HEADER	メッシュファイルのタイトル
!INCLUDE	外部ファイル入力
!INITIAL CONDITION	初期条件
!MATERIAL	材料情報
!NGROUP	節点グループ
!NODE	節点情報
!SECTION	セクション情報

!SGROUP	面グループ
!ZERO	絶対零度

各ヘッダには、パラメータとそれぞれのヘッダに対応したデータの項目がある。以下、上記各ヘッダについてデータ作成例とともに簡単に説明する。

!!,

コメント行先頭フラグ「!!」または「#」が先頭にある行はコメント行とみなされ、無視される。

1 行目

!!<comment>

又は

#<comment>

パラメータ

なし

2 行目以降

なし

使用例

```
!NODE
!! location of each node is defined by special code.
1, 0.0, 0.1, 0.2
# node 2 is on the surface
2, 3.0, 4.0, 5.0
```


!AMPLITUDE

ステップ内での荷重条件を与える変数の時間変化を指定する.

1 行目

!AMPLITUDE, NAME=<name> [, optional parameter]

パラメータ	
NAME	名前 (必須)
DEFINITION	タイプ (省略可)
TIME	時間の種類 (省略可)
VALUE	値の種類 (省略可)
INPUT	外部ファイル名 (省略可)

パラメータ名	パラメータ値	内 容
NAME	<name>	AMPLITUDE 名
DEFINITION	TABULAR	デフォルト (現版ではデフォルトのみ許す)
TIME	STEP TIME	デフォルト (現版ではデフォルトのみ許す)
VALUE	RELATIVE	相対値 (デフォルト)
	ABSOLUTE	絶対値
INPUT	<filename>	外部ファイル名 (省略可), 2 行目以降との併用も可能

2 行目以降

(2 行目以降) VAL1, T1, VAL2, T2, VAL3, T3 ... (一行に四項まで)
以下繰り返し

変数名	属性	内容
VAL1	R	時間 T1 における値

T1	R	時間 T1
VAL2	R	時間 T2 における値
T2	R	時間 T2
VAL3	R	時 T3 における値
T3	R	時間 T3

!EGROUP

要素グループの定義

1 行目

!EGROUP, EGRP=<egrp> [, optional parameter]

パラメータ	
EGRP	要素グループ名（必須）
GENERATE	要素グループに属する節点の自動生成（省略可）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
EGRP	<egrp>	要素グループ名
GENERATE	なし	要素グループに属する節点の自動生成
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降（GENERATE を使用しない場合）

（2 行目） elem1, elem2, elem3 ...

（以下同様）

変数名	属性	内 容
elemX	I	要素グループに属する要素番号

2 行目以降（GENERATE を使用する場合）

（2 行目） elem1, elem2, elem3

(以下同様)

変数名	属 性	内 容
Elem1	I	要素グループ内の最初の要素番号
Elem2	I	要素グループ内の最後の要素番号
Elem3	I	要素番号増分 (省略可能, 省略時は elem3=1 となる)

注意

- 1 行に任意の数の要素を入れることができる。また次のオプションが始まるまで、任意の数の行を挿入することができる。
- 指定する要素は「!EGROUP」より前に定義されている必要がある。
- 「!ELEMENT」オプションで定義されていない要素は除外され、警告メッセージが表示される。
- 指定された要素が既に同じグループ内に存在する場合は無視され、警告メッセージが表示される。
- 全ての要素は、「ALL」という名前の要素グループに属している (自動的に生成される)。
- 一つのグループを複数回にわけて定義できる。

使用例

<pre>!EGROUP, EGRP= EA01 1, 2, 3, 4, 5, 6 101, 102 205 !EGROUP, EGRP= EA02 101, 102 !EGROUP, EGRP= EA01 501, 505 !EGROUP, EGRP= EA04, GENERATE 301, 309, 2 311, 313</pre>	<p>グループ「EA01」に「501, 505」が追加される。</p> <p>グループ「EA04」に 「301, 303, 305, 307, 309, 311, 312, 313」が追加される。</p>
---	--

!ELEMENT

要素の定義

1 行目

!ELEMENT, TYPE=<type> [, optional parameter]

パラメータ	
TYPE	要素タイプ（必須）
EGRP	要素グループ（省略可）
MATITEM	材料物性を要素ごとに定義する場合の物性項目数（セクションごとに物性を定義する場合は使用しない）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
TYPE	111	ロッド,リンク, トラス要素 (1 次)
	112	ロッド, リンク, トラス要素 (2 次)
	231	三角形要素 (1 次)
	232	三角形要素 (2 次)
	241	四角形要素 (1 次)
	242	四角形要素 (2 次)
	341	四面体要素 (1 次)
	342	四面体要素 (2 次)
	351	三角柱要素 (1 次)
	352	三角柱要素 (2 次)
	361	六面体要素 (1 次)
	362	六面体要素 (2 次)
	371	四角錐要素 (1 次)
	372	四角錐要素 (2 次)
	431	マスタースレーブ要素 (三角形断面, 1 次)
	432	マスタースレーブ要素 (三角形断面, 2 次)

	441	マスタースレーブ要素（四角形断面，1 次）
	442	マスタースレーブ要素（四角形断面，2 次）
	531	インタフェース要素（三角形断面，1 次）
	532	インタフェース要素（三角形断面，2 次）
	541	インタフェース要素（四角形断面，1 次）
	542	インタフェース要素（四角形断面，2 次）
	611	梁要素（1 次）
	612	梁要素（2 次）
	731	三角形シェル要素（1 次）
	732	三角形シェル要素（2 次）
	741	四角形シェル要素（1 次）
	742	四角形シェル要素（2 次）
EGRP	<egrp>	要素グループ（省略可）
MATITEM	<matiem>	材料物性を要素ごとに定義する場合の物性項目数
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

（2 行目）ELEM_ID, nod1, nod2, nod3, ..., MAT1, MAT2, ...

以下同様

変数名	属 性	内 容
ELEM_ID	I	要素番号
nodX	I	コネクティビティ
MATy	R	要素ごとの物性値

注意

- 要素タイプ、コネクティビティについての詳細は、「付録 HEC-MW 要素ライブラリ」を参照のこと。
- 要素番号は連続している必要はない。

- 「!ELEMENT」オプションは何回でも定義できる。
- 要素番号は自然数でなければならない。省略は不可。
- 同じ要素番号を重複して使用する場合、最後に入力した値が使用される。この場合、警告メッセージが出力される。
- 定義されていない節点をコネクティビティに使用することはできない。
- パラメータ「MATITEM」を使用して要素ごとに物性を入力した場合の値が優先して使用される。この場合、後述の「!SECTION」, 「!MATERIAL」オプションを使用して入力した物性値は使用されない。
- 一つの要素の定義を複数行にわたって記述してもよい。

使用例

```
!ELEMENT, TYPE=231
1, 1, 2, 3
2, 4, 8, 5
4, 6, 7, 8
!ELEMENT, TYPE=361
101, 101, 102, 122, 121, 201, 202, 222, 221
102, 102, 103, 123, 122, 202, 203, 223, 222
103, 103, 104, 124, 123, 203, 204, 224, 223
!ELEMENT, TYPE=361, MATITEM= 2
501, 501, 502, 522, 521, 601, 602, 622, 621, 1.00, 2.00
502, 502, 503, 523, 522, 602, 603, 623, 622, 1.20, 2.50
```

!END

メッシュデータの終端

このヘッダが表れると、メッシュデータの読み込みを終了する.

1 行目

!END

パラメータ

なし

2 行目以降

なし

!EQUATION

拘束節点グループの定義

1 行目

!EQUATION [, optional parameter]

パラメータ	
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

（2 行目 ） NEQ

（3 行目以降） nod1, DOF1, A1, nod2, DOF2, A2 ...（一行に四項まで）

以下繰り返し

変数名	属 性	内 容
NEQ	I	方程式の項数
nod1	I/C	第 1 節点または節点グループ
DOF1	I	第 1 節点または節点グループの拘束自由度
A1	R	第 1 節点または節点グループの係数
nod2	I/C	第 2 節点または節点グループ
DOF2	I	第 2 節点または節点グループの拘束自由度
A2	R	第 2 節点または節点グループの係数

注意

- 「!NODE」で定義されていない節点，節点グループが指定された場合は無視され，警告メッセージが表示される．
- 第一項に指定された節点または節点グループの自由度は従属自由度となる。第二項以降には独立自由度を指定する必要があるため、第一項に指定した自由度を他の「!EQUATION」で第二項以降に指定した場合はエラーとなる。
- 「nod1=nod2」の場合は無視され，警告メッセージが表示される．
- 節点グループを指定した場合，節点数の整合性が取れない場合はエラーとなる．
- 自由度番号は解析のタイプ，要素によって異なる．整合がとれない自由度については無視され，警告メッセージが表示される．

使用例

```
!EQUATION
3
101, 1, 1.0, 102, 1, -1.0, 103, 1, -1.0
2
NG1, 2, 1.0, NG5, 2, -1.0
5
2061, 1, 1.0, 1011, 1, -0.25, 1111, 1, -0.25, 3011, 1, -0.25
3111, 1, -0.25
```

!HEADER

メッシュファイルのタイトル

1 行目

!HEADER

パラメータ

なし

2 行目以降

(2 行目) TITLE

変数名	属性	内容
TITLE	C	ヘッダータイトル

使用例

```
!HEADER
Mesh for CFD Analysis
```

注意

- 省略可能.
- ヘッダーは複数行にわたってもよいが、ヘッダーとして認識されるのは最初の行の 127 カラム目までである.
- 「!HEADER」を複数回定義すると、内容が更新され、警告メッセージが表示される.

!INITIAL CONDITION

初期条件の定義

1 行目

!INITIAL CONDITION, TYPE=<type> [, optional parameter]

パラメータ	
TYPE	タイプ（必須）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
TYPE	TEMPERATURE	温度
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

（2 行目以降）nod1, VAL1（一行に一組）

以下繰り返し

変数名	属 性	内 容
nod1	I/C	節点番号または節点グループ
VAL1	R	節点値

注意

- 「!NODE」で定義されていない節点，節点グループが指定された場合は無視され，警告メッセージが表示される。

- 同じ節点に対して再定義した場合はエラーとなる.

使用例

```
!INITIAL CONDITION, TYPE=TEMPERATURE  
101, 25.0  
NA01, 38.0
```

!MATERIAL

材料物性の定義

物性が温度依存している場合は対応する温度ごとにテーブル入力が可能である

1 行目

!MATERIAL, NAME=<name> [, optional parameter]

パラメータ	
NAME	材料名（必須）
ITEM	物性項目数（省略可，省略した場合は「1」となる）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
NAME	<name>	材料名
ITEM	<ITEMnum>	ユーザー定義による物性項目数
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

(2 行目) !ITEM=1, SUBITEM=<k>

(3 行目) VAL1-1-1, VAL1-1-2, ... VAL1-1-k, TEMP1-1

(4 行目) VAL1-2-1, VAL1-2-2, ... VAL1-2-k, TEMP1-2

...

(L+2 行目) VAL1-L-1, VAL1-L-2, ... VAL1-L-k, TEMP1-L

以下「!ITEM=<ITEMnum>」まで繰り返し定義する

サブパラメータ（「!ITEM」に対するもの）	
SUBITEM	各物性項目で定義されるサブ物性項目数（省略可，省略した場合は「1」となる，異方性等を定義するのに使用）

サブパラメータ名	パラメータ値	内 容
SUBITEM	<subITEMnum>	ユーザー定義によるサブ物性項目数

【m 番目の物性が温度依存している場合】

温度依存のテーブルの項目数が N の場合，以下のように入力する：

!ITEM=m, SUBITEM=k

VALm1-1, ..., VALm1-k, TEMPm1

VALm2-1, ..., VALm2-k, TEMPm2

...

VALmN-1, ..., VALmN-k, TEMPm-N

変数名	属 性	内 容
VALmn-k	R	物性値（温度依存）
TEMPmn	R	対応する温度

- TEMPm1 < TEMPm2 < ... < TEMPmN でなければならない
- 温度が TEMPm1 以下の場合は VALm1, TEMPmN 以上の場合は VALmN が使用される。

【m 番目の物性が温度依存していない場合】

!ITEM=m, SUBITEM=k

VALm1-1, ..., VALm1-k

VALm2-1, ..., VALm2-k

...

VALmN-1, ..., VALmN-k

変数名	属 性	内 容
VALmn-k	R	物性値（温度依存なし）

注意

- 「!SECTION」オプションで参照されている MATERIAL が定義されていない場合はエラーとなる。
- 「!ELEMENT」オプションで、パラメータ「MATITEM」を使用して要素ごとに物性を入力した場合の値が優先して使用される。この場合、「!MATERIAL」オプションを使用して入力した物性値は使用されない。
- 「!ITEM=m」サブオプションの数と、パラメータ「ITEM」の数が整合しない場合、定義されていないサブオプションがある場合はエラーとなる。
- 「!ITEM=m」サブオプションは、m の小さい順番に並んでいなくてもよい。
- 「!SUBITEM=k」サブオプション、温度依存性を使用する場合、省略した値は「0.0」となる。
- 温度依存性を使用する場合、温度の低い順に定義しなければならない。
- 温度依存性を使用する場合、同じ温度を 2 回以上使用した場合はエラーとなる。
- 材料名が重複した場合はエラーとなる。

使用例

```
!MATERIAL, NAME= STEEL, ITEM= 2
!ITEM=1          温度依存性なし
35.0
!ITEM=2
40.0,    0.0
45.0, 100.0
50.0, 200.0
!MATERIAL, NAME= CUPPER 項目数=1（デフォルト値）
!ITEM=1          温度依存性なし
80.0
```


誤った使用例

例 1 【パラメータ「ITEM」 と「!ITEM=m」サブオプションの数が整合していない-1】

```
!MATERIAL, NAME= STEEL, ITEM= 2
!ITEM=3
20.0
!ITEM=1
35.0
!ITEM= 2
40.0
```

例 2 【パラメータ「ITEM」 と「!ITEM=m」サブオプションの数が整合していない-2】

```
!MATERIAL, NAME= STEEL, ITEM= 3
!ITEM=3
20.0
!ITEM= 2
40.0
!MATERIAL, NAME= CUPPER
...
```

参考

```
program TEST
use hecmw
implicit REAL*8 (A-H,O-Z)
type (hecmwT_local_mesh) :: hecMESH

!C
!C      !MATERIAL, NAME=SUS304, ITEM=3
!C      !ITEM=1, SUBITEM= 3
!C          100.0, 200.0, 300.0, 0.00
!C          101.0, 210.0, 301.0, 1.00
!C          102.0, 220.0, 302.0, 2.00
!C          103.0, 230.0, 303.0, 3.00
!C      !ITEM=3, SUBITEM= 2
!C          1000.0, , 0.00
!C          1001.0, 1., 1.00
!C          1002.0, 2., 2.00
!C          1003.0, 3., 3.00
!C      !ITEM=2
!C          5000.0
!C
!C      !MATERIAL, NAME=FEC, ITEM=2
!C      !ITEM=1, SUBITEM= 3
!C          2100.0, 2200.0, 2300.0, 0.00
!C          2101.0, 2210.0, 2301.0, 1.00
!C          2102.0, 2220.0, 2302.0, 2.00
!C          2103.0, 2230.0, 2303.0, 3.00
!C          3103.0, 3230.0, 2304.0, 4.00
!C      !ITEM=2
!C          6000.0, 10.0
!C          6500.0, 30.0
!C

hecMESH%material%n_mat      = 2

nn= hecMESH%material%n_mat
```

```

allocate (hecMESH%material%mat_name(nn))
  hecMESH%material%mat_name(1)= 'SUS304'
  hecMESH%material%mat_name(2)= 'FEC'

nn= hecMESH%material%n_mat
allocate (hecMESH%material%mat_ITEM_index(0:nn))
  hecMESH%material%mat_ITEM_index(0)= 0
  hecMESH%material%mat_ITEM_index(1)= 3
  hecMESH%material%mat_ITEM_index(2)= hecMESH%material%mat_ITEM_index(1) + 2

  hecMESH%material%n_mat_ITEM=
hecMESH%material%mat_ITEM_index(hecMESH%material%n_mat)

nn= hecMESH%material%n_mat_ITEM
allocate (hecMESH%material%mat_subITEM_index(0:nn))
  hecMESH%material%mat_subITEM_index(0)= 0
  hecMESH%material%mat_subITEM_index(1)= 3
  hecMESH%material%mat_subITEM_index(2)= hecMESH%material%mat_subITEM_index(1) +
1
  hecMESH%material%mat_subITEM_index(3)= hecMESH%material%mat_subITEM_index(2) +
2
  hecMESH%material%mat_subITEM_index(4)= hecMESH%material%mat_subITEM_index(3) +
3
  hecMESH%material%mat_subITEM_index(5)= hecMESH%material%mat_subITEM_index(4) +
1

  hecMESH%material%n_mat_subITEM=
&      hecMESH%material%mat_subITEM_index(hecMESH%material%n_mat_ITEM)

nn= hecMESH%material%n_mat_subITEM
allocate (hecMESH%material%mat_TABLE_index(0:nn))
  hecMESH%material%mat_TABLE_index( 0)= 0
  hecMESH%material%mat_TABLE_index( 1)= 4
  hecMESH%material%mat_TABLE_index( 2)= hecMESH%material%mat_TABLE_index( 1) + 4
  hecMESH%material%mat_TABLE_index( 3)= hecMESH%material%mat_TABLE_index( 2) + 4
  hecMESH%material%mat_TABLE_index( 4)= hecMESH%material%mat_TABLE_index( 3) + 1
  hecMESH%material%mat_TABLE_index( 5)= hecMESH%material%mat_TABLE_index( 4) + 4
  hecMESH%material%mat_TABLE_index( 6)= hecMESH%material%mat_TABLE_index( 5) + 4
  hecMESH%material%mat_TABLE_index( 7)= hecMESH%material%mat_TABLE_index( 6) + 5
  hecMESH%material%mat_TABLE_index( 8)= hecMESH%material%mat_TABLE_index( 7) + 5
  hecMESH%material%mat_TABLE_index( 9)= hecMESH%material%mat_TABLE_index( 8) + 5
  hecMESH%material%mat_TABLE_index(10)= hecMESH%material%mat_TABLE_index( 9) + 2

  hecMESH%material%n_mat_TABLE=
&
hecMESH%material%mat_TABLE_index(hecMESH%material%n_mat_subITEM)

nn= hecMESH%material%n_mat_TABLE
allocate (hecMESH%material%mat_VAL (nn))
allocate (hecMESH%material%mat_TEMP(nn))

  hecMESH%material%mat_VAL = 0. d0
  hecMESH%material%mat_TEMP= 0. d0

  hecMESH%material%mat_VAL ( 1)= 100. 0d0
  hecMESH%material%mat_TEMP( 1)= 0. 0d0
  hecMESH%material%mat_VAL ( 2)= 101. 0d0
  hecMESH%material%mat_TEMP( 2)= 1. 0d0
  hecMESH%material%mat_VAL ( 3)= 102. 0d0
  hecMESH%material%mat_TEMP( 3)= 2. 0d0
  hecMESH%material%mat_VAL ( 4)= 103. 0d0
  hecMESH%material%mat_TEMP( 4)= 3. 0d0

  hecMESH%material%mat_VAL ( 5)= 200. 0d0
  hecMESH%material%mat_TEMP( 5)= 0. 0d0

  hecMESH%material%mat_VAL (13)= 5000. 0d0

  hecMESH%material%mat_VAL (14)= 1000. 0d0
  hecMESH%material%mat_TEMP(14)= 0. 0d0
  hecMESH%material%mat_VAL (15)= 1001. 0d0
  hecMESH%material%mat_TEMP(15)= 1. 0d0

```

```

hecMESH%material%mat_VAL (16)= 1002.0d0
hecMESH%material%mat_TEMP (16)= 2.0d0
hecMESH%material%mat_VAL (17)= 1003.0d0
hecMESH%material%mat_TEMP (17)= 3.0d0

hecMESH%material%mat_VAL (18)= 0.0d0
hecMESH%material%mat_TEMP (18)= 0.0d0
hecMESH%material%mat_VAL (19)= 1.0d0
hecMESH%material%mat_TEMP (19)= 1.0d0
hecMESH%material%mat_VAL (20)= 2.0d0
hecMESH%material%mat_TEMP (20)= 2.0d0
hecMESH%material%mat_VAL (21)= 3.0d0
hecMESH%material%mat_TEMP (21)= 3.0d0

hecMESH%material%mat_VAL (22)= 2100.0d0
hecMESH%material%mat_TEMP (22)= 0.0d0
hecMESH%material%mat_VAL (23)= 2101.0d0
hecMESH%material%mat_TEMP (23)= 1.0d0
hecMESH%material%mat_VAL (24)= 2102.0d0
hecMESH%material%mat_TEMP (24)= 2.0d0
hecMESH%material%mat_VAL (25)= 2103.0d0
hecMESH%material%mat_TEMP (25)= 3.0d0
hecMESH%material%mat_VAL (26)= 3103.0d0
hecMESH%material%mat_TEMP (26)= 4.0d0

write (*, '(a,i10)') '%n_mat_ITEM ', hecMESH%material%n_mat_ITEM
write (*, '(a,i10)') '%n_mat_subITEM', hecMESH%material%n_mat_subITEM
write (*, '(a,i10)') '%n_mat_TABLE ', hecMESH%material%n_mat_TABLE

end program TEST

```

!NGROUP

節点グループの定義

1 行目

!NGROUP, NGRP=<ngrp> [, optional parameter]

パラメータ	
NGRP	節点グループ名（必須）
GENERATE	節点グループに属する節点の自動生成（省略可）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
NGRP	<ngrp>	節点グループ名
GENERATE	なし	節点グループに属する節点の自動生成
INPUT	<filename>	外部ファイル名（省略可）, 2 行目以降との併用も可能

2 行目以降（GENERATE を使用しない場合）

（2 行目） nod1, nod2, nod3

（以下同様）

変数名	属 性	内 容
nodX	I	節点グループに属する節点番号

2 行目以降（GENERATE を使用する場合）

（2 行目） nod1, nod2, nod3

(以下同様)

変数名	属 性	内 容
nod1	I	節点グループ内の最初の節点番号
nod2	I	節点グループ内の最後の節点番号
nod3	I	節点番号増分（省略可能、省略時は nod3=1 となる）

注意

- 1 行に任意の数の節点を入れることができる。また次のオプションが始まるまで、任意の数の行を挿入することができる。
- 指定する節点は「!NGROUP」より前に定義されている必要がある。
- 「!NODE」オプションで定義されていない節点は除外され、警告メッセージが表示される。
- 指定された節点が既に同じグループ内に存在する場合は無視され、警告メッセージが表示される。
- 全ての節点は、「ALL」という名前の節点グループに属している（自動的に生成される）。
- 一つのグループを複数回にわけて定義できる。

使用例

!NGROUP, NGRP= NA01	
1, 2, 3, 4, 5, 6	
101, 102	
!NGROUP, NGRP= NA02	
101, 102	
!NGROUP, NGRP= NA01	グループ「NA01」に「501, 505」が追加される。
501, 505	
!NGROUP, NGRP= NA02	グループ「NA02」に「501, 505」が追加される。
501, 505	
!NGROUP, NGRP= NA04, GENERATE	グループ「NA04」に
301, 309, 2	「301, 303, 305, 307, 309, 311, 312, 313」が追加される。
311, 313	

!NODE

節点座標の定義

1 行目

!NODE [, optional parameter]

パラメータ	
SYSTEM	座標系（省略可）
NGRP	節点グループ（省略可）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内容
SYSTEM	R	デカルト座標系（デフォルト値）
	C	円筒座標系
NGRP	<ngrp>	節点グループ（省略可）
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

（2 行目）NODE_ID, Xcoord, Ycoord, Zcoord 省略された場合は「0.0」が入る
以下同様

変数名	属 性	内容
NODE_ID	I	節点番号
Xcoord	R	X 座標
Ycoord	R	Y 座標
Zcoord	R	Z 座標

注意

- 節点座標は区切り記号を含めて全て省略可能である.
- 既に定義されている節点を再定義した場合, 内容が更新され, 警告メッセージが表示される.
- 「!ELEMENT」で参照されない節点は除外される.

使用例

```
!NODE, NGRP= TEST
1, 0.0, 0.0, 0.0
2, 0.0, 0.1, 0.0
3, 0.0,, 1.5      Y座標は「0.0」
4,                X,Y,Z座標は「0.0」
```

!SECTION

セクションの定義

1 行目

!SECTION, TYPE=<type>, EGRP=<egrp> [, optional parameter]

パラメータ	
TYPE	セクションタイプ (必須)
EGRP	要素グループ (必須)
MATERIAL	ユーザー定義材料名 (省略可, 省略した場合, 材料名は「ALL」となるが, 「ALL」という材料についてはユーザーが定義する必要がある)
COMPOSITE	複数の材料から構成されている場合 (MATERIAL とは排他, 省略可, 省略した場合は=1 となる) 現版では使用不可.
SECOPT	要素タイプ用補助パラメータ (省略可, 省略した場合は=0 となる)
INPUT	外部ファイル名 (省略可)

パラメータ名	パラメータ値	内容
TYPE	SOLID	ロッド, 三角形, 四角形, 四面体, 三角柱, 六面体要素
	SHELL	シェル要素
	INTERFACE	インタフェース要素
EGRP	<egrp>	要素グループ (必須)
MATERIAL	<material >	ユーザー定義による材料名
COMPOSITE	<n_comp>	複合材料数
SECOPT	<secopt>	= 0 : 指定なし, 平面応力
		= 1 : 平面歪
		= 2 : 軸対称
		=10 : 0+次数低減積分
		=11 : 1+次数低減積分
		=12 : 2+次数低減積分

INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能
-------	------------	----------------------------

2 行目以降

【TYPE=SOLID】の場合

（2 行目）THICKNESS

変数名	属 性	内 容
THICKNESS	R	要素厚さ，断面積

- 「TYPE=SOLID」の場合，「THICKNESS」は省略可，デフォルト値（1.0）が入る．
- THICKNESS は 0.0 より大きい値でなければならない．

【TYPE=SHELL】の場合

（2 行目）THICKNESS, INTEGPOINTS

変数名	属 性	内 容
THICKNESS	R	シェル断面厚さ
INTEGPOINTS	I	シェル断面方向積分点

- THICKNESS は 0.0 より大きい値，INTEGPOINTS は自然数でなければならない．

【TYPE=INTERFACE】の場合

（2 行目）THICKNESS, GAPCON, GAPRAD1, GAPRAD2

変数名	属 性	内 容
THICKNESS	R	断面厚さ
GAPCON	R	ギャップ熱伝達係数（省略時 0）
GAPRAD1	R	ギャップ輻射熱伝達係数-1（省略時 0）
GAPRAD2	R	ギャップ輻射熱伝達係数-2（省略時 0）

- THICKNESS は 0.0 より大きい値でなければならない。

注意

- パラメータ「TYPE」が要素タイプと整合していない場合はエラーとなる。
- 「!SECTION, EGRP=ALL」が必要となる場合はユーザーが定義しなければならない。
- SECTION 情報を持たない要素がある場合はエラーとなる。
- セクション名が重複した場合はエラーとなる。

使用例

```
!SECTION, EGRP=SHELL1, TYPE=SOLID, MATERIAL=STEEL
2.5
!SECTION, EGRP=SHELL2, TYPE=SOLID, MATERIAL=STEEL 厚さ1.0
!SECTION, EGRP=BEAM1, TYPE=BEAM, MATERIAL=CUPPER
3.0, 5.0
```

!SGROUP

面グループの定義

1 行目

!SGROUP, SGRP=<grp> [, optional parameter]

パラメータ	
SGRP	面グループ名（必須）
INPUT	外部ファイル名（省略可）

パラメータ名	パラメータ値	内 容
SGRP	<grp>	面グループ名
INPUT	<filename>	外部ファイル名（省略可），2 行目以降との併用も可能

2 行目以降

（2 行目） elem1, lsuf1, elem2, lsuf2, elem3, lsuf3, ...

（以下同様）

変数名	属 性	内 容
elemX	I	面グループに属する要素番号
lsufX	I	面グループに属する要素の局所面番号

注意

- 要素タイプと面番号については、「付録 HEC-MW 要素ライブラリ」を参照のこと。
- （要素，局所面番号）という組み合わせによって面を構成する。1 行に任意の数の面

を入れることができる。また次のオプションが始まるまで、任意の数の行を挿入することができる。（要素，局所面番号）という組み合わせは必ず同一の行になければならない。

- 指定する要素は「!SGROUP」より前に定義されている必要がある。
- 要素が「!ELEMENT」オプションで定義されていない場合は無視され，警告メッセージが表示される。
- 「!ELEMENT」オプションで定義されていない要素を含む面は除外され，警告メッセージが表示される。
- 要素タイプと面番号の整合性が取れない面は除外され，警告メッセージが表示される。
- 一つのグループを複数回にわけて定義できる。

使用例

```
!SGROUP, SGRP= SUF01
101, 1, 102, 1, 103, 2, 104, 2
201, 1, 202, 1
501, 1
!SGROUP, SGRP= SUF02
101, 2, 102, 2
!SGROUP, SGRP= EA01
601, 1
602, 2
```

グループ「EA01」に「(601, 1), (602, 2)」が追加。

誤った使用例

例 1 【（要素，局所面番号）の組が複数行にわたっている】

```
!SGROUP, SGRP= SUF01
101, 1, 102, 1, 103
1, 104, 1
```

例 2 【局所面番号と要素タイプの整合性がとれない】

```
!ELEMENT, TYPE= 231, SECTION= A
101, 1, 2, 3
102, 2, 3, 4
...
!SGROUP, SGRP= SUF01
101, 1
101, 2
101, 4
```

三角形要素に第4面は存在しないので，この組み合わせは無視される

!ZERO

絶対零度

1 行目

!ZERO

パラメータ

なし

2 行目以降

(2 行目) ZERO

変数名	属 性	内 容
ZERO	R	絶対零度

注意

- 省略可能. 省略された場合は「絶対零度=0」となる.
- 「!ZERO」を複数回定義すると, 内容が更新され, 警告メッセージが表示される.

使用例

!ZERO -273.16

4.4. 特記事項

(1) !ELEMENT における材料物性値

!ELEMENT で要素ごとに材料物性を指定した場合、それを表す材料情報が自動生成され、!SECTION で与えられた材料物性を上書きする。

例えば、以下の!ELEMENT が定義された場合、

```
!ELEMENT, TYPE=361, MATITEM=2  
501, 501, 502, 522, 521, 601, 602, 622, 621, 1.00, 2.00
```

このデータから生成される材料物性の定義情報は、次の!MATERIAL によって生成されるそれと等しい。

```
!MATERIAL, NAME=HECMW-MAT501  
!ITEM=1, SUBITEM=2  
1.00, 2.00
```

材料名は、HECMW-MAT[global element ID]となる。

(2) !EQUATION に関する処理

!EQUATION の入力があった場合、その情報からリンク要素を自動生成する。

生成されるリンク要素は、!EQUATION の第一項と、第二～N 項との組み合わせで、それぞれの項で指定されている節点をコネクティビティとする。リンク要素の要素番号は 9XY であり、X は第一項の拘束自由度、Y はもう一つの項の拘束自由度である。

分散データ構造体において、リンク要素の hecmwST_local_mesh%elem_mat_ID と hecmwST_local_mesh%section_ID はそれぞれ本来の意味を持たない。

elem_mat_ID には、リンク要素を作成した際に、!EQUATION の第一項とリンクした項の番号が入り、section_ID には、拘束方程式の通し番号が入る。

5. 分散メッシュデータ構造体の概要

分散メッシュデータ関連構造体は以下の各部分から構成されている. (*) をつけたものは実際に分散メッシュデータから読み込まれる情報を含んでいる :

(1) 制御情報 (*)

内 容 : 全体的な制御に関する情報

(2) 分散メッシュ情報 (*)

内 容 : 節点, 要素, 通信, 階層構造

構造体 : `hecmwST_local_mesh`

(3) セクション情報 (*)

内 容 : 各要素の属するセクションの情報

構造体 : `hecmwST_section`

(4) 材料物性情報 (*)

内 容 : 各材料の物性情報, 温度依存性テーブルも含む

構造体 : `hecmwST_material`

(5) 拘束グループ情報 (*)

内 容 : 拘束グループに関する情報

構造体 : `hecmwST_mpc`

(6) グループ情報 (*)

内 容 : 節点, 要素, 面グループに関する情報

構造体 : `hecmwST_node_grp`, `hecmwST_elem_grp`, `hecmwST_surf_grp`

5.1. 一般パラメータ設定

```
module hecmw_util
  implicit none
  include 'mpif.h'
  public

  integer(kind=4),parameter:: kint = 4
  integer(kind=4),parameter:: kreal = 8

  integer(kind=kint),parameter :: HECMW_NAME_LEN      = 63
  integer(kind=kint),parameter :: HECMW_HEADER_LEN    = 127
  integer(kind=kint),parameter :: HECMW_MSG_LEN       = 255
  integer(kind=kint),parameter :: HECMW_FILENAME_LEN  = 1023

  integer(kind=kint),parameter :: hecmw_sum           = 46801
  integer(kind=kint),parameter :: hecmw_prod         = 46802
  integer(kind=kint),parameter :: hecmw_max          = 46803
  integer(kind=kint),parameter :: hecmw_min          = 46804
  integer(kind=kint),parameter :: hecmw_integer       = 53951
  integer(kind=kint),parameter :: hecmw_single_precision = 53952
  integer(kind=kint),parameter :: hecmw_double_precision = 53953
  integer(kind=kint),parameter :: hecmw_character    = 53954
```

変数名	内容
kint	INT 変数のバイト長
kreal	REAL 変数のバイト長
HECMW_NAME_LEN	名前の最大長
HECMW_HEADER_LEN	HEADER の最大長
HECMW_MSG_LEN	ログメッセージの最大長
HECMW_FILENAME_LEN	ファイル名の最大長

5.2. 分散メッシュ情報

分散メッシュ情報について，Fortran 版を用いて説明する．C 版においても構造や変数名は同じである¹が，配列の添字が 0 から始まることに注意しなければならない．

```
type hecmwST_local_mesh
```

(1) 全体情報

```
character(HECMW_FILENAME_LEN)      :: gridfile
character(HECMW_FILENAME_LEN), pointer :: files(:)
character(HECMW_HEADER_LEN)         :: header
integer(kind=kint) :: hecmw_flag_adapt
integer(kind=kint) :: hecmw_flag_initcon
integer(kind=kint) :: hecmw_n_file
integer(kind=kint) :: hecmw_flag_parttype
integer(kind=kint) :: hecmw_flag_partdepth
integer(kind=kint) :: hecmw_flag_version
real(kind=kreal)   :: zero_temp
```

変数名	配列要素数	内容
gridfile		グリッドファイル名
hecmw_n_file		もろもろのファイル数
files	(hecmw_n_file)	もろもろのファイル
header		ヘッダ
hecmw_flag_adapt		階層構造使用の有無 (=0 : NO, =1 : YES)
hecmw_flag_initcon		初期条件使用 (=0 : NO, =1 : YES)
hecmw_flag_parttype		領域分割形式 (=1 : Node-based, =2 : Element-based)
hecmw_flag_partdepth		部分領域間袖領域のオーバーラップ深さ
hecmw_flag_version		バージョン
zero_temp		絶対零度 (温度)

¹ 一部例外あり．hecmwST_local_mesh の MPI_COMM は，C 版では HECMW_COMM となる．

(2) 節点情報

```

integer(kind=kint)      :: n_node
integer(kind=kint)      :: nn_internal
integer(kind=kint)      :: n_dof
integer(kind=kint)      :: n_dof_grp
real(kind=kreal), pointer :: node(:)
integer(kind=kint), pointer :: node_ID(:)
integer(kind=kint), pointer :: global_node_ID(:)
integer(kind=kint), pointer :: node_val_index(:)
real(kind=kreal), pointer :: node_val_item(:)
integer(kind=kint), pointer :: node_dof_index(:)
integer(kind=kint), pointer :: node_dof_item(:)
integer(kind=kint), pointer :: node_init_val_index(:)
real(kind=kreal), pointer :: node_init_val_item(:)
integer(kind=kint), pointer :: node_internal_list(:)

```

変数名	配列要素数	内容
n_node		総節点数
nn_internal		内部節点数
n_dof		節点最大自由度数
n_dof_grp		自由度グループ数
node	3*n_node	節点座標 node(3*k-2) : 節点 k の X 座標 node(3*k-1) : 節点 k の Y 座標 node(3*k) : 節点 k の Z 座標
node_ID	2*n_node	節点 ID (ローカル番号, 所属領域) node_id(2*i-1) : 節点 i のローカル番号 node_id(2*i) : 節点 i の所属領域
global_node_ID	n_node	グローバル節点 ID
node_val_index	0:n_node	節点物理量インデックス
node_val_item	node_val_index(n_node)	節点物理量
node_dof_index	0:n_dof_grp	自由度用インデックス
node_dof_item	n_dof_grp	自由度
node_init_val_index	0: n_node	初期条件用インデックス
node_init_val_item	node_init_val_index(n_node)	初期条件
node_internal_list	nn_internal	内部節点リスト

(3) 要素情報

```

integer(kind=kint)      :: n_elem
integer(kind=kint)      :: ne_internal
integer(kind=kint)      :: n_elem_type
integer(kind=kint)      :: n_elem_mat_ID
integer(kind=kint), pointer :: elem_type_index(:)
integer(kind=kint), pointer :: elem_type_item(:)
integer(kind=kint), pointer :: elem_type(:)
integer(kind=kint), pointer :: section_ID(:)
integer(kind=kint), pointer :: elem_mat_ID_index(:)
integer(kind=kint), pointer :: elem_mat_ID_item(:)
integer(kind=kint), pointer :: elem_node_index(:)
integer(kind=kint), pointer :: elem_node_item(:)
integer(kind=kint), pointer :: elem_ID(:)
integer(kind=kint), pointer :: global_elem_ID(:)
integer(kind=kint), pointer :: elem_internal_list(:)
integer(kind=kint), pointer :: elem_mat_int_index(:)
real(kind=kreal), pointer :: elem_mat_int_val(:)
integer(kind=kint), pointer :: elem_val_index(:)
real(kind=kreal), pointer :: elem_val_item(:)

```

変数名	配列要素数	内容
n_elem		総要素数
ne_internal		内部要素数
n_elem_type		要素タイプの種類
n_elem_mat_ID		= elem_mat_ID_index(n_elem)
elem_type_index	0:n_elem_type	要素タイプのインデックス
elem_type_item	n_elem_type	要素タイプの内容
elem_type	n_elem	要素タイプ
section_ID	n_elem	セクション番号
elem_mat_ID_index	0:n_elem	材料物性 ID のための一次元インデックス (COMPOSITE 用)
elem_mat_ID_item	elem_mat_ID_index(n_elem)	材料物性 ID 配列 (COMPOSITE 用)
elem_node_index	0:n_elem	要素コネクティビティ用一次元インデックス
elem_node_item	elem_node_index(n_elem)	要素コネクティビティ用配列
elem_ID	2*n_elem	要素 ID (ローカル番号, 所属領域)
global_elem_ID	n_elem	グローバル要素 ID
elem_internal_list	ne_internal	内部要素リスト
elem_mat_int_index	0: n_elem	材料物性用一次元インデックス (内部処理用)
elem_mat_int_val	elem_mat_int_index(n_elem)	材料物性用配列 (内部処理用)

elem_val_index	0:n_elem	要素物理量インデックス
elem_val_item	elem_val_index(n_elem)	要素物理量

(4) PE および通信情報

```

integer(kind=kint)      :: zero
integer(kind=kint)      :: MPI_COMM
integer(kind=kint)      :: PETOT
integer(kind=kint)      :: PEsmptTOT
integer(kind=kint)      :: my_rank
integer(kind=kint)      :: errnof
integer(kind=kint)      :: n_subdomain
integer(kind=kint)      :: n_neighbor_pe
integer(kind=kint), pointer :: neighbor_pe(:)
integer(kind=kint), pointer :: import_index(:)
integer(kind=kint), pointer :: import_item(:)
integer(kind=kint), pointer :: export_index(:)
integer(kind=kint), pointer :: export_item(:)
integer(kind=kint), pointer :: shared_index(:)
integer(kind=kint), pointer :: shared_item(:)

```

変数名	配列要素数	内容
MPI_COMM		MPI 用コミュニケーター
zero		領域番号=0 であれば「1」、それ以外は「0」
PETOT		総領域数
PEsmptTOT		SMP ノードあたりの PE 数
my_rank		プロセス番号
errnof		エラーID (FORTRAN でのみ使用)
n_subdomain		総領域数 (局所分散データからの読み込み)
n_neighbor_pe		隣接領域数
neighbor_pe	n_neighbor_pe	隣接領域 ID
import_index	0: n_neighbor_pe	受信テーブル用一次元インデックス
import_item	import_index(n_neighbor_pe)	受信テーブル用配列
export_index	0: n_neighbor_pe	送信テーブル用一次元インデックス
export_item	export_index(n_neighbor_pe)	送信テーブル用配列
shared_index	0: n_neighbor_pe	送受信テーブル用一次元インデックス
shared_item	shared_index(n_neighbor_pe)	送受信テーブル用配列

(5) 階層構造

```
integer(kind=kint)      :: coarse_grid_level
integer(kind=kint)      :: n_adapt
integer(kind=kint), pointer :: when_i_was_refined_node(:)
integer(kind=kint), pointer :: when_i_was_refined_elem(:)
integer(kind=kint), pointer :: adapt_parent_type(:)
integer(kind=kint), pointer :: adapt_type(:)
integer(kind=kint), pointer :: adapt_level(:)
integer(kind=kint), pointer :: adapt_parent(:)
integer(kind=kint), pointer :: adapt_children_index(:)
integer(kind=kint), pointer :: adapt_children_item(:)
```

変数名	配列要素数	内容
coarse_grid_level		最も粗いメッシュのレベル
n_adapt		Refinement された回数
when_i_was_refined_node	n_node	各節点の生成された Refinement レベル
when_i_was_refined_elem	n_elem	各要素の生成された Refinement レベル
adapt_parent_type	n_elem	各要素の親要素の分割タイプ
adapt_type	n_elem	各要素の分割タイプ
adapt_level	n_elem	各要素の分割レベル
adapt_parent	2*n_elem	各要素の親要素 ID (ローカル番号, 所属領域)
adapt_children_index	0:n_elem	各要素の子要素用一次元インデックス
adapt_children_item	2*adapt_children_index(n_elem)	各要素の子要素 ID (ローカル番号, 所属領域)

(6) 下部構造

```
type (hecmwST_section)  :: section
type (hecmwST_material) :: material
type (hecmwST_mpc)      :: mpc
type (hecmwST_amplitude) :: amp
type (hecmwST_node_grp)  :: node_group
type (hecmwST_elem_grp)  :: elem_group
type (hecmwST_surf_grp)  :: surf_group
```

```
end type hecmwST_local_mesh
```

5.3. セクション情報

```

type hecmwST_section
  integer(kind=kint)          :: n_sect
  integer(kind=kint), pointer :: sect_type(:)
  integer(kind=kint), pointer :: sect_opt(:)
  integer(kind=kint), pointer :: sect_mat_ID_index(:)
  integer(kind=kint), pointer :: sect_mat_ID_item(:)
  integer(kind=kint), pointer :: sect_I_index(:)
  integer(kind=kint), pointer :: sect_I_item(:)
  integer(kind=kint), pointer :: sect_R_index(:)
  real(kind=kreal), pointer  :: sect_R_item(:)
end type hecmwST_section

```

変数名	配列要素数	内容
n_sect		セクション数
sect_type	n_sect	セクションタイプ (SOLID : 1, SHELL : 2, BEAM : 3, INTERFACE:4)
sect_opt	n_sect	SECOPT
sect_mat_ID_index	0:n_sect	材料物性 ID のための一次元インデックス (COMPOSITE 用)
sect_mat_ID_item	sect_mat_ID_index(n_sect)	材料物性 ID
sect_I_index	0:n_sect	セクション値用一次元インデックス (整数)
sect_I_item	sect_I_index(n_sect)	セクション値配列 (整数)
sect_R_index	0:n_sect	セクション値用一次元インデックス (実数)
sect_R_item	sect_R_index(n_sect)	セクション値配列 (実数)

5.4. 材料物性情報

```

type hecmwST_material
  integer(kind=kint)      :: n_mat
  integer(kind=kint)      :: n_mat_item
  integer(kind=kint)      :: n_mat_subitem
  integer(kind=kint)      :: n_mat_table
  character(HECMW_NAME_LEN), pointer :: mat_name(:)
  integer(kind=kint), pointer :: mat_item_index(:)
  integer(kind=kint), pointer :: mat_subitem_index(:)
  integer(kind=kint), pointer :: mat_table_index(:)
  real(kind=kreal), pointer :: mat_val(:)
  real(kind=kreal), pointer :: mat_temp(:)
end type hecmwST_material

```

変数名	配列要素数	内容
n_mat		材料数
n_mat_item		材料物性総数
n_mat_subitem		材料物性サブ項目総数
n_mat_table		材料物性テーブル総数
mat_name	n_mat	材料物性名
mat_item_index	0:n_mat	材料数一次元インデックス
mat_subitem_index	0:n_mat_item	材料物性数一次元インデックス
mat_table_index	0:n_mat_subitem	材料物性テーブル一次元インデックス
mat_val	mat_table_index(n_mat_subitem)	材料物性
mat_temp	mat_table_index(n_mat_subitem)	温度依存テーブル

5.5. 拘束グループ情報

```

type hecmwST_mpc
  integer(kind=kint)      :: n_mpc
  integer(kind=kint),pointer :: mpc_index(:)
  integer(kind=kint),pointer :: mpc_item(:)
  integer(kind=kint),pointer :: mpc_dof(:)
  real(kind=kreal),pointer  :: mpc_val(:)
end type hecmwST_mpc

```

変数名	配列要素数	内容
n_mpc		拘束グループ数
mpc_index	0:n_mpc	拘束グループ用一次元インデックス
mpc_item	mpc_index(n_mpc)	拘束グループ節点番号
mpc_dof	mpc_index(n_mpc)	拘束自由度情報
mpc_val	mpc_index(n_mpc)	拘束自由度係数

5.6. AMPLITUDE 情報

```

type hecmwST_amplitude
  integer(kind=kint)      :: n_amp
  character(len=HECMW_NAME_LEN),pointer :: amp_name(:)
  integer(kind=kint),pointer :: amp_type_definition(:)
  integer(kind=kint),pointer :: amp_type_time(:)
  integer(kind=kint),pointer :: amp_type_value(:)
  integer(kind=kint),pointer :: amp_index(:)
  real(kind=kreal),pointer  :: amp_val(:)
  real(kind=kreal),pointer  :: amp_table(:)
end type hecmwST_amplitude

```

変数名	配列要素数	内容
n_amp		AMPLITUDE グループ数
amp_name	n_amp	AMPLITUDE グループ名
amp_type_definition	n_amp	AMPLITUDE タイプ (TABLAR : 1)
amp_type_time	n_amp	AMPLITUDE タイプ (STEP TIME : 1)
amp_type_value	n_amp	AMPLITUDE タイプ (RELATIVE : 1, ABSOLUTE : 2)
amp_index	0:n_amp	AMPLITUDE インデックス
amp_val	amp_index(n_amp)	AMPLITUDE 値
amp_table	amp_index(n_amp)	AMPLITUDE 時間

5.7. グループ情報（節点）

```

type hecmwST_node_grp
  integer(kind=kint)      :: n_grp
  integer(kind=kint)      :: n_bc
  character(HECMW_NAME_LEN), pointer :: grp_name(:)
  integer(kind=kint), pointer :: grp_index(:)
  integer(kind=kint), pointer :: grp_item(:)
  integer(kind=kint), pointer :: bc_grp_ID(:)
  integer(kind=kint), pointer :: bc_grp_type(:)
  integer(kind=kint), pointer :: bc_grp_index(:)
  integer(kind=kint), pointer :: bc_grp_dof(:)
  real(kind=kreal), pointer :: bc_grp_val(:)
end type hecmwST_node_grp

```

変数名	配列要素数	内容
n_grp		グループ数
grp_name	n_grp	グループ名
grp_index	0:n_grp	グループ要素用一次元インデックス
grp_item	grp_index(n_grp)	グループ要素用配列
n_bc		境界条件設定節点数（自由度数）
bc_grp_ID	n_bc	所属節点グループ番号
bc_grp_type	n_bc	境界条件タイプ, =1 : 変位, =2 : Flux 負の場合はユーザーサブルーチン
bc_grp_index	n_bc	境界条件設定節点番号
bc_grp_dof	n_bc	境界条件設定自由度
bc_grp_val	n_bc	境界条件値

5.8. グループ情報（要素）

```

type hecmwST_elem_grp
  integer(kind=kint)      :: n_grp
  integer(kind=kint)      :: n_bc
  character(HECMW_NAME_LEN), pointer :: grp_name(:)
  integer(kind=kint), pointer :: grp_index(:)
  integer(kind=kint), pointer :: grp_item(:)
  integer(kind=kint), pointer :: bc_grp_ID(:)
  integer(kind=kint), pointer :: bc_grp_type(:)
  integer(kind=kint), pointer :: bc_grp_index(:)
  real(kind=kreal), pointer :: bc_grp_val(:)
end type hecmwST_elem_grp

```

変数名	配列要素数	内容
n_grp		グループ数
grp_name	n_grp	グループ名
grp_index	0:n_grp	グループ要素用一次元インデックス
grp_item	grp_index(n_grp)	グループ要素用配列
n_bc		境界条件設定要素数
bc_grp_ID	n_bc	所属要素グループ番号
bc_grp_type	n_bc	境界条件タイプ, =1 : 変位, =2 : Flux 負の場合はユーザーサブルーチン
bc_grp_index	n_bc	境界条件設定要素番号
bc_grp_val	n_bc	境界条件値

5.9. グループ情報（面）

```

type hecmwST_surf_grp
  integer(kind=kint)      :: n_grp
  integer(kind=kint)      :: n_bc
  character(HECMW_NAME_LEN), pointer :: grp_name(:)
  integer(kind=kint), pointer :: grp_index(:)
  integer(kind=kint), pointer :: grp_item(:)
  integer(kind=kint), pointer :: bc_grp_ID(:)
  integer(kind=kint), pointer :: bc_grp_type(:)
  integer(kind=kint), pointer :: bc_grp_index(:)
  real(kind=kreal), pointer :: bc_grp_val(:)
end type hecmwST_surf_grp

```

変数名	配列要素数	内容
n_grp		面グループ数
grp_name	n_grp	面グループ名
grp_index	0:n_grp	面グループ要素用一次元インデックス
grp_item	2*grp_index(n_grp)	面グループ要素用配列（要素，局所面番号） grp_item(2*k-1)：要素番号 grp_item(2*k)：局所面番号
n_bc		境界条件設定要素数（自由度数）
bc_grp_ID	n_bc	所属要素グループ番号
bc_grp_type	n_bc	境界条件タイプ，=1：変位，=2： Flux 負の場合はユーザーサブルーチン
bc_grp_index	2*n_bc	境界条件設定面番号（要素，局所面番号） bc_grp_index(2*k-1)：要素番号 bc_grp_index(2*k)：局所面番号
bc_grp_val	n_bc	境界条件値

```

end module hecmw_util

```

6. 結果データ

6.1. 結果データ構造体

```
type hecmwST_result_data
  integer(kind=kint) :: nn_component
  integer(kind=kint) :: ne_component
  integer(kind=kint), pointer :: nn_dof(:)
  integer(kind=kint), pointer :: ne_dof(:)
  character(len=HECMW_NAME_LEN), pointer :: node_label(:)
  character(len=HECMW_NAME_LEN), pointer :: elem_label(:)
  real(kind=kreal), pointer :: node_val_item(:)
  real(kind=kreal), pointer :: elem_val_item(:)
end type hecmwST_result_data
```

変数名	内容
nn_component	節点コンポーネント数
ne_component	要素コンポーネント数
nn_dof	節点自由度数
ne_dof	要素自由度数
node_label	節点ラベル
elem_label	要素ラベル
node_val_item	節点値 値の並びは、節点ごとにその節点の全コンポーネントが格納される
elem_val_item	要素値 値の並びは、要素ごとにその要素の全コンポーネントが格納される

6.2. 結果ファイルフォーマット

```
header
n_node n_elem
nn_component ne_component
nn_dof(1) nn_dof(2) ... nn_dof(nn_comonent)
node_label(1)
node_label(2)
...
node_label(nn_component)
node_global_ID(1)
node_val_item(1) node_val_item(2) ... node_val_item(sum(nn_dof))
node_global_ID(2)
...
ne_dof(1) ne_dof(2) ... ne_dof(ne_comonent)
elem_label(1)
elem_label(2)
...
elem_label(ne_component)
elem_global_ID(1)
elem_val_item(1) elem_val_item(2) ... elem_val_item(sum(ne_dof))
...
```

ただし,

header : hecmw_result_init API で指定した任意の文字列
n_node : 分散メッシュ構造体の節点数 n_node
n_elem : 分散メッシュ構造体の要素数 n_elem

である.

7. Fortran API リファレンス

hecmw_init

HEC-MW の初期化処理を行います。

```
subroutine hecmw_init()
```

引数

なし

説明

HEC-MW の初期化処理を行います。

これは、プログラムの開始直後に必ず呼び出さねばなりません。

MPI の初期化を行うため、この呼び出し以降、並列処理が可能となります。また、全体制御ファイルが自動的に読み込まれます。

hecmw_finalize

HEC-MW の終了処理を行います。

```
subroutine hecmw_finalize
```

引数

なし

説明

HEC-MW の終了処理を行います。

これは、プログラムの終了直前に呼び出さねばなりません。

MPI の終了処理はこの呼び出しで行われます。

hecmw_get_mesh

メッシュデータをファイルから読み込みます.

```
subroutine hecmw_get_mesh(name_ID, mesh)

character(len=HECMW_NAME_LEN) :: name_ID
type(hecmwST_local_mesh) :: mesh
```

引数

name_ID

!MESH または!MESH GROUP を特定する識別子

mesh

読み込まれたメッシュデータの格納先

説明

ファイルからメッシュデータを読み込みます.

この API は, 全体制御ファイルから入力ファイルの情報を取得します.

読み込み可能なメッシュファイルの種類は以下のとおりです.

- HEC-MW 分散メッシュデータ
- HEC-MW 単一領域メッシュデータ
- GeoFEM メッシュデータ

メッシュファイルの種類は全体制御ファイルで指定します.

読み込むメッシュファイルは, 全体制御ファイルの!MESH で定義されており, かつ NAME が name_ID のものです. !MESH GROUP によってメッシュファイルがグループ化されている場合は, グループ内の全てのメッシュファイルを読み込みます.

読み込まれるメッシュタイプが分散メッシュデータの場合, 実際に読み込むファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなります.

hecmw_dist_free

分散メッシュデータ構造体に確保されているメモリを解放します.

```
subroutine hecmw_dist_free(mesh)

type(hecmwST_local_mesh) :: mesh
```

引数

mesh

解放する分散メッシュデータ構造体

説明

分散メッシュデータ構造体に確保されているメモリを解放します.

hecmw_put_mesh

メッシュデータをファイルに出力します.

```
subroutine hecmw_put_mesh(name_ID, mesh)

character(len=HECMW_NAME_LEN) :: name_ID
type(hecmwST_local_mesh) :: mesh
```

引数

name_ID

!MESH を特定する識別子

mesh

出力するメッシュデータ

説明

メッシュデータをファイルに出力します.

出力対象となるメッシュデータは、引数 **mesh** に格納されているデータで、出力ファイルの種類は分散メッシュデータとなります.

出力されるファイルは、全体制御ファイルの**!MESH** で定義されており、かつ **NAME** が **name_ID** のものです.

出力されるファイルのファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものととなります.

hecmw_result_init

結果ファイル出力の初期化処理を行います.

```
subroutine hecmw_result_init(mesh, tstep, header)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: tstep
character(len=HECMW_HEADER_LEN) :: header
```

引数

mesh

メッシュデータ

tstep

タイムステップ

header

ヘッダ

説明

結果ファイル出力の初期化処理を行います.

これにより、結果ファイル出力のために必要な情報を取得します. ただし, `header` は結果ファイルにコメントとして用いられるだけで, 何ら影響を与えません.

hecmw_result_add

結果ファイルに出力するデータを指定します.

```
subroutine hecmw_result_add(node_or_elem, n_dof, label, data)

integer(kind=kint) :: node_or_elem
integer(kind=kint) :: n_dof
character(len=HECMW_NAME_LEN) :: label
real(kind=kreal) :: data
```

引数

node_or_elem

指定する値が節点値なのか要素値なのかを示す

1:節点 2:要素

n_dof

自由度数

label

ラベル

data

結果データ

説明

結果ファイルに出力するデータを指定します.

これは、複数回呼び出すことが可能です. この呼び出しによって指定されたデータの情報は、一旦 HEC-MW の内部に蓄えられます. 蓄えられたデータは、hecmw_result_write または hecmw_result_write_by_name によって出力されます.

この呼び出し以前に、hecmw_result_init によって初期化が行われていなければなりません.

hecmw_result_write

結果データをファイルへ出力します.

```
subroutine hecmw_result_write()
```

引数

なし

説明

結果データをファイルへ出力します.

出力されるデータは, `hecmw_result_add` で指定されたデータです.

出力されるファイルは, 全体制御ファイルの `!RESULT` で定義されており, `IO=OUT` かつ全体制御ファイル内で最初に定義されているファイルです.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなります.

hecmw_result_write_by_name

結果データをファイルへ出力します.

```
subroutine hecmw_result_write_by_name(name_ID)

character(len=HECMW_NAME_LEN) :: name_ID
```

引数

name_ID

!RESULT を特定する識別子

説明

結果データをファイルへ出力します.

出力ファイルは, 全体制御ファイルで指定された!RESULT のうち, NAME が name_ID のものです. この場合, !RESULT の IO パラメータは無視されます.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなります.

ファイル名の取得方法以外は, hecmw_result_write と同等です.

hecmw_result_write_st

結果データをファイルへ出力します.

```
subroutine hecmw_result_write_st(result_data, n_node, n_elem, timestep,
header)

type(hecmwST_result_data)::result_data
integer(kind=kint)::n_node,n_elem,timestep
character(len=HECMW_HEADER_LEN)::header
```

引数

result_data	結果データ構造体
n_node	節点数
n_elem	要素数
timestep	タイムステップ
header	結果ファイルヘッダ

説明

結果データをファイルへ出力します.

出力されるデータは, **result_data** で指定されたデータです.

出力されるファイルは, 全体制御ファイルの**!RESULT** で定義されており, **IO=OUT** かつ全体制御ファイル内で最初に定義されているファイルです.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<timestep>」を付加したものとなります.

hecmw_result_write_st_by_name

結果データをファイルへ出力します.

```
subroutine    hecmw_result_write_st_by_name(name_ID,    result_data,
n_node, n_elem, tstep, header)

character(len=HECMW_NAME_LEN) :: name_ID
type(hecmwST_result_data)::result_data
integer(kind=kint)::n_node,n_elem,tstep
character(len=HECMW_HEADER_LEN)::header
```

引数

name_ID

!RESULT を特定する識別子

result_data

結果データ構造体

n_node

節点数

n_elem

要素数

tstep

タイムステップ

header

結果ファイルヘッダ

説明

引数 **result_data** の結果データをファイルへ出力します.

出力ファイルは, 全体制御ファイルで指定された!RESULT のうち, NAME が **name_ID** のものです. この場合, !RESULT の IO パラメータは無視されます.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなります.

hecmw_result_finalize

結果データ出力の後処理を行います.

```
subroutine hecmw_result_finalize()
```

引数

なし

説明

hecmw_result_add で指定された結果データの登録をクリアします.

hecmw_result_read

結果ファイルから結果データを入力します.

```
subroutine hecmw_result_read(tstep, result)

integer(kind=kint) :: tstep
type(hecmwST_result_data) :: result
```

引数

tstep

タイムステップ

result

結果データ格納用構造体

説明

結果ファイル内の結果データが読み込まれ, **result** に格納されます.

入力されるファイルは, 全体制御ファイルの **!RESULT** で定義されており, **IO=IN** かつ全体制御ファイル内で最初に定義されているファイルです.

入力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなります.

hecmw_result_read_by_name

結果ファイルから結果データを入力します.

```
subroutine hecmw_result_read_by_name(name_ID, tstep, result)

character(len=HECMW_NAME_LEN) :: name_ID
integer(kind=kint) :: tstep
type(hecmwST_result_data) :: result
```

引数

name_ID

!RESULT を特定する識別子

tstep

タイムステップ

result

結果データ格納用構造体

説明

結果ファイル内の結果データが読み込まれ, result に格納されます.

入力ファイルは, 全体制御ファイルで指定された!RESULT のうち, NAME が name_ID のものです. この場合, !RESULT の IO パラメータは無視されます.

ファイル名の取得方法以外は, hecmw_result_read と同等です.

hecmw_restart_add_int

リスタートファイルに出力するデータを指定します.

```
subroutine hecmw_restart_add_int(data, n_data)

integer(kind=kint),dimension(:) :: data
integer(kind=kint) :: n_data
```

引数

data

リスタートデータ (整数型一次元配列)

n_data

配列要素数

説明

リスタートファイルに出力するデータを指定します.

これは、複数回呼び出すことが可能です. この呼び出しによって指定されたデータの情報は、一旦 HEC-MW の内部に蓄えられます. 蓄えられたデータは、hecmw_restart_write または hecmw_restart_write_by_name によって出力されます. したがって、実際に出力が完了するまでデータを変更してはなりません.

hecmw_restart_add_real

リスタートファイルに出力するデータを指定します.

```
subroutine hecmw_restart_add_int(data, n_data)

integer(kind=kint),dimension(:) :: data
integer(kind=kint) :: n_data
```

引数

data

リスタートデータ（浮動小数点型一次元配列）

n_data

配列要素数

説明

引数の型が浮動小数点型一次元配列であること意外は, hecmw_restart_add_real と同等です.

hecmw_restart_write

リスタートデータをファイルへ出力します.

```
subroutine hecmw_restart_write()
```

引数

なし

説明

リスタートデータをファイルへ出力します.

出力されるデータは, hecmw_restart_add_int または hecmw_restart_real で指定されたデータです.

出力されるファイルは, 全体制御ファイルの!RESTART で定義されており, IO=OUT または IO=INOUT, かつ全体制御ファイル内で最初に定義されているファイルです.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなります.

hecmw_restart_write_by_name

リスタートデータをファイルへ出力します.

```
subroutine hecmw_restart_write_by_name(name_ID)

character(len=HECMW_NAME_LEN) :: name_ID
```

引数

name_ID

!RESTART を特定する識別子

説明

リスタートデータをファイルへ出力します.

出力ファイルは, 全体制御ファイルで指定された!RESTART のうち, NAME が name_ID のものです. この場合, !RESTART の IO パラメータは無視されます.

出力されるファイルのファイル名は, 全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなります.

ファイル名の取得方法以外は, hecmw_restart_write と同等です.

hecmw_restart_open

リスタートファイルを入力用にオープンします.

```
hecmw_restart_open()
```

引数

なし

説明

リスタートファイルを入力用にオープンします.

オープンされるファイルは、全体制御ファイルの!RESTART で定義されており, IO=IN または IO=INOUT, かつ全体制御ファイル内で最初に定義されているファイルです.

hecmw_restart_open_by_name

リスタートファイルを入力用にオープンします.

```
subroutine hecmw_restart_open_by_name(name_ID)

character(len=HECMW_NAME_LEN) :: name_ID
```

引数

name_ID

!RESTART を特定する識別子

説明

リスタートファイルを入力用にオープンします.

出力ファイルは、全体制御ファイルで指定された!RESTART のうち、NAME が name_ID のものです. この場合、!RESTART の IO パラメータは無視されます.

hecmw_restart_read_int

リスタートデータからデータを入力します.

```
subroutine hecmw_restart_read_int(dst)

integer(kind=kint), dimension(:) :: dst
```

引数

dst

リスタートデータ格納先 (整数型一次元配列)

説明

リスタートファイル内のデータの一部が読み込まれ, dst に格納されます.

この呼び出し以前に, hecmw_restart_open または hecmw_restart_open_by_name によって入力リスタートファイルがオープンされていなければなりません.

dst には, データ格納に必要な領域を事前に確保しておかなければなりません.

この呼び出しで注意すべきことは, リスタートファイルの入力と出力で整合性がとれていなければならないということです.

リスタートファイルの入力には hecmw_restart_write_int または hecmw_restart_write_real を使用しますが, その順番とそれぞれの出力で出力したデータサイズに入力時もある必要があります. つまり, hecmw_restart_write_int で出力したデータは, hecmw_restart_read_int で入力せねばならず, かつその配列の要素サイズを同じにしておかなければなりません. これは, hecmw_restart_write_real, hecmw_restart_read_real についても同じです.

hecmw_restart_read_real

リスタートデータからデータを入力します.

```
subroutine hecmw_restart_read_real(dst)

real(kind=kreal),dimension(:) :: dst
```

引数

dst

リスタートデータ格納先 (浮動小数点型一次元配列)

説明

引数の型が浮動小数点型一次元配列であること意外は, hecmw_restart_read_real と同等です.

hecmw_restart_close

リスタートファイルをクローズします.

```
subroutine hecmw_restart_close()
```

引数

なし

説明

オープンされているリスタートファイルをクローズします.

hecmw_visualize_init

可視化の初期化処理を行います.

```
subroutine hecmw_visualize_init()
```

引数

なし

説明

可視化の初期化処理を行います.

hecmw_visualize(mesh, result, tstep, max_step, is_force)

可視化を行います.

```
subroutine hecmw_visualize(mesh, result, tstep, max_step, is_force)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_result_data) :: result
integer(kind=kint) :: tstep
integer(kind=kint) :: max_step
integer(kind=kint) :: is_force
```

引数

mesh

メッシュデータ

result

可視化用結果データ

t step

タイムステップ

max_step

最大タイムステップ

is_force

最後のステップも強制的に描くかどうかを示す.

0:描かない 1:描く

説明

可視化を行います.

hecmw_visualize_init によって事前に初期化されている必要があります.

可視化に必要な結果データには **result**, メッシュには **mesh** が使用されます.

hecmw_visualize_finalize

可視化の後処理を行います.

```
subroutine hecmw_visualize_finalize
```

引数

なし

説明

可視化の後処理を行います.

hecmw_ctrl_get_control_file

制御ファイル名を取得します.

```
subroutine hecmw_ctrl_get_control_file(name_ID, filename)

character(len=HECMW_NAME_LEN) :: name_ID
character(len=HECMW_FILENAME_LEN) :: filename
```

引数

name_ID

!CONTROL を特定する識別子

filename

制御ファイル名格納先

説明

全体制御ファイルの!CONTROL で定義した制御ファイル名を取得します.

8. C API リファレンス

8.1. データ構造一覧

以下にデータ構造の一覧を示す.

hecmw_bit_array (ビット配列構造体)	97
hecmw_coord (座標を表す構造体)	99
hecmw_ctrl_meshfile (全体制御ファイルから取得したメッシュファイルの情報を格納する)	101
hecmw_ctrl_meshfiles (メッシュ情報構造体)	103
hecmw_io_amplitude (AMPLITUDE情報)	105
hecmw_io_amplitude::hecmw_io_amplitude_item (AMPLITUDEアイテム)	107
hecmw_io_egrp (要素グループ情報)	109
hecmw_io_elem_surf (面グループアイテム)	110
hecmw_io_element (要素情報)	111
hecmw_io_header (ヘッダ情報)	113
hecmw_io_id (Int型IDの単方向リンクリストを表す)	114
hecmw_io_id_array (Int型IDの配列を表す)	115
hecmw_io_initial (初期条件情報)	116
hecmw_io_material (材料情報)	118
hecmw_io_material::hecmw_io_matitem (材料ITEM)	120
hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem (材料SUBITEM)	122
hecmw_io_mpc (拘束グループ情報)	124
hecmw_io_mpc::hecmw_io_mpcitem (拘束グループアイテム)	126
hecmw_io_ngrp (節点グループ情報)	128
hecmw_io_node (節点情報)	129
hecmw_io_section (セクション情報)	130
hecmw_io_section::hecmw_io_section_item (セクションアイテム)	132
hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface (インタフェースセ クション)	134
hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell (シェルセクション) ..	136

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid (ソリッドセクション)	138
hecmw_io_sgrp (面グループ情報)	139
hecmw_io_zero (絶対零度情報)	140
hecmw_map_int_value (マップの整数キーと値のペアを格納する)	141
hecmw_map_int_pair (マップの整数キーと内部インデックスのペアを格納する)	143
hecmw_map_int (整数キーとデータのマップ構造体)	145
hecmw_msgent (HEC-MWメッセージ情報構造体)	148
hecmw_set_int (整数データの集合構造体)	150
hecmw_system_param (座標変換パラメータ)	153
hecmwST_amplitude (AMPLITUDE情報構造体)	155
hecmwST_elem_grp (要素グループ情報構造体)	159
hecmwST_local_mesh (メッシュデータ構造体)	163
hecmwST_material (材料情報構造体)	183
hecmwST_mpc (拘束グループ情報構造体)	187
hecmwST_node_grp (節点グループ情報構造体)	189
hecmwST_result_data (結果データ構造体)	193
hecmwST_section (セクション情報構造体)	196
hecmwST_surf_grp (面グループ情報構造体)	200

8.2. HEC-MW データ構造

構造体 `hecmw_bit_array`

ビット配列構造体

```
#include <hecmw_bit_array.h>
```

変数

- `int len`
配列の長さ
- `unsigned long * vals`
ビット配列を格納する配列

説明

ビット配列構造体

構造体

`int hecmw_bit_array::len`

配列の長さ

`unsigned long* hecmw_bit_array::vals`

ビット配列を格納する配列

この構造体の説明は次のファイルから生成されました:

- **hecmw_bit_array.h**

構造体 hecmw_coord

座標を表す構造体

```
#include <hecmw_geometric.h>
```

変数

- double **x**
x座標値
- double **y**
y座標値
- double **z**
z座標値

説明

座標を表す構造体

構造体

double hecmw_coord::x

x座標値

double hecmw_coord::y

y座標値

double hecmw_coord::z

z座標値

この構造体の説明は次のファイルから生成されました:

- **hecmw_geometric.h**

構造体 `hecmw_ctrl_meshfile`

全体制御ファイルから取得したメッシュファイルの情報を格納する

```
#include <hecmw_control.h>
```

変数

- `int type`
メッシュの種類(!*MESH*の*TYPE*)
- `int io`
メッシュの入出力属性(!*MESH*の*IO*パラメータ)
- `char * filename`
メッシュのファイル名 (パスを含む)

説明

全体制御ファイルから取得したメッシュファイルの情報を格納する

一つのメッシュ情報を格納する

構造体

`int hecmw_ctrl_meshfile::type`

メッシュの種類(!*MESH*の*TYPE*)

int hecmw_ctrl_meshfile::io

メッシュの入出力属性(!MESHのIOパラメータ)

char* hecmw_ctrl_meshfile::filename

メッシュのファイル名 (パスを含む)

この構造体の説明は次のファイルから生成されました:

- **hecmw_control.h**

構造体 hecmw_ctrl_meshfiles

メッシュ情報構造体

```
#include <hecmw_control.h>
```

変数

- **int n_mesh**
この構造体に含まれるメッシュ情報の個数
- **hecmw_ctrl_meshfile * meshfiles**
メッシュ情報格納先

説明

メッシュ情報構造体

全体制御ファイルから取得したメッシュファイルの情報を格納する 複数のメッシュ情報を格納することができる

構造体

```
int hecmw_ctrl_meshfiles::n_mesh
```

この構造体に含まれるメッシュ情報の個数

```
struct hecmw_ctrl_meshfile* hecmw_ctrl_meshfiles::meshfiles
```

メッシュ情報格納先

meshfiles[0] ... meshfiles[n_mesh-1]

この構造体の説明は次のファイルから生成されました:

- **hecmw_control.h**

構造体 hecmw_io_amplitude

AMPLITUDE情報.

```
#include <hecmw_io_struct.h>
```

変数

- char **name** [HECMW_NAME_LEN+1]
 - int **type_def**
 - int **type_time**
 - int **type_val**
 - **hecmw_io_amplitude::hecmw_io_amplitude_item * item**
AMPLITUDE アイテム.
 - **hecmw_io_amplitude * next**
-

説明

AMPLITUDE情報.

構造体

```
char hecmw_io_amplitude::name[HECMW_NAME_LEN+1]
```

```
int hecmw_io_amplitude::type_def
```

```
int hecmw_io_amplitude::type_time
```

```
int hecmw_io_amplitude::type_val
```

```
struct hecmw_io_amplitude::hecmw_io_amplitude_item *  
hecmw_io_amplitude::item
```

AMPLITUDEアイテム.

```
struct hecmw_io_amplitude* hecmw_io_amplitude::next
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 `hecmw_io_amplitude::hecmw_io_amplitude_item`

AMPLITUDEアイテム.

```
#include <hecmw_io_struct.h>
```

変数

- `double val`
 - `double table`
 - `hecmw_io_amplitude_item * next`
-

説明

AMPLITUDEアイテム.

構造体

```
double hecmw_io_amplitude::hecmw_io_amplitude_item::val
```

```
double hecmw_io_amplitude::hecmw_io_amplitude_item::table
```

```
struct hecmw_io_amplitude_item*  
hecmw_io_amplitude::hecmw_io_amplitude_item::next
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 hecmw_io_egrp

要素グループ情報

```
#include <hecmw_io_struct.h>
```

変数

- char **name** [HECMW_NAME_LEN+1]
 - hecmw_set_int * **elem**
 - hecmw_io_egrp * **next**
-

説明

要素グループ情報

構造体

```
char hecmw_io_egrp::name[HECMW_NAME_LEN+1]
```

```
struct hecmw_set_int* hecmw_io_egrp::elem
```

```
struct hecmw_io_egrp* hecmw_io_egrp::next
```

この構造体の説明は次のファイルから生成されました:

- hecmw_io_struct.h

構造体 `hecmw_io_elem_surf`

面グループアイテム

```
#include <hecmw_io_struct.h>
```

変数

- `int elem`
 - `int surf`
-

説明

面グループアイテム

構造体

```
int hecmw_io_elem_surf::elem
```

```
int hecmw_io_elem_surf::surf
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 hecmw_io_element

要素情報

```
#include <hecmw_io_struct.h>
```

変数

- int **type**
 - int * **node**
 - int **nmatitem**
 - double * **matitem**
 - char **matname** [HECMW_NAME_LEN+1]
 - int **mpc_matid**
 - int **mpc_sectid**
-

説明

要素情報

構造体

int hecmw_io_element::type

int* hecmw_io_element::node

int hecmw_io_element::nmatitem

double* hecmw_io_element::matitem

char hecmw_io_element::matname[HECMW_NAME_LEN+1]

int hecmw_io_element::mpc_matid

int hecmw_io_element::mpc_sectid

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 `hecmw_io_header`

ヘッダ情報

```
#include <hecmw_io_struct.h>
```

変数

- `char header [HECMW_HEADER_LEN+1]`
-

説明

ヘッダ情報

構造体

```
char hecmw_io_header::header[HECMW_HEADER_LEN+1]
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 `hecmw_io_id`

`int`型IDの単方向リンクリストを表す

```
#include <hecmw_io_struct.h>
```

変数

- `int id`
 - `hecmw_io_id * next`
-

説明

`int`型IDの単方向リンクリストを表す

構造体

```
int hecmw_io_id::id
```

```
struct hecmw_io_id* hecmw_io_id::next
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 `hecmw_io_id_array`

`int`型IDの配列を表す

```
#include <hecmw_io_struct.h>
```

変数

- `int n`
 - `int * id`
-

説明

`int`型IDの配列を表す

構造体

`int hecmw_io_id_array::n`

`int* hecmw_io_id_array::id`

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 hecmw_io_initial

初期条件情報

```
#include <hecmw_io_struct.h>
```

変数

- int **type**
 - int **node**
 - char **ngrp** [HECMW_NAME_LEN+1]
 - double **val**
 - **hecmw_io_initial** * next
-

説明

初期条件情報

構造体

int hecmw_io_initial::type

int hecmw_io_initial::node

char hecmw_io_initial::ngrp[HECMW_NAME_LEN+1]

double hecmw_io_initial::val

struct hecmw_io_initial* hecmw_io_initial::next

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_material

材料情報

```
#include <hecmw_io_struct.h>
```

変数

- char **name** [HECMW_NAME_LEN+1]
 - int **nitem**
 - **hecmw_io_material::hecmw_io_matitem * item**
材料ITEM
 - **hecmw_io_material * next**
-

説明

材料情報

構造体

```
char hecmw_io_material::name[HECMW_NAME_LEN+1]
```

```
int hecmw_io_material::nitem
```

```
struct hecmw_io_material::hecmw_io_matitem * hecmw_io_material::item
```

材料ITEM

struct hecmw_io_material* hecmw_io_material::next

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_material::hecmw_io_matitem

材料ITEM

```
#include <hecmw_io_struct.h>
```

変数

- int **item**
- int **nval**
- hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem * **subitem**
材料SUBITEM

説明

材料ITEM

構造体

```
int hecmw_io_material::hecmw_io_matitem::item
```

```
int hecmw_io_material::hecmw_io_matitem::nval
```

```
struct hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem *  
hecmw_io_material::hecmw_io_matitem::subitem
```

材料SUBITEM

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体

hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem

材料SUBITEM

```
#include <hecmw_io_struct.h>
```

変数

- **double * val**
 - **double temp**
 - **hecmw_io_matsubitem * next**
-

説明

材料SUBITEM

構造体

double* hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem::val

double hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem::temp

struct hecmw_io_matsubitem*

hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem::next

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_mpc

拘束グループ情報

```
#include <hecmw_io_struct.h>
```

変数

- `int neq`
 - `hecmw_io_mpc::hecmw_io_mpcitem * item`
拘束グループアイテム
 - `hecmw_io_mpc * next`
-

説明

拘束グループ情報

構造体

```
int hecmw_io_mpc::neq
```

```
struct hecmw_io_mpc::hecmw_io_mpcitem * hecmw_io_mpc::item
```

拘束グループアイテム

```
struct hecmw_io_mpc* hecmw_io_mpc::next
```

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 `hecmw_io_mpc::hecmw_io_mpcitem`

拘束グループアイテム

```
#include <hecmw_io_struct.h>
```

変数

- `char ngrp [HECMW_NAME_LEN+1]`
 - `int node`
 - `int dof`
 - `double a`
-

説明

拘束グループアイテム

構造体

```
char hecmw_io_mpc::hecmw_io_mpcitem::ngrp[HECMW_NAME_LEN+1]
```

```
int hecmw_io_mpc::hecmw_io_mpcitem::node
```

```
int hecmw_io_mpc::hecmw_io_mpcitem::dof
```

```
double hecmw_io_mpc::hecmw_io_mpcitem::a
```

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_ngrp

節点グループ情報

```
#include <hecmw_io_struct.h>
```

変数

- `char name [HECMW_NAME_LEN+1]`
 - `hecmw_set_int * node`
 - `hecmw_io_ngrp * next`
-

説明

節点グループ情報

構造体

```
char hecmw_io_ngrp::name[HECMW_NAME_LEN+1]
```

```
struct hecmw_set_int* hecmw_io_ngrp::node
```

```
struct hecmw_io_ngrp* hecmw_io_ngrp::next
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 hecmw_io_node

節点情報

```
#include <hecmw_io_struct.h>
```

変数

- double **x**
 - double **y**
 - double **z**
-

説明

節点情報

構造体

double hecmw_io_node::x

double hecmw_io_node::y

double hecmw_io_node::z

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_section

セクション情報

```
#include <hecmw_io_struct.h>
```

変数

- char **egrp** [HECMW_NAME_LEN+1]
 - char **material** [HECMW_NAME_LEN+1]
 - int **composite**
 - int **secopt**
 - int **type**
 - **hecmw_io_section::hecmw_io_section_item** sect
セクションアイテム
 - **hecmw_io_section** * next
-

説明

セクション情報

構造体

```
char hecmw_io_section::egrp[HECMW_NAME_LEN+1]
```

```
char hecmw_io_section::material[HECMW_NAME_LEN+1]
```

```
int hecmw_io_section::composite
```

int hecmw_io_section::secopt

int hecmw_io_section::type

union hecmw_io_section::hecmw_io_section_item hecmw_io_section::sect

セクションアイテム

struct hecmw_io_section* hecmw_io_section::next

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

共用体 `hecmw_io_section::hecmw_io_section_item`

セクションアイテム

```
#include <hecmw_io_struct.h>
```

変数

- `hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid solid`
ソリッドセクション
- `hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell shell`
シェルセクション
- `hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface interface`
インタフェースセクション

説明

セクションアイテム

構造体

```
struct hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid  
hecmw_io_section::hecmw_io_section_item::solid
```

ソリッドセクション


```
struct hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell  
hecmw_io_section::hecmw_io_section_item::shell
```

シェルセクション

```
struct hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface  
hecmw_io_section::hecmw_io_section_item::interface
```

インタフェースセクション

この共用体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface

インタフェースセクション

```
#include <hecmw_io_struct.h>
```

変数

- double **thickness**
- double **gapcon**
- double **gaprad1**
- double **gaprad2**

説明

インタフェースセクション

構造体

double

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface::thickness

double

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface::gapcon

double

**hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface::gapr
ad1**

double

**hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface::gapr
ad2**

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell

シェルセクション

```
#include <hecmw_io_struct.h>
```

変数

- double **thickness**
 - int **integpoints**
-

説明

シェルセクション

構造体

double

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell::thickness

int

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell::integpoints

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid

ソリッドセクション

```
#include <hecmw_io_struct.h>
```

変数

- double **thickness**
-

説明

ソリッドセクション

構造体

double

hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid::thickness

s

この構造体の説明は次のファイルから生成されました:

- **hecmw_io_struct.h**

構造体 hecmw_io_sgrp

面グループ情報

```
#include <hecmw_io_struct.h>
```

変数

- `char name [HECMW_NAME_LEN+1]`
 - `hecmw_set_int * item`
 - `hecmw_io_sgrp * next`
-

説明

面グループ情報

構造体

```
char hecmw_io_sgrp::name[HECMW_NAME_LEN+1]
```

```
struct hecmw_set_int* hecmw_io_sgrp::item
```

```
struct hecmw_io_sgrp* hecmw_io_sgrp::next
```

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 `hecmw_io_zero`

絶対零度情報

```
#include <hecmw_io_struct.h>
```

変数

- `double zero`
-

説明

絶対零度情報

構造体

`double hecmw_io_zero::zero`

この構造体の説明は次のファイルから生成されました:

- `hecmw_io_struct.h`

構造体 `hecmw_map_int_value`

マップの整数キーと値のペアを格納する

```
#include <hecmw_map_int.h>
```

変数

- `int key`
キー
- `void * val`
データ

説明

マップの整数キーと値のペアを格納する

構造体

`char* hecmw_map_int_value::key`

キー

`void* hecmw_map_int_value::val`

データ

この構造体の説明は次のファイルから生成されました:

- **hecmw_map_int.h**

構造体 `hecmw_map_int_pair`

マップの整数キーと内部インデックスのペアを格納する

```
#include <hecmw_map_int.h>
```

変数

- `int key`
キー
- `int local`
内部インデックス

説明

マップの整数キーと内部インデックスのペアを格納する

構造体

`char* hecmw_map_int_pair::key`

キー

`void* hecmw_map_int_pair::local`

内部インデックス

この構造体の説明は次のファイルから生成されました:

- **hecmw_map_int.h**

構造体 `hecmw_map_int`

整数キーとデータのマップ構造体

```
#include <hecmw_map_int.h>
```

変数

- `int n_val`
データの個数
- `int max_val`
確保されているメモリに格納可能なデータの最大個数
- `struct hecmw_map_int_value *vals`
キーとデータのペアの配列
- `struct hecmw_map_int_pair *pairs`
キーと内部インデックスのペアの配列
- `int checked`
キーに重複がないことが確認されていれば1、未確認なら0
- `int sorted`
`pairs`がキーに関してソートされていれば1、ソートされていなければ0
- `struct hecmw_bit_array *mark`
データのマーキング用ビット配列
- `int iter`
データを連続して取り出す際の現在のインデックス

- `void (* freefunc)(void *)`
データ開放用の関数へのポインタ

説明

整数キーとデータのマップ構造体

構造体

`int hecmw_map_int::n_val`

データの個数

`int hecmw_map_int::max_val`

確保されているメモリに格納可能なデータの最大個数

`struct hecmw_map_int_value * hecmw_map_int::vals`

キーとデータのペアの配列

`struct hecmw_map_int_pair * hecmw_map_int::pairs`

キーと内部インデックスのペアの配列

int hecmw_map_int::checked

キーに重複がないことが確認されていれば1、未確認なら0

int hecmw_map_int::sorted

pairsがキーに関してソートされていれば1、ソートされていなければ0

struct hecmw_bit_array * hecmw_map_int::mark

データのマーキング用ビット配列

int hecmw_map_int::iter

データを連続して取り出す際の現在のインデックス

void (*hecmw_map_int::freefunc)(void *)

データ開放用の関数へのポインタ

この構造体の説明は次のファイルから生成されました:

- **hecmw_map_int.h**

構造体 `hecmw_msgent`

HEC-MWメッセージ情報構造体.

```
#include <hecmw_msg.h>
```

変数

- `int msgno`
メッセージ番号
- `char * msgno_str`
メッセージ番号の文字列表記
- `char * msg`
メッセージ文字列

説明

HEC-MWメッセージ情報構造体.

構造体

`int hecmw_msgent::msgno`

メッセージ番号

`char* hecmw_msgent::msgno_str`

メッセージ番号の文字列表記

char* hecmw_msgent::msg

メッセージ文字列

この構造体の説明は次のファイルから生成されました:

- **hecmw_msg.h**

構造体 `hecmw_set_int`

整数データの集合構造体

```
#include <hecmw_set_int.h>
```

変数

- **`int n_val`**
データの個数
- **`int max_val`**
確保されているメモリに格納可能なデータの最大個数
- **`int *vals`**
データの配列
- **`int checked`**
データに重複がないことが確認されていれば1、未確認なら0
- **`int sorted`**
データがソートされていれば1、ソートされていなければ0
- **`int iter`**
データを連続して取り出す際の現在のインデックス

説明

整数データの集合構造体

構造体

int hecmw_set_int::n_val

データの個数

int hecmw_set_int::max_val

確保されているメモリに格納可能なデータの最大個数

int * hecmw_set_int::vals

データの配列

int hecmw_set_int::checked

データに重複がないことが確認されていれば1、未確認なら0

int hecmw_set_int::sorted

データがソートされていれば1、ソートされていなければ0

int hecmw_set_int::iter

データを連続して取り出す際の現在のインデックス

この構造体の説明は次のファイルから生成されました:

- **hecmw_set_int.h**

構造体 `hecmw_system_param`

座標変換パラメータ

```
#include <hecmw_system.h>
```

変数

- `double xa`
 - `double ya`
 - `double za`
 - `double xb`
 - `double yb`
 - `double zb`
 - `double xc`
 - `double yc`
 - `double zc`
-

説明

座標変換パラメータ

構造体

`double hecmw_system_param::xa`

`double hecmw_system_param::ya`

`double hecmw_system_param::za`

double hecmw_system_param::xb

double hecmw_system_param::yb

double hecmw_system_param::zb

double hecmw_system_param::xc

double hecmw_system_param::yc

double hecmw_system_param::zc

この構造体の説明は次のファイルから生成されました:

- **hecmw_system.h**

構造体 hecmwST_amplitude

AMPLITUDE情報構造体.

```
#include <hecmw_struct.h>
```

変数

- **int n_amp**
AMPLITUDE グループ数.
- **char ** amp_name**
AMPLITUDE グループ名.
- **int * amp_type_definition**
AMPLITUDE タイプ(DEFINITION).
- **int * amp_type_time**
AMPLITUDE タイプ(TIME).
- **int * amp_type_value**
AMPLITUDE タイプ(VALUE).
- **int * amp_index**
AMPLITUDE インデックス.
- **double * amp_val**
AMPLITUDE 値.
- **double * amp_table**
AMPLITUDE 時間.

説明

AMPLITUDE情報構造体.

構造体

int hecmwST_amplitude::n_amp

AMPLITUDEグループ数.

char hecmwST_amplitude::amp_name**

AMPLITUDEグループ名.

サイズ : n_amp

int* hecmwST_amplitude::amp_type_definition

AMPLITUDEタイプ(DEFINITION).

サイズ : n_amp

- 1: TABULAR

int* hecmwST_amplitude::amp_type_time

AMPLITUDEタイプ(TIME).

サイズ : n_amp

- 1:STEP TIME

int* hecmwST_amplitude::amp_type_value

AMPLITUDEタイプ(VALUE).

サイズ : n_amp

- 1: RELATIVE
- 2: ABSOLUTE

int* hecmwST_amplitude::amp_index

AMPLITUDEインデックス.

サイズ : n_amp+1

double* hecmwST_amplitude::amp_val

AMPLITUDE値.

サイズ : amp_index[n_amp]

double* hecmwST_amplitude::amp_table

AMPLITUDE時間.

サイズ : amp_index[n_amp]

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_elem_grp

要素グループ情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_grp**
要素グループ数
- **char ** grp_name**
要素グループ名
- **int * grp_index**
要素グループ要素用インデックス
- **int * grp_item**
要素グループ要素用配列
- **int n_bc**
境界条件設定要素数
- **int * bc_grp_ID**
所属要素グループ番号
- **int * bc_grp_type**
境界条件タイプ
- **int * bc_grp_index**
境界条件設定要素番号

- `double * bc_grp_val`

境界条件値

説明

要素グループ情報構造体

構造体

`int hecmwST_elem_grp::n_grp`

要素グループ数

`char** hecmwST_elem_grp::grp_name`

要素グループ名

サイズ : `n_grp`

`int* hecmwST_elem_grp::grp_index`

要素グループ要素用インデックス

サイズ : `n_grp+1`

`int* hecmwST_elem_grp::grp_item`

要素グループ要素用配列

サイズ : `grp_index[n_grp]`

int hecmwST_elem_grp::n_bc

境界条件設定要素数

int* hecmwST_elem_grp::bc_grp_ID

所属要素グループ番号

サイズ : n_bc

int* hecmwST_elem_grp::bc_grp_type

境界条件タイプ.

サイズ : n_bc

- 1:変位
- 2:Flux
- 負:ユーザサブルーチン

int* hecmwST_elem_grp::bc_grp_index

境界条件設定要素番号

サイズ : n_bc

double* hecmwST_elem_grp::bc_grp_val

境界条件値

サイズ : n_bc

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_local_mesh

メッシュデータ構造体

```
#include <hecmw_struct.h>
```

変数

- **int hecmw_flag_adapt**
階層構造使用の有無
- **int hecmw_flag_initcon**
初期条件使用
- **int hecmw_flag_parttype**
領域分割形式
- **int hecmw_flag_partdepth**
部分領域間袖領域のオーバーラップ深さ
- **int hecmw_flag_version**
バージョン
- **char gridfile [HECMW_FILENAME_LEN+1]**
グリッドファイル名
- **int hecmw_n_file**
もろもろのファイル数
- **char ** files**
もろもろのファイル

- char **header** [HECMW_HEADER_LEN+1]

ヘッダ

- double **zero_temp**

絶対零度

- int **n_node**

総節点数

- int **nn_internal**

内部節点数

- int * **node_internal_list**

内部節点リスト

- int * **node_ID**

節点番号(ローカル番号,所属領域).

- int * **global_node_ID**

グローバル節点番号

- double * **node**

節点座標.

- int **n_dof**

節点最大自由度数

- int **n_dof_grp**

自由度グループ数

- int * **node_dof_index**

自由度インデックス

- **int * node_dof_item**
自由度
- **int * node_val_index**
節点物理量インデックス
- **double * node_val_item**
節点物理量
- **int * node_init_val_index**
初期条件インデックス
- **double * node_init_val_item**
初期条件
- **int n_elem**
総要素数
- **int ne_internal**
内部要素数
- **int * elem_internal_list**
内部要素リスト
- **int * elem_ID**
要素番号(ローカル番号, 所属領域).
- **int * global_elem_ID**
グローバル要素番号

- **int * elem_type**
要素タイプ
- **int n_elem_type**
要素タイプの数
- **int * elem_type_index**
要素タイプのインデックス
- **int * elem_type_item**
要素タイプ
- **int * elem_node_index**
要素コネクティビティ用インデックス
- **int * elem_node_item**
要素コネクティビティ用配列
- **int * section_ID**
セクション番号
- **int * elem_mat_ID_index**
材料物性ID用インデックス
- **int * elem_mat_ID_item**
材料物性ID配列
- **int n_elem_mat_ID**
= elem_mat_ID_index[n_elem]

- **int * elem_mat_int_index**
材料物性用インデックス(内部処理用)
- **double * elem_mat_int_val**
材料物性用配列
- **int * elem_val_index**
要素物理量インデックス
- **double * elem_val_item**
要素物理量
- **int zero**
領域番号が0か否か
- **HECMW_Comm HECMW_COMM**
MPI コミュニケータ.
- **int PETOT**
総領域数
- **int PEsmptTOT**
SMP ノードあたりのPE数.
- **int my_rank**
プロセス番号
- **int errnof**
エラー番号(Fotran でのみ使用)
- **int n_subdomain**

総領域数(局所分散データからの読み込み)

- **int n_neighbor_pe**
隣接領域数
- **int * neighbor_pe**
隣接領域番号
- **int * import_index**
受信テーブル用インデックス
- **int * import_item**
受信テーブル用配列
- **int * export_index**
送信テーブル用インデックス
- **int * export_item**
送信テーブル用配列
- **int * shared_index**
送受信テーブル用インデックス
- **int * shared_item**
送受信テーブル用配列
- **int coarse_grid_level**
最も荒いメッシュのレベル
- **int n_adapt**
Refinementされた回数

- **int * when_i_was_refined_node**
各節点の生成されたRefinement レベル
- **int * when_i_was_refined_elem**
各要素の生成されたRefinement レベル
- **int * adapt_parent_type**
各要素の親要素の分割タイプ
- **int * adapt_type**
各要素の分割タイプ
- **int * adapt_level**
各要素の分割レベル
- **int * adapt_parent**
各要素の親要素ID(ローカル番号, 所属領域).
- **int * adapt_children_index**
各要素の子要素用インデックス
- **int * adapt_children_item**
- **hecmwST_section * section**
セクション情報
- **hecmwST_material * material**
材料情報
- **hecmwST_mpc * mpc**
拘束グループ情報

- **hecmwST_amplitude * amp**
AMPLITUDE情報
- **hecmwST_node_grp * node_group**
節点グループ情報
- **hecmwST_elem_grp * elem_group**
要素グループ情報
- **hecmwST_surf_grp * surf_group**
面グループ情報

説明

メッシュデータ構造体

構造体

int hecmwST_local_mesh::hecmw_flag_adapt

階層構造使用の有無

0:NO, 1:YES

int hecmwST_local_mesh::hecmw_flag_initcon

初期条件使用

0:NO, 1:YES

int hecmwST_local_mesh::hecmw_flag_parttype

領域分割形式

0:Unknown, 1:Node-based, 2:Element-based

int hecmwST_local_mesh::hecmw_flag_partdepth

部分領域間袖領域のオーバーラップ深さ

int hecmwST_local_mesh::hecmw_flag_version

バージョン

char hecmwST_local_mesh::gridfile[HECMW_FILENAME_LEN+1]

グリッドファイル名

int hecmwST_local_mesh::hecmw_n_file

もろもろのファイル数

char hecmwST_local_mesh::files**

もろもろのファイル

char hecmwST_local_mesh::header[HECMW_HEADER_LEN+1]

ヘッダ

double hecmwST_local_mesh::zero_temp

絶対零度

int hecmwST_local_mesh::n_node

総節点数

int hecmwST_local_mesh::nn_internal

内部節点数

int* hecmwST_local_mesh::node_internal_list

内部節点リスト

サイズ : nn_internal

int* hecmwST_local_mesh::node_ID

節点番号(ローカル番号,所属領域).

サイズ : 2*n_node

- node_id[2*i] : local ID
- node_id[2*i+1]: PE

int* hecmwST_local_mesh::global_node_ID

グローバル節点番号

サイズ : n_node

double* hecmwST_local_mesh::node

節点座標.

サイズ : 3*n_node node[3*i] : X node[3*i+1]: Y node[3*i+2]: Z

int hecmwST_local_mesh::n_dof

節点最大自由度数

int hecmwST_local_mesh::n_dof_grp

自由度グループ数

int* hecmwST_local_mesh::node_dof_index

自由度インデックス

サイズ : n_dof_grp+1

int* hecmwST_local_mesh::node_dof_item

自由度

サイズ : n_dof_grp

int* hecmwST_local_mesh::node_val_index

節点物理量インデックス

サイズ : n_node+1

double* hecmwST_local_mesh::node_val_item

節点物理量

サイズ : node_val_index[n_node]

int* hecmwST_local_mesh::node_init_val_index

初期条件インデックス

サイズ : n_node+1

double* hecmwST_local_mesh::node_init_val_item

初期条件

サイズ : node_init_val_index[n_node]

int hecmwST_local_mesh::n_elem

総要素数

int hecmwST_local_mesh::ne_internal

内部要素数

int* hecmwST_local_mesh::elem_internal_list

内部要素リスト

サイズ : ne_internal

int* hecmwST_local_mesh::elem_ID

要素番号(ローカル番号, 所属領域).

サイズ : $2 * n_elem$

- $elem_ID[2*i]$: local ID
- $elem_ID[2*i+1]$: PE

int* hecmwST_local_mesh::global_elem_ID

グローバル要素番号

サイズ : n_elem

int* hecmwST_local_mesh::elem_type

要素タイプ

サイズ : n_elem

int hecmwST_local_mesh::n_elem_type

要素タイプの数

int* hecmwST_local_mesh::elem_type_index

要素タイプのインデックス

サイズ : n_elem_type+1

int* hecmwST_local_mesh::elem_type_item

要素タイプ

サイズ : n_elem_type

int* hecmwST_local_mesh::elem_node_index

要素コネクティビティ用インデックス

サイズ : n_elem+1

int* hecmwST_local_mesh::elem_node_item

要素コネクティビティ用配列

サイズ : elem_node_index[n_elem]

int* hecmwST_local_mesh::section_ID

セクション番号

サイズ : n_elem

int* hecmwST_local_mesh::elem_mat_ID_index

材料物性ID用インデックス

サイズ : n_elem+1

int* hecmwST_local_mesh::elem_mat_ID_item

材料物性ID配列

サイズ : elem_mat_ID_index[n_elem]

int hecmwST_local_mesh::n_elem_mat_ID

= elem_mat_ID_index[n_elem]

int* hecmwST_local_mesh::elem_mat_int_index

材料物性用インデックス(内部処理用)

サイズ : n_elem+1

double* hecmwST_local_mesh::elem_mat_int_val

材料物性用配列

サイズ : elem_mat_int_index[n_elem]

int* hecmwST_local_mesh::elem_val_index

要素物理量インデックス

サイズ : n_elem+1

double* hecmwST_local_mesh::elem_val_item

要素物理量

サイズ : elem_val_index[n_elem]

int hecmwST_local_mesh::zero

領域番号が0か否か

1:領域番号==0, 1:領域番号!=0

HECMW_Comm hecmwST_local_mesh::HECMW_COMM

MPIコミュニケーター.

int hecmwST_local_mesh::PETOT

総領域数

int hecmwST_local_mesh::PEsmpTOT

SMPノードあたりのPE数.

int hecmwST_local_mesh::my_rank

プロセス番号

int hecmwST_local_mesh::errnof

エラー番号(Fotranでのみ使用)

int hecmwST_local_mesh::n_subdomain

総領域数(局所分散データからの読み込み)

int hecmwST_local_mesh::n_neighbor_pe

隣接領域数

int* hecmwST_local_mesh::neighbor_pe

隣接領域番号

サイズ : n_neighbor_pe

int* hecmwST_local_mesh::import_index

受信テーブル用インデックス

サイズ : n_neighbor_pe+1

int* hecmwST_local_mesh::import_item

受信テーブル用配列

サイズ : import_index[n_neighbor_pe]

int* hecmwST_local_mesh::export_index

送信テーブル用インデックス

サイズ : n_neighbor_pe+1

int* hecmwST_local_mesh::export_item

送信テーブル用配列

サイズ : export_index[n_neighbor_pe]

int* hecmwST_local_mesh::shared_index

送受信テーブル用インデックス

サイズ : n_neighbor_pe+1

int* hecmwST_local_mesh::shared_item

送受信テーブル用配列

サイズ : shared_index[n_neighbor_pe]

int hecmwST_local_mesh::coarse_grid_level

最も荒いメッシュのレベル

int hecmwST_local_mesh::n_adapt

Refinementされた回数.

int* hecmwST_local_mesh::when_i_was_refined_node

各節点の生成されたRefinementレベル

サイズ : n_node

int* hecmwST_local_mesh::when_i_was_refined_elem

各要素の生成されたRefinementレベル

サイズ : n_elem

int* hecmwST_local_mesh::adapt_parent_type

各要素の親要素の分割タイプ

サイズ : n_elem

int* hecmwST_local_mesh::adapt_type

各要素の分割タイプ

サイズ : n_elem

int* hecmwST_local_mesh::adapt_level

各要素の分割レベル

サイズ : n_elem

int* hecmwST_local_mesh::adapt_parent

各要素の親要素ID(ローカル番号, 所属領域).

サイズ : 2*n_elem

- adapt_parent[2*i] : local ID
- adapt_parent[2*i+1]: PE

int* hecmwST_local_mesh::adapt_children_index

各要素の子要素用インデックス

サイズ : n_elem+1

int* hecmwST_local_mesh::adapt_children_item

struct hecmwST_section* hecmwST_local_mesh::section

セクション情報

サイズ : 2*adapt_children_index[n_elem]

- adapt_children_item[2*i] : local_ID
- adapt_children_item[2*i+1]: PE

struct hecmwST_material* hecmwST_local_mesh::material

材料情報

struct hecmwST_mpc* hecmwST_local_mesh::mpc

拘束グループ情報

struct hecmwST_amplitude* hecmwST_local_mesh::amp

AMPLITUDE情報.

```
struct hecmwST_node_grp* hecmwST_local_mesh::node_group
```

節点グループ情報

```
struct hecmwST_elem_grp* hecmwST_local_mesh::elem_group
```

要素グループ情報

```
struct hecmwST_surf_grp* hecmwST_local_mesh::surf_group
```

面グループ情報

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_material

材料情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_mat**
材料数
- **int n_mat_item**
材料物性総数
- **int n_mat_subitem**
材料物性サブ項目総数
- **int n_mat_table**
材料物性テーブル総数
- **char ** mat_name**
材料物性名
- **int * mat_item_index**
材料数インデックス
- **int * mat_subitem_index**
材料物性数インデックス
- **int * mat_table_index**
材料物性テーブルインデックス

- double * **mat_val**
材料物性
- double * **mat_temp**
温度依存テーブル

説明

材料情報構造体

構造体

int hecmwST_material::n_mat

材料数

int hecmwST_material::n_mat_item

材料物性総数

int hecmwST_material::n_mat_subitem

材料物性サブ項目総数

int hecmwST_material::n_mat_table

材料物性テーブル総数

char hecmwST_material::mat_name**

材料物性名

サイズ：n_mat

int* hecmwST_material::mat_item_index

材料数インデックス

サイズ：n_mat+1

int* hecmwST_material::mat_subitem_index

材料物性数インデックス

サイズ：n_mat_item+1

int* hecmwST_material::mat_table_index

材料物性テーブルインデックス

サイズ：n_mat_subitem+1

double* hecmwST_material::mat_val

材料物性

サイズ：mat_table_index[n_mat_subitem]

double* hecmwST_material::mat_temp

温度依存テーブル

サイズ：mat_table_index[n_mat_subitem]

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_mpc

拘束グループ情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_mpc**
拘束グループ数
- **int * mpc_index**
拘束グループ用インデックス
- **int * mpc_item**
拘束グループ節点番号
- **int * mpc_dof**
拘束自由度情報
- **double * mpc_val**
拘束自由度係数

説明

拘束グループ情報構造体

構造体

int hecmwST_mpc::n_mpc

拘束グループ数

int* hecmwST_mpc::mpc_index

拘束グループ用インデックス

サイズ : n_mpc+1

int* hecmwST_mpc::mpc_item

拘束グループ節点番号

サイズ : mpc_index[n_mpc]

int* hecmwST_mpc::mpc_dof

拘束自由度情報

サイズ : mpc_index[n_mpc]

double* hecmwST_mpc::mpc_val

拘束自由度係数

サイズ : mpc_index[n_mpc]

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_node_grp

節点グループ情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_grp**
節点グループ数
- **char ** grp_name**
節点グループ名
- **int * grp_index**
節点グループ要素用インデックス
- **int * grp_item**
節点グループ要素用配列
- **int n_bc**
境界条件設定節点数(自由度数)
- **int * bc_grp_ID**
所属節点グループ番号
- **int * bc_grp_type**
境界条件タイプ
- **int * bc_grp_index**
境界条件設定節点番号

- `int * bc_grp_dof`
境界条件設定自由度
- `double * bc_grp_val`
境界条件値

説明

節点グループ情報構造体

構造体

`int hecmwST_node_grp::n_grp`

節点グループ数

`char** hecmwST_node_grp::grp_name`

節点グループ名

サイズ : `n_grp`

`int* hecmwST_node_grp::grp_index`

節点グループ要素用インデックス

サイズ : `n_grp+1`

`int* hecmwST_node_grp::grp_item`

節点グループ要素用配列

サイズ : grp_index[n_grp]

int hecmwST_node_grp::n_bc

境界条件設定節点数(自由度数)

int* hecmwST_node_grp::bc_grp_ID

所属節点グループ番号

サイズ : n_bc

int* hecmwST_node_grp::bc_grp_type

境界条件タイプ.

サイズ : n_bc

- 1:変位
- 2:Flux
- 負:ユーザサブルーチン

int* hecmwST_node_grp::bc_grp_index

境界条件設定節点番号

サイズ : n_bc

int* hecmwST_node_grp::bc_grp_dof

境界条件設定自由度

サイズ : n_bc

double* hecmwST_node_grp::bc_grp_val

境界条件値

サイズ : n_bc

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_result_data

結果データ構造体

```
#include <hecmw_result.h>
```

変数

- **int nn_component**
節点コンポーネント数
- **int ne_component**
要素コンポーネント数
- **int * nn_dof**
節点自由度数
- **int * ne_dof**
要素自由度数
- **char ** node_label**
節点コンポーネントラベル
- **char ** elem_label**
要素コンポーネントラベル
- **double * node_val_item**
節点値
- **double * elem_val_item**
要素値

説明

結果データ構造体

構造体

int hecmwST_result_data::nn_component

節点コンポーネント数

int hecmwST_result_data::ne_component

要素コンポーネント数

int* hecmwST_result_data::nn_dof

節点自由度数

nn_dof[0]...nn_dof[nn_component-1]

int* hecmwST_result_data::ne_dof

要素自由度数

ne_dof[0]...ne_dof[ne_component-1]

char hecmwST_result_data::node_label**

節点コンポーネントラベル

node_label[0]...node_label[nn_component-1]

char hecmwST_result_data::elem_label**

要素コンポーネントラベル

elem_label[0]...elem_label[ne_component-1]

double* hecmwST_result_data::node_val_item

節点値

node_val_item[0]...node_val_item[sum(nn_dof)*nn_component*節点数-1]

double* hecmwST_result_data::elem_val_item

要素値

elem_val_item[0]...elem_val_item[sum(ne_dof)*ne_component*要素数-1]

この構造体の説明は次のファイルから生成されました:

- **hecmw_result.h**

構造体 hecmwST_section

セクション情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_sect**
セクション数
- **int * sect_type**
セクションタイプ.
- **int * sect_opt**
- **int * sect_mat_ID_index**
材料物性用インデックス
- **int * sect_mat_ID_item**
材料物性ID
- **int * sect_I_index**
セクション値用インデックス(整数)
- **int * sect_I_item**
セクション値配列(整数)
- **int * sect_R_index**
セクション値用インデックス(実数)
- **double * sect_R_item**
セクション値用配列(実数)

説明

セクション情報構造体

構造体

int hecmwST_section::n_sect

セクション数

int* hecmwST_section::sect_type

セクションタイプ.

サイズ : n_sect

- 1:SOLID
- 2:SHELL
- 3:BEAM
- 4:INTERFACE

int* hecmwST_section::sect_opt

int* hecmwST_section::sect_mat_ID_index

材料物性用インデックス

サイズ : n_sect+1

int* hecmwST_section::sect_mat_ID_item

材料物性ID

サイズ : sect_mat_ID_index[n_sect]

int* hecmwST_section::sect_I_index

セクション値用インデックス(整数)

サイズ : n_sect+1

int* hecmwST_section::sect_I_item

セクション値配列(整数)

サイズ : sect_I_index[n_sect]

int* hecmwST_section::sect_R_index

セクション値用インデックス(実数)

サイズ : n_sect+1

double* hecmwST_section::sect_R_item

セクション値用配列(実数)

サイズ : sect_R_index[n_sect]

この構造体の説明は次のファイルから生成されました:

- **hecmw_struct.h**

構造体 hecmwST_surf_grp

面グループ情報構造体

```
#include <hecmw_struct.h>
```

変数

- **int n_grp**
面グループ数
- **char ** grp_name**
面グループ名
- **int * grp_index**
面グループ要素用インデックス
- **int * grp_item**
面グループ要素用配列(ローカル要素番号, 局所面番号) サイズ: $2 * \text{grp_index}[\text{n_grp}]$
- **int n_bc**
境界条件設定用素数(自由度数)
- **int * bc_grp_ID**
所属要素グループ番号
- **int * bc_grp_type**
境界条件タイプ
- **int * bc_grp_index**
境界条件設定面番号(要素, 局所面番号).

- `double * bc_grp_val`

境界条件値

説明

面グループ情報構造体

構造体

`int hecmwST_surf_grp::n_grp`

面グループ数

`char** hecmwST_surf_grp::grp_name`

面グループ名

サイズ : `n_grp`

`int* hecmwST_surf_grp::grp_index`

面グループ要素用インデックス

サイズ : `n_grp+1`

`int* hecmwST_surf_grp::grp_item`

面グループ要素用配列(ローカル要素番号, 局所面番号) サイズ : `2*grp_index[n_grp]`

- `grp_index[i*2]` : ローカル要素番号

- `grp_index[i*2+1]`: 局所面番号

`int hecmwST_surf_grp::n_bc`

境界条件設定用素数(自由度数)

`int* hecmwST_surf_grp::bc_grp_ID`

所属要素グループ番号

サイズ : `n_bc`

`int* hecmwST_surf_grp::bc_grp_type`

境界条件タイプ.

サイズ : `n_bc`

- 1:変位
- 2:Flux

`int* hecmwST_surf_grp::bc_grp_index`

境界条件設定面番号(要素, 局所面番号).

サイズ : `2*n_bc`

- `bc_grp_index[2*i]`: 要素番号

- `bc_grp_index[2*i+1]`: 局所面番号

`double* hecmwST_surf_grp::bc_grp_val`

境界条件値

サイズ : `n_bc`

この構造体の説明は次のファイルから生成されました:

- **`hecmw_struct.h`**

8.3. HEC-MW ファイル一覧

以下にファイル一覧を示す.

hecmw_ablex.h (ABAQUSメッシュデータの字句解析ルーチン)	206
hecmw_bit_array.h (ビット配列)	216
hecmw_comm.h (コミュニケーター関連)	220
hecmw_common.h (Commonディレクトリ内のヘッダをインクルード)	227
hecmw_common_define.h (要素情報の定数定義)	228
hecmw_config.h (HEC-MWで使用する定数等の定義)	294
hecmw_control.h (全体制御ファイルの読み込みと, その情報取得を行う)	300
hecmw_ctrllex.h (全体制御データの字句解析ルーチン)	312
hecmw_debug_write_dist.h (分散メッシュデータのDEBUG出力)	318
hecmw_dist.h (メッシュデータ構造体に関する処理)	333
hecmw_dist_copy_f2c.h (FortranからCへメッシュデータをコピーする)	336
hecmw_dist_free.h (メッシュデータ構造体のメモリ領域を解放する.)	338
hecmw_dist_print.h (メッシュデータ構造体の内容出力する)	342
hecmw_error.h (エラーが発生した場合, その情報を保存し, 他の関数から取得可能とする.)	350
hecmw_etype.h (要素情報を提供する)	354
hecmw_finalize.h (HEC-MWの全体終了処理を行う)	359
hecmw_geometric.h (幾何学的計算・変換処理を行う)	361
hecmw_gflex.h (GeoFEMメッシュデータの字句解析ルーチン)	364
hecmw_heclex.h (HEC-MW単一領域メッシュデータの字句解析ルーチン)	368
hecmw_init.h (HEC-MWの全体初期化処理を行う)	376
hecmw_io.h (IO関連のヘッダをインクルード)	378
hecmw_io_abaqus.h (ABAQUSメッシュデータ入力ルーチン)	379
hecmw_io_dist.h (HEC-MW分散メッシュデータ入出力ルーチン)	381
hecmw_io_geofem.h (GeoFEMメッシュデータ入力ルーチン)	383
hecmw_io_get_mesh.h (メッシュデータ取得ルーチン)	385
hecmw_io_hec.h (HEC-MW単一領域メッシュデータ入力ルーチン)	388
hecmw_io_mesh.h (全体メッシュデータの取得とメッシュデータ構造体生成)	390

hecmw_io_put_mesh.h (メッシュデータをファイルに出力する)	408
hecmw_io_struct.h (メッシュデータ入力時に使用するデータ構造)	410
hecmw_lib_fc.h (FortranとC間のデータ変換ユーティリティ)	413
hecmw_log.h (ログ出力を可能とするための関数ライブラリ)	416
hecmw_malloc.h (メモリリーク検出ライブラリ)	424
hecmw_map_int.h (正数をキーとするマップ)	431
hecmw_msg.h (HEC-MWメッセージ情報ライブラリ)	438
hecmw_path.h (パス名に関する処理を行うユーティリティ)	441
hecmw_reorder.h (メッシュ情報の並べ替え実装)	444
hecmw_restart.h (リスタートデータの入出力を行う)	446
hecmw_result.h (結果データ 入出力)	454
hecmw_result_copy_c2f.h (CからFortranへ結果データのコピーを行う)	461
hecmw_result_copy_f2c.h (FortranからCへ結果データのコピーを行う)	463
hecmw_set_int.h (整数の集合)	465
hecmw_struct.h (メッシュデータ構造体定義)	470
hecmw_system.h (局所座標系から全体座標系への変換を行う)	475
hecmw_ucd_print.h (UCDファイルを出力する)	477
hecmw_util.h (HEC-MWユーティリティ)	478

8.4. HEC-MW ファイル

hecmw_ablex.h

ABAQUSメッシュデータの字句解析ルーチン.

```
#include <stdio.h>
```

列挙型

- enum { **HECMW_ABLEX_NL** = 1000, **HECMW_ABLEX_INT**,
HECMW_ABLEX_DOUBLE, **HECMW_ABLEX_NAME**, **HECMW_ABLEX_FILENAME**,
HECMW_ABLEX_HEADER, **HECMW_ABLEX_H_AMPLITUDE** = 2000,
HECMW_ABLEX_H_CONDUCTIVITY, **HECMW_ABLEX_H_DENSITY**,
HECMW_ABLEX_H_ELASTIC, **HECMW_ABLEX_H_ELEMENT**,
HECMW_ABLEX_H_ELSET, **HECMW_ABLEX_H_EQUATION**,
HECMW_ABLEX_H_HEADING, **HECMW_ABLEX_H_INCLUDE**,
HECMW_ABLEX_H_INITIAL, **HECMW_ABLEX_H_MATERIAL**,
HECMW_ABLEX_H_NODE, **HECMW_ABLEX_H_NSET**,
HECMW_ABLEX_H_SHELL_SECTION, **HECMW_ABLEX_H_SOLID_SECTION**,
HECMW_ABLEX_H_SPECIFIC_HEAT, **HECMW_ABLEX_H_SYSTEM**,
HECMW_ABLEX_K_ABSOLUTE = 3000, **HECMW_ABLEX_K_ANISOTROPIC**,
HECMW_ABLEX_K_ELSET, **HECMW_ABLEX_K_ENGINEERING_CONSTANTS**,
HECMW_ABLEX_K_DEFINITION, **HECMW_ABLEX_K_DEPENDENCIES**,
HECMW_ABLEX_K_GENERATE, **HECMW_ABLEX_K_INPUT**,
HECMW_ABLEX_K_ISOTROPIC, **HECMW_ABLEX_K_LAMINA**,
HECMW_ABLEX_K_MATERIAL, **HECMW_ABLEX_K_NAME**,
HECMW_ABLEX_K_NSET, **HECMW_ABLEX_K_ORTHOTROPIC**,
HECMW_ABLEX_K_RELATIVE, **HECMW_ABLEX_K_STEP_TIME**,
HECMW_ABLEX_K_SYSTEM, **HECMW_ABLEX_K_TABULAR**,
HECMW_ABLEX_K_TEMPERATURE, **HECMW_ABLEX_K_TIME**,
HECMW_ABLEX_K_TYPE, **HECMW_ABLEX_K_UNSORTED**,
HECMW_ABLEX_K_VALUE, **HECMW_ABLEX_E_B31** = 4000,

HECMW_ABLEX_E_B32, HECMW_ABLEX_E_C3D4, HECMW_ABLEX_E_C3D6,
 HECMW_ABLEX_E_C3D8, HECMW_ABLEX_E_C3D8I, HECMW_ABLEX_E_C3D10,
 HECMW_ABLEX_E_C3D15, HECMW_ABLEX_E_C3D20, HECMW_ABLEX_E_CAX3,
 HECMW_ABLEX_E_CAX4, HECMW_ABLEX_E_CAX4I, HECMW_ABLEX_E_CAX4R,
 HECMW_ABLEX_E_CAX6, HECMW_ABLEX_E_CAX8, HECMW_ABLEX_E_CAX8R,
 HECMW_ABLEX_E_CPE3, HECMW_ABLEX_E_CPE4, HECMW_ABLEX_E_CPE4I,
 HECMW_ABLEX_E_CPE4R, HECMW_ABLEX_E_CPE6, HECMW_ABLEX_E_CPE8,
 HECMW_ABLEX_E_CPE8R, HECMW_ABLEX_E_CPS3, HECMW_ABLEX_E_CPS4,
 HECMW_ABLEX_E_CPS4I, HECMW_ABLEX_E_CPS4R, HECMW_ABLEX_E_CPS6,
 HECMW_ABLEX_E_CPS8, HECMW_ABLEX_E_CPS8R, HECMW_ABLEX_E_DC1D2,
 HECMW_ABLEX_E_DC1D3, HECMW_ABLEX_E_DC2D3, HECMW_ABLEX_E_DC2D4,
 HECMW_ABLEX_E_DC2D6, HECMW_ABLEX_E_DC2D8, HECMW_ABLEX_E_DC3D4,
 HECMW_ABLEX_E_DC3D6, HECMW_ABLEX_E_DC3D8, HECMW_ABLEX_E_DC3D10,
 HECMW_ABLEX_E_DC3D15, HECMW_ABLEX_E_DC3D20,
 HECMW_ABLEX_E_DCAX3, HECMW_ABLEX_E_DCAX4, HECMW_ABLEX_E_DCAX6,
 HECMW_ABLEX_E_DCAX8, HECMW_ABLEX_E_DINTER4,
 HECMW_ABLEX_E_DINTER8, HECMW_ABLEX_E_DS4, HECMW_ABLEX_E_DS8,
 HECMW_ABLEX_E_INTER4, HECMW_ABLEX_E_INTER8, HECMW_ABLEX_E_S3R,
 HECMW_ABLEX_E_S4R, HECMW_ABLEX_E_S8R, HECMW_ABLEX_E_T3D2,
 HECMW_ABLEX_E_T3D3 }

トークン番号定義

関数

- double **HECMW_ablex_get_number** (void)
 直前に読み込んだトークンが数値であった場合、その値を返す
- char * **HECMW_ablex_get_text** (void)
 直前に読み込んだトークンの文字列表記を返す
- int **HECMW_ablex_get_lineno** (void)
 読み込んでいるファイルの現在の行番号を返す

- **int HECMW_ablex_next_token** (void)
次のトークンを返す
- **int HECMW_ablex_next_token_skip** (int skip_token)
次のトークンを返す
- **int HECMW_ablex_set_input** (FILE *fp)
字句解析ルーチンで読み込むファイルのファイルポインタをセットする
- **int HECMW_ablex_skip_line** (void)
現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す
- **int HECMW_ablex_switch_to_include** (const char *filename)
現在読み込み中の字句解析ルーチンを中断し、新たなファイル(インクルードファイル)を解析対象とする
- **int HECMW_ablex_unput_token** (void)
直前に読み込んだトークンを押し戻す
- **int HECMW_ablex_is_including** (void)
現在解析中のファイルがインクルードファイルか否かを返す

説明

ABAQUS メッシュデータの字句解析ルーチン.

日付:

2004年2月16日

作者:

坂根 一彰

列挙型

anonymous enum

トークン番号定義

列挙型の値:

HECMW_ABLEX_NL 改行文字
HECMW_ABLEX_INT 整数
HECMW_ABLEX_DOUBLE 浮動小数点
HECMW_ABLEX_NAME 名前
HECMW_ABLEX_FILENAME ファイル名a
HECMW_ABLEX_HEADER ヘッダデータ
HECMW_ABLEX_H_AMPLITUDE *AMPLITUDE
HECMW_ABLEX_H_CONDUCTIVITY *CONDUCTIVITY
HECMW_ABLEX_H_DENSITY *DENSITY
HECMW_ABLEX_H_ELASTIC *ELASTIC
HECMW_ABLEX_H_ELEMENT *ELEMENT
HECMW_ABLEX_H_ELSET *ELSET
HECMW_ABLEX_H_EQUATION *EQUATION
HECMW_ABLEX_H_HEADING *HEADING
HECMW_ABLEX_H_INCLUDE *INCLUDE
HECMW_ABLEX_H_INITIAL *INITIAL CONDITIONS
HECMW_ABLEX_H_MATERIAL *MATERIAL
HECMW_ABLEX_H_NODE *NODE
HECMW_ABLEX_H_NSET *NSET
HECMW_ABLEX_H_SHELL_SECTION *SHELL SECTION
HECMW_ABLEX_H_SOLID_SECTION *SOLID SECTION
HECMW_ABLEX_H_SPECIFIC_HEAT *SPECIFIC HEAT
HECMW_ABLEX_H_SYSTEM *SYSTEM
HECMW_ABLEX_K_ABSOLUTE ABSOLUTE.
HECMW_ABLEX_K_ANISOTROPIC ANISOTROPIC.

HECMW_ABLEX_K_ELSET ELSE.
HECMW_ABLEX_K_ENGINEERING_CONSTANTS ENGINEERING CONSTANTS.
HECMW_ABLEX_K_DEFINITION DEFINITION.
HECMW_ABLEX_K_DEPENDENCIES DEPENDENCIES.
HECMW_ABLEX_K_GENERATE GENERATE.
HECMW_ABLEX_K_INPUT INPUT.
HECMW_ABLEX_K_ISOTROPIC ISOTROPIC.
HECMW_ABLEX_K_LAMINA LAMINA.
HECMW_ABLEX_K_MATERIAL MATERIAL.
HECMW_ABLEX_K_NAME NAME.
HECMW_ABLEX_K_NSET NSET.
HECMW_ABLEX_K_ORTHOTROPIC ORTHOTROPIC.
HECMW_ABLEX_K_RELATIVE RELATIVE.
HECMW_ABLEX_K_STEP_TIME STEP TIME.
HECMW_ABLEX_K_SYSTEM SYSTEM.
HECMW_ABLEX_K_TABULAR TABULAR.
HECMW_ABLEX_K_TEMPERATURE TEMPERATURE.
HECMW_ABLEX_K_TIME TIME.
HECMW_ABLEX_K_TYPE TYPE.
HECMW_ABLEX_K_UNSORTED UNSORTED.
HECMW_ABLEX_K_VALUE VALUE.
HECMW_ABLEX_E_B31 B31.
HECMW_ABLEX_E_B32 B32.
HECMW_ABLEX_E_C3D4 C3D4.
HECMW_ABLEX_E_C3D6 C3D6.
HECMW_ABLEX_E_C3D8 C3D8.
HECMW_ABLEX_E_C3D8I C3D8I.
HECMW_ABLEX_E_C3D10 C3D10.
HECMW_ABLEX_E_C3D15 C3D15.
HECMW_ABLEX_E_C3D20 C3D20.
HECMW_ABLEX_E_CAX3 CAX3.
HECMW_ABLEX_E_CAX4 CAX4.
HECMW_ABLEX_E_CAX4I CAX4I.
HECMW_ABLEX_E_CAX4R CAX4R.
HECMW_ABLEX_E_CAX6 CAX6.
HECMW_ABLEX_E_CAX8 CAX8.

HECMW_ABLEX_E_CAX8R CAX8R.
HECMW_ABLEX_E_CPE3 CPE3.
HECMW_ABLEX_E_CPE4 CPE4.
HECMW_ABLEX_E_CPE4I CPE4I.
HECMW_ABLEX_E_CPE4R CPE4R.
HECMW_ABLEX_E_CPE6 CPE6.
HECMW_ABLEX_E_CPE8 CPE8.
HECMW_ABLEX_E_CPE8R CPE8R.
HECMW_ABLEX_E_CPS3 CPES3.
HECMW_ABLEX_E_CPS4 CPS4.
HECMW_ABLEX_E_CPS4I CPS4I.
HECMW_ABLEX_E_CPS4R CPS4R.
HECMW_ABLEX_E_CPS6 CPS6.
HECMW_ABLEX_E_CPS8 CPS8.
HECMW_ABLEX_E_CPS8R CPS8R.
HECMW_ABLEX_E_DC1D2 DC1D2.
HECMW_ABLEX_E_DC1D3 DC1D3.
HECMW_ABLEX_E_DC2D3 DC2D3.
HECMW_ABLEX_E_DC2D4 DC2D4.
HECMW_ABLEX_E_DC2D6 DC2D6.
HECMW_ABLEX_E_DC2D8 DC2D8.
HECMW_ABLEX_E_DC3D4 DC3D4.
HECMW_ABLEX_E_DC3D6 DC3D6.
HECMW_ABLEX_E_DC3D8 DC3D8.
HECMW_ABLEX_E_DC3D10 DC3D10.
HECMW_ABLEX_E_DC3D15 DC3D15.
HECMW_ABLEX_E_DC3D20 DC3D20.
HECMW_ABLEX_E_DCAX3 DCAX3.
HECMW_ABLEX_E_DCAX4 DCAX4.
HECMW_ABLEX_E_DCAX6 DCAX6.
HECMW_ABLEX_E_DCAX8 DCAX8.
HECMW_ABLEX_E_DINTER4 DINTER4.
HECMW_ABLEX_E_DINTER8 DINTER8.
HECMW_ABLEX_E_DS4 DS4.
HECMW_ABLEX_E_DS8 DS8.
HECMW_ABLEX_E_INTER4 INTER4.

HECMW_ABLEX_E_INTER8 INTER8.

HECMW_ABLEX_E_S3R S3R.

HECMW_ABLEX_E_S4R S4R.

HECMW_ABLEX_E_S8R S8R.

HECMW_ABLEX_E_T3D2 T3D2.

HECMW_ABLEX_E_T3D3 T3D3.

関数

double HECMW_ablex_get_number (void)

直前に読み込んだトークンが数値であった場合、その値を返す

戻り値:

直前に読み込まれた数値

char* HECMW_ablex_get_text (void)

直前に読み込んだトークンの文字列表記を返す

戻り値:

直前に読み込まれたトークンの文字列表記

int HECMW_ablex_get_lineno (void)

読み込んでいるファイルの現在の行番号を返す

戻り値:

行番号

int HECMW_ablex_next_token (void)

次のトークンを返す

EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_ablex_next_token_skip (int *skip_token*)

次のトークンを返す

*skip_token*で指定されたトークンの連続は読み飛ばされる。それ以外のトークンが現れたらそれを返す。 EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_ablex_set_input (FILE * *fp*)

字句解析ルーチンで読み込むファイルのファイルポインタをセットする

引数:

fp 字句解析対象のファイルへのファイルポインタ

戻り値:

成功すれば0を返す。 引数が無効なら-1を返す。

int HECMW_ablex_skip_line (void)

現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す

戻り値:

現在の行の最後のトークン番号。EOFに達した場合は0を返す。

int HECMW_ablex_switch_to_include (const char * *filename*)

現在読み込み中の字句解析ルーチンを中断し、新たなファイル(インクルードファイル)を解析対象とする

解析中のファイルの情報は保存される。インクルードファイルへの切替えは、この関数の呼び出し直後から有効になり、次に返されるトークンはインクルードファイルのものとなる。

インクルードファイルのトークンがなくなると、自動的に保存されていた状態に切り替わり、前のファイルの解析が再開される。インクルードファイルの解析が終了したことを通知することはないが、**HECMW_ablex_is_including()**によって、現在インクルードファイルを読んでいるのか否かを知ることができる。

引数:

filename インクルードファイル名

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_ablex_unput_token (void)

直前に読み込んだトークンを押し戻す

次回取得したトークンは、押し戻したトークンとなる。

戻り値:

常に0を返す.

int HECMW_ablex_is_including (void)

現在解析中のファイルがインクルードファイルか否かを返す

戻り値:

インクルードファイル解析中なら1を返し、そうでなければ0を返す

hecmw_bit_array.h

ビット配列

データ構造

- `struct hecmw_bit_array`
ビット配列構造体

関数

- `int HECMW_bit_array_init (struct hecmw_bit_array *ba, int len)`
ビット配列を初期化する
- `void HECMW_bit_array_finalize (struct hecmw_bit_array *ba)`
ビット配列のメモリ領域を解放する
- `int HECMW_bit_array_len (struct hecmw_bit_array *ba)`
ビット配列の長さを取得する
- `int HECMW_bit_array_set (struct hecmw_bit_array *ba, int index)`
ビット配列の`index`番目のビットを1にする
- `int HECMW_bit_array_get (struct hecmw_bit_array *ba, int index)`
ビット配列の`index`番目のビットを取得する
- `int HECMW_bit_array_set_all (struct hecmw_bit_array *ba)`
ビット配列のすべてのビットを1にする

- **void HECMW_bit_array_unset** (struct hecmw_bit_array *ba, int index)

ビット配列のindex番目のビットを0にする

説明

ビット配列

日付:

2007年11月22日

作者:

後藤 和哉

関数

int HECMW_bit_array_init (struct hecmw_bit_array *ba, int len)

ビット配列を初期化する

引数:

ba ビット配列

len 長さ

戻り値:

常に1を返す.

void HECMW_bit_array_finalize (struct hecmw_bit_array *ba)

ビット配列のメモリ領域を解放する

引数:

ba ビット配列

戻り値:

常に1を返す.

int HECMW_bit_array_len (struct hecmw_bit_array *ba)

ビット配列の長さを取得する

引数:

ba ビット配列

戻り値:

常に1を返す.

int HECMW_bit_array_set (struct hecmw_bit_array *ba, int index)

ビット配列のindex番目のビットを1にする

引数:

ba ビット配列

index インデックス

戻り値:

常に1を返す.

int HECMW_bit_array_get (struct hecmw_bit_array *ba, int index)

ビット配列のindex番目のビットを取得する

引数:

ba ビット配列

index インデックス

戻り値:

常に1を返す.

int HECMW_bit_array_set_all (struct hecmw_bit_array *ba)

ビット配列のすべてのビットを1にする

引数:

ba ビット配列

戻り値:

常に1を返す.

void HECMW_bit_array_unset (struct hecmw_bit_array *ba, int index)

ビット配列のindex番目のビットを0にする

引数:

ba ビット配列

index インデックス

戻り値:

常に1を返す.

hecmw_comm.h

コミュニケーター関連

```
#include "hecmw_config.h"
```

関数

- int **HECMW_Barrier** (**HECMW_Comm** comm)
- int **HECMW_Wait** (**HECMW_Request** *request, **HECMW_Status** *status)
- int **HECMW_Waitall** (int count, **HECMW_Request** *array_of_requests, **HECMW_Status** *array_of_statuses)
- int **HECMW_Bcast** (void *buffer, int count, **HECMW_Datatype** datatype, int root, **HECMW_Comm** comm)
- int **HECMW_Send** (void *buffer, int count, **HECMW_Datatype** datatype, int dest, int tag, **HECMW_Comm** comm)
- int **HECMW_Recv** (void *buffer, int count, **HECMW_Datatype** datatype, int source, int tag, **HECMW_Comm** comm, **HECMW_Status** *status)
- int **HECMW_Isend** (void *buffer, int count, **HECMW_Datatype** datatype, int dest, int tag, **HECMW_Comm** comm, **HECMW_Request** *request)
- int **HECMW_Irecv** (void *buffer, int count, **HECMW_Datatype** datatype, int source, int tag, **HECMW_Comm** comm, **HECMW_Request** *request)
- int **HECMW_Allreduce** (void *sendbuf, void *recvbuf, int count, **HECMW_Datatype** datatype, **HECMW_Op** op, **HECMW_Comm** comm)
- int **HECMW_Allgather** (void *sendbuf, int sendcount, **HECMW_Datatype** sendtype, void *recvbuf, int recvcount, **HECMW_Datatype** recvttype, **HECMW_Comm** comm)
- int **HECMW_Group_incl** (**HECMW_Group** group, int n, int *ranks, **HECMW_Group** *newgroup)
- int **HECMW_Group_excl** (**HECMW_Group** group, int n, int *ranks, **HECMW_Group** *newgroup)
- int **HECMW_Comm_create** (**HECMW_Comm** comm, **HECMW_Group** group, **HECMW_Comm** *comm_out)
- int **HECMW_Group_rank** (**HECMW_Group** group, int *rank)
- int **HECMW_Group_size** (**HECMW_Group** group, int *size)

- **int HECMW_Comm_group (HECMW_Comm comm, HECMW_Group *group)**
- **int HECMW_comm_init (int *argc, char ***argv)**
 コミュニケータ初期化処理を行う
- **HECMW_Comm HECMW_comm_get_comm (void)**
 コミュニケータを返す
- **int HECMW_comm_get_size (void)**
 総領域数を返す
- **int HECMW_comm_get_rank (void)**
 ランク番号を返す
- **HECMW_Group HECMW_comm_get_group (void)**
 MPI グループを返す.
- **HECMW_Fint HECMW_comm_c2f (HECMW_Comm comm)**
 C のコミュニケータをFortran のコミュニケータに変換する
- **HECMW_Comm HECMW_comm_f2c (HECMW_Fint comm)**
 Fortran のコミュニケータをC のコミュニケータに変換する
- **int HECMW_comm_is_initialized (void)**
 コミュニケータ初期化が行われているか否かを返す

説明

コミュニケータ関連

MPI 関数を使用してコミュニケータ等の初期化を行う。取得した情報をいつでも得られるように、アクセス関数を用意する。C においては、MPI はできるだ

け隠蔽されるので、情報が欲しい場合はこれらのラッパー関数を使用することになる。

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_Barrier (HECMW_Comm *comm*)

int HECMW_Wait (HECMW_Request * *request*, HECMW_Status * *statuse*)

**int HECMW_Waitall (int *count*, HECMW_Request * *array_of_requests*,
HECMW_Status * *array_of_statuses*)**

**int HECMW_Bcast (void * *buffer*, int *count*, HECMW_Datatype *datatype*, int *root*,
HECMW_Comm *comm*)**

**int HECMW_Send (void * *buffer*, int *count*, HECMW_Datatype *datatype*, int *dest*,
int *tag*, HECMW_Comm *comm*)**

**int HECMW_Recv (void * *buffer*, int *count*, HECMW_Datatype *datatype*, int *source*,
int *tag*, HECMW_Comm *comm*, HECMW_Status * *status*)**

**int HECMW_Isend (void * *buffer*, int *count*, HECMW_Datatype *datatype*, int *dest*,
int *tag*, HECMW_Comm *comm*, HECMW_Request * *request*)**

**int HECMW_Irecv (void * *buffer*, int *count*, HECMW_Datatype *datatype*, int *source*,
int *tag*, HECMW_Comm *comm*, HECMW_Request * *request*)**

**int HECMW_Allreduce (void * *sendbuf*, void * *recvbuf*, int *count*,
HECMW_Datatype *datatype*, HECMW_Op *op*, HECMW_Comm *comm*)**

**int HECMW_Allgather (void * *sendbuf*, int *sendcount*, HECMW_Datatype *sendtype*,
void * *recvbuf*, int *recvcount*, HECMW_Datatype *recvtype*, HECMW_Comm *comm*)**

**int HECMW_Group_incl (HECMW_Group *group*, int *n*, int * *ranks*, HECMW_Group
* *newgroup*)**

**int HECMW_Group_excl (HECMW_Group *group*, int *n*, int * *ranks*, HECMW_Group
* *newgroup*)**

**int HECMW_Comm_create (HECMW_Comm *comm*, HECMW_Group *group*,
HECMW_Comm * *comm_out*)**

int HECMW_Group_rank (HECMW_Group *group*, int * *rank*)

int HECMW_Group_size (HECMW_Group *group*, int * *size*)

int HECMW_Comm_group (HECMW_Comm *comm*, HECMW_Group * *group*)

int HECMW_comm_init (int * *argc*, char * *argv*)**

コミュニケーター初期化処理を行う

引数:

argc main()の*argc*引数へのポインタ

argv main()の*argv*引数へのポインタ

HECMW_Comm HECMW_comm_get_comm (void)

コミュニケーターを返す

戻り値:

MPIコミュニケーター

int HECMW_comm_get_size (void)

総領域数を返す

int HECMW_comm_get_rank (void)

ランク番号を返す

HECMW_Group HECMW_comm_get_group (void)

MPIグループを返す.

戻り値:

MPIグループ

HECMW_Fint HECMW_comm_c2f (HECMW_Comm comm)

CのコミュニケータをFortranのコミュニケータに変換する

引数:

comm Cのコミュニケータ

戻り値:

Fortranのコミュニケータを返す.

HECMW_Comm HECMW_comm_f2c (HECMW_Fint comm)

FortranのコミュニケータをCのコミュニケータに変換する

引数:

comm Fortranのコミュニケータ

戻り値:

Cのコミュニケータを返す.

int HECMW_comm_is_initialized (void)

コミュニケータ初期化が行われているか否かを返す

戻り値:

初期化されていれば1を返し，そうでなければ0を返す.

hecmw_common.h

commonディレクトリ内のヘッダをインクルード

```
#include "hecmw_common_define.h"
#include "hecmw_etype.h"
#include "hecmw_debug_write_dist.h"
#include "hecmw_reorder.h"
```

説明

common ディレクトリ内のヘッダをインクルード
削除予定

日付:

2004年2月16日

作者:

坂根 一彰

hecmw_common_define.h

要素情報の定数定義

マクロ定義

- #define **HECMW_COMMON_E_ALLOCATION** 1111111
- #define **HECMW_COMMON_E_OUT_OF_RANGE** 1111112
- #define **HECMW_COMMON_W_NO_EQN_BLOCK** 1111113
- #define **HECMW_ETYPE_MAX** 966
- #define **HECMW_ETYPE_ROD1** 111
- #define **HECMW_ETYPE_ROD2** 112
- #define **HECMW_ETYPE_TRI1** 231
- #define **HECMW_ETYPE_TRI2** 232
- #define **HECMW_ETYPE_QUA1** 241
- #define **HECMW_ETYPE_QUA2** 242
- #define **HECMW_ETYPE_TET1** 341
- #define **HECMW_ETYPE_TET2** 342
- #define **HECMW_ETYPE_PRI1** 351
- #define **HECMW_ETYPE_PRI2** 352
- #define **HECMW_ETYPE_HEX1** 361
- #define **HECMW_ETYPE_HEX2** 362
- #define **HECMW_ETYPE_PYR1** 371
- #define **HECMW_ETYPE_PYR2** 372
- #define **HECMW_ETYPE_MST1** 431
- #define **HECMW_ETYPE_MST2** 432
- #define **HECMW_ETYPE_MSQ1** 441
- #define **HECMW_ETYPE_MSQ2** 442
- #define **HECMW_ETYPE_JTT1** 531
- #define **HECMW_ETYPE_JTT2** 532
- #define **HECMW_ETYPE_JTQ1** 541
- #define **HECMW_ETYPE_JTQ2** 542
- #define **HECMW_ETYPE_BEM1** 611

- #define **HECMW_ETYPE_BEM2** 612
- #define **HECMW_ETYPE_SHT1** 731
- #define **HECMW_ETYPE_SHT2** 732
- #define **HECMW_ETYPE_SHQ1** 741
- #define **HECMW_ETYPE_SHQ2** 742
- #define **HECMW_ETYPE_LN11** 911
- #define **HECMW_ETYPE_LN12** 912
- #define **HECMW_ETYPE_LN13** 913
- #define **HECMW_ETYPE_LN14** 914
- #define **HECMW_ETYPE_LN15** 915
- #define **HECMW_ETYPE_LN16** 916
- #define **HECMW_ETYPE_LN21** 921
- #define **HECMW_ETYPE_LN22** 922
- #define **HECMW_ETYPE_LN23** 923
- #define **HECMW_ETYPE_LN24** 924
- #define **HECMW_ETYPE_LN25** 925
- #define **HECMW_ETYPE_LN26** 926
- #define **HECMW_ETYPE_LN31** 931
- #define **HECMW_ETYPE_LN32** 932
- #define **HECMW_ETYPE_LN33** 933
- #define **HECMW_ETYPE_LN34** 934
- #define **HECMW_ETYPE_LN35** 935
- #define **HECMW_ETYPE_LN36** 936
- #define **HECMW_ETYPE_LN41** 941
- #define **HECMW_ETYPE_LN42** 942
- #define **HECMW_ETYPE_LN43** 943
- #define **HECMW_ETYPE_LN44** 944
- #define **HECMW_ETYPE_LN45** 945
- #define **HECMW_ETYPE_LN46** 946
- #define **HECMW_ETYPE_LN51** 951
- #define **HECMW_ETYPE_LN52** 952
- #define **HECMW_ETYPE_LN53** 953
- #define **HECMW_ETYPE_LN54** 954
- #define **HECMW_ETYPE_LN55** 955
- #define **HECMW_ETYPE_LN56** 956
- #define **HECMW_ETYPE_LN61** 961

- #define **HECMW_ETYPE_LN62** 962
- #define **HECMW_ETYPE_LN63** 963
- #define **HECMW_ETYPE_LN64** 964
- #define **HECMW_ETYPE_LN65** 965
- #define **HECMW_ETYPE_LN66** 966
- #define **HECMW_GEOFEM_ETYPE_MAX** 722
- #define **HECMW_GEOFEM_ETYPE_ROD1** 111
- #define **HECMW_GEOFEM_ETYPE_ROD2** 112
- #define **HECMW_GEOFEM_ETYPE_TRI1** 211
- #define **HECMW_GEOFEM_ETYPE_TRI2** 212
- #define **HECMW_GEOFEM_ETYPE_QUA1** 221
- #define **HECMW_GEOFEM_ETYPE_QUA2** 222
- #define **HECMW_GEOFEM_ETYPE_TET1** 311
- #define **HECMW_GEOFEM_ETYPE_TET2** 312
- #define **HECMW_GEOFEM_ETYPE_PRI1** 321
- #define **HECMW_GEOFEM_ETYPE_PRI2** 322
- #define **HECMW_GEOFEM_ETYPE_HEX1** 331
- #define **HECMW_GEOFEM_ETYPE_HEX2** 332
- #define **HECMW_GEOFEM_ETYPE_MST1** 411
- #define **HECMW_GEOFEM_ETYPE_MST2** 412
- #define **HECMW_GEOFEM_ETYPE_MSQ1** 421
- #define **HECMW_GEOFEM_ETYPE_MSQ2** 422
- #define **HECMW_GEOFEM_ETYPE_JTT1** 511
- #define **HECMW_GEOFEM_ETYPE_JTT2** 512
- #define **HECMW_GEOFEM_ETYPE_JTQ1** 521
- #define **HECMW_GEOFEM_ETYPE_JTQ2** 522
- #define **HECMW_GEOFEM_ETYPE_BEM1** 611
- #define **HECMW_GEOFEM_ETYPE_BEM2** 612
- #define **HECMW_GEOFEM_ETYPE_SHT1** 711
- #define **HECMW_GEOFEM_ETYPE_SHT2** 712
- #define **HECMW_GEOFEM_ETYPE_SHQ1** 721
- #define **HECMW_GEOFEM_ETYPE_SHQ2** 722
- #define **HECMW_MESH_ETYPE_MAX** 64
- #define **HECMW_MESH_ETYPE_PNT** 0
- #define **HECMW_MESH_ETYPE_ROD1** 1
- #define **HECMW_MESH_ETYPE_ROD2** 2

- #define **HECMW_MESH_ETYPE_TRI1** 3
- #define **HECMW_MESH_ETYPE_TRI2** 4
- #define **HECMW_MESH_ETYPE_QUA1** 5
- #define **HECMW_MESH_ETYPE_QUA2** 6
- #define **HECMW_MESH_ETYPE_TET1** 7
- #define **HECMW_MESH_ETYPE_TET2** 8
- #define **HECMW_MESH_ETYPE_PRI1** 9
- #define **HECMW_MESH_ETYPE_PRI2** 10
- #define **HECMW_MESH_ETYPE_HEX1** 11
- #define **HECMW_MESH_ETYPE_HEX2** 12
- #define **HECMW_MESH_ETYPE_PYR1** 13
- #define **HECMW_MESH_ETYPE_PYR2** 14
- #define **HECMW_MESH_ETYPE_MST1** 15
- #define **HECMW_MESH_ETYPE_MST2** 16
- #define **HECMW_MESH_ETYPE_MSQ1** 17
- #define **HECMW_MESH_ETYPE_MSQ2** 18
- #define **HECMW_MESH_ETYPE_JTT1** 19
- #define **HECMW_MESH_ETYPE_JTT2** 20
- #define **HECMW_MESH_ETYPE_JTQ1** 21
- #define **HECMW_MESH_ETYPE_JTQ2** 22
- #define **HECMW_MESH_ETYPE_BEM1** 23
- #define **HECMW_MESH_ETYPE_BEM2** 24
- #define **HECMW_MESH_ETYPE_SHT1** 25
- #define **HECMW_MESH_ETYPE_SHT2** 26
- #define **HECMW_MESH_ETYPE_SHQ1** 27
- #define **HECMW_MESH_ETYPE_SHQ2** 28
- #define **HECMW_MESH_ETYPE_LN11** 29
- #define **HECMW_MESH_ETYPE_LN12** 30
- #define **HECMW_MESH_ETYPE_LN13** 31
- #define **HECMW_MESH_ETYPE_LN14** 32
- #define **HECMW_MESH_ETYPE_LN15** 33
- #define **HECMW_MESH_ETYPE_LN16** 34
- #define **HECMW_MESH_ETYPE_LN21** 35
- #define **HECMW_MESH_ETYPE_LN22** 36
- #define **HECMW_MESH_ETYPE_LN23** 37
- #define **HECMW_MESH_ETYPE_LN24** 38

- #define **HECMW_MESH_ETYPE_LN25** 39
- #define **HECMW_MESH_ETYPE_LN26** 40
- #define **HECMW_MESH_ETYPE_LN31** 41
- #define **HECMW_MESH_ETYPE_LN32** 42
- #define **HECMW_MESH_ETYPE_LN33** 43
- #define **HECMW_MESH_ETYPE_LN34** 44
- #define **HECMW_MESH_ETYPE_LN35** 45
- #define **HECMW_MESH_ETYPE_LN36** 46
- #define **HECMW_MESH_ETYPE_LN41** 47
- #define **HECMW_MESH_ETYPE_LN42** 48
- #define **HECMW_MESH_ETYPE_LN43** 49
- #define **HECMW_MESH_ETYPE_LN44** 50
- #define **HECMW_MESH_ETYPE_LN45** 51
- #define **HECMW_MESH_ETYPE_LN46** 52
- #define **HECMW_MESH_ETYPE_LN51** 53
- #define **HECMW_MESH_ETYPE_LN52** 54
- #define **HECMW_MESH_ETYPE_LN53** 55
- #define **HECMW_MESH_ETYPE_LN54** 56
- #define **HECMW_MESH_ETYPE_LN55** 57
- #define **HECMW_MESH_ETYPE_LN56** 58
- #define **HECMW_MESH_ETYPE_LN61** 59
- #define **HECMW_MESH_ETYPE_LN62** 60
- #define **HECMW_MESH_ETYPE_LN63** 61
- #define **HECMW_MESH_ETYPE_LN64** 62
- #define **HECMW_MESH_ETYPE_LN65** 63
- #define **HECMW_MESH_ETYPE_LN66** 64
- #define **HECMW_UCD_LABEL_PNT** "pt"
- #define **HECMW_UCD_LABEL_ROD1** "line"
- #define **HECMW_UCD_LABEL_ROD2** "line2"
- #define **HECMW_UCD_LABEL_TRI1** "tri"
- #define **HECMW_UCD_LABEL_TRI2** "tri2"
- #define **HECMW_UCD_LABEL_QUA1** "quad"
- #define **HECMW_UCD_LABEL_QUA2** "quad2"
- #define **HECMW_UCD_LABEL_TET1** "tet"
- #define **HECMW_UCD_LABEL_TET2** "tet2"
- #define **HECMW_UCD_LABEL_PRI1** "prism"

- #define **HECMW_UCD_LABEL_PRI2** "prism2"
- #define **HECMW_UCD_LABEL_HEX1** "hex"
- #define **HECMW_UCD_LABEL_HEX2** "hex2"
- #define **HECMW_UCD_LABEL_PYR1** "pyr"
- #define **HECMW_UCD_LABEL_PYR2** "pyr2"
- #define **HECMW_UCD_LABEL_MST1** "tet"
- #define **HECMW_UCD_LABEL_MST2** "tet2"
- #define **HECMW_UCD_LABEL_MSQ1** "pyr"
- #define **HECMW_UCD_LABEL_MSQ2** "pyr2"
- #define **HECMW_UCD_LABEL_JTT1** "pri"
- #define **HECMW_UCD_LABEL_JTT2** "pri2"
- #define **HECMW_UCD_LABEL_JTQ1** "hex"
- #define **HECMW_UCD_LABEL_JTQ2** "hex2"
- #define **HECMW_UCD_LABEL_BEM1** "line"
- #define **HECMW_UCD_LABEL_BEM2** "line2"
- #define **HECMW_UCD_LABEL_SHT1** "tri"
- #define **HECMW_UCD_LABEL_SHT2** "tri2"
- #define **HECMW_UCD_LABEL_SHQ1** "quad"
- #define **HECMW_UCD_LABEL_SHQ2** "quad2"
- #define **HECMW_UCD_LABEL_LN11** "line"
- #define **HECMW_UCD_LABEL_LN12** "line"
- #define **HECMW_UCD_LABEL_LN13** "line"
- #define **HECMW_UCD_LABEL_LN14** "line"
- #define **HECMW_UCD_LABEL_LN15** "line"
- #define **HECMW_UCD_LABEL_LN16** "line"
- #define **HECMW_UCD_LABEL_LN21** "line"
- #define **HECMW_UCD_LABEL_LN22** "line"
- #define **HECMW_UCD_LABEL_LN23** "line"
- #define **HECMW_UCD_LABEL_LN24** "line"
- #define **HECMW_UCD_LABEL_LN25** "line"
- #define **HECMW_UCD_LABEL_LN26** "line"
- #define **HECMW_UCD_LABEL_LN31** "line"
- #define **HECMW_UCD_LABEL_LN32** "line"
- #define **HECMW_UCD_LABEL_LN33** "line"
- #define **HECMW_UCD_LABEL_LN34** "line"
- #define **HECMW_UCD_LABEL_LN35** "line"

- #define **HECMW_UCD_LABEL_LN36** "line"
- #define **HECMW_UCD_LABEL_LN41** "line"
- #define **HECMW_UCD_LABEL_LN42** "line"
- #define **HECMW_UCD_LABEL_LN43** "line"
- #define **HECMW_UCD_LABEL_LN44** "line"
- #define **HECMW_UCD_LABEL_LN45** "line"
- #define **HECMW_UCD_LABEL_LN46** "line"
- #define **HECMW_UCD_LABEL_LN51** "line"
- #define **HECMW_UCD_LABEL_LN52** "line"
- #define **HECMW_UCD_LABEL_LN53** "line"
- #define **HECMW_UCD_LABEL_LN54** "line"
- #define **HECMW_UCD_LABEL_LN55** "line"
- #define **HECMW_UCD_LABEL_LN56** "line"
- #define **HECMW_UCD_LABEL_LN61** "line"
- #define **HECMW_UCD_LABEL_LN62** "line"
- #define **HECMW_UCD_LABEL_LN63** "line"
- #define **HECMW_UCD_LABEL_LN64** "line"
- #define **HECMW_UCD_LABEL_LN65** "line"
- #define **HECMW_UCD_LABEL_LN66** "line"
- #define **HECMW_MAX_NODE_MAX** 20
- #define **HECMW_MAX_NODE_PNT** 1
- #define **HECMW_MAX_NODE_ROD1** 2
- #define **HECMW_MAX_NODE_ROD2** 3
- #define **HECMW_MAX_NODE_TRI1** 3
- #define **HECMW_MAX_NODE_TRI2** 6
- #define **HECMW_MAX_NODE_QUA1** 4
- #define **HECMW_MAX_NODE_QUA2** 8
- #define **HECMW_MAX_NODE_TET1** 4
- #define **HECMW_MAX_NODE_TET2** 10
- #define **HECMW_MAX_NODE_PRI1** 6
- #define **HECMW_MAX_NODE_PRI2** 15
- #define **HECMW_MAX_NODE_HEX1** 8
- #define **HECMW_MAX_NODE_HEX2** 20
- #define **HECMW_MAX_NODE_PYR1** 5
- #define **HECMW_MAX_NODE_PYR2** 13
- #define **HECMW_MAX_NODE_MST1** 4

- #define **HECMW_MAX_NODE_MST2** 7
- #define **HECMW_MAX_NODE_MSQ1** 5
- #define **HECMW_MAX_NODE_MSQ2** 9
- #define **HECMW_MAX_NODE_JTT1** 6
- #define **HECMW_MAX_NODE_JTT2** 12
- #define **HECMW_MAX_NODE_JTQ1** 8
- #define **HECMW_MAX_NODE_JTQ2** 16
- #define **HECMW_MAX_NODE_BEM1** 2
- #define **HECMW_MAX_NODE_BEM2** 3
- #define **HECMW_MAX_NODE_SHT1** 3
- #define **HECMW_MAX_NODE_SHT2** 6
- #define **HECMW_MAX_NODE_SHQ1** 4
- #define **HECMW_MAX_NODE_SHQ2** 8
- #define **HECMW_MAX_NODE_LN11** 2
- #define **HECMW_MAX_NODE_LN12** 2
- #define **HECMW_MAX_NODE_LN13** 2
- #define **HECMW_MAX_NODE_LN14** 2
- #define **HECMW_MAX_NODE_LN15** 2
- #define **HECMW_MAX_NODE_LN16** 2
- #define **HECMW_MAX_NODE_LN21** 2
- #define **HECMW_MAX_NODE_LN22** 2
- #define **HECMW_MAX_NODE_LN23** 2
- #define **HECMW_MAX_NODE_LN24** 2
- #define **HECMW_MAX_NODE_LN25** 2
- #define **HECMW_MAX_NODE_LN26** 2
- #define **HECMW_MAX_NODE_LN31** 2
- #define **HECMW_MAX_NODE_LN32** 2
- #define **HECMW_MAX_NODE_LN33** 2
- #define **HECMW_MAX_NODE_LN34** 2
- #define **HECMW_MAX_NODE_LN35** 2
- #define **HECMW_MAX_NODE_LN36** 2
- #define **HECMW_MAX_NODE_LN41** 2
- #define **HECMW_MAX_NODE_LN42** 2
- #define **HECMW_MAX_NODE_LN43** 2
- #define **HECMW_MAX_NODE_LN44** 2
- #define **HECMW_MAX_NODE_LN45** 2

- #define **HECMW_MAX_NODE_LN46** 2
- #define **HECMW_MAX_NODE_LN51** 2
- #define **HECMW_MAX_NODE_LN52** 2
- #define **HECMW_MAX_NODE_LN53** 2
- #define **HECMW_MAX_NODE_LN54** 2
- #define **HECMW_MAX_NODE_LN55** 2
- #define **HECMW_MAX_NODE_LN56** 2
- #define **HECMW_MAX_NODE_LN61** 2
- #define **HECMW_MAX_NODE_LN62** 2
- #define **HECMW_MAX_NODE_LN63** 2
- #define **HECMW_MAX_NODE_LN64** 2
- #define **HECMW_MAX_NODE_LN65** 2
- #define **HECMW_MAX_NODE_LN66** 2
- #define **HECMW_MAX_EDGE_MAX** 24
- #define **HECMW_MAX_EDGE_PNT** 0
- #define **HECMW_MAX_EDGE_ROD1** 1
- #define **HECMW_MAX_EDGE_ROD2** 2
- #define **HECMW_MAX_EDGE_TRI1** 3
- #define **HECMW_MAX_EDGE_TRI2** 6
- #define **HECMW_MAX_EDGE_QUA1** 4
- #define **HECMW_MAX_EDGE_QUA2** 8
- #define **HECMW_MAX_EDGE_TET1** 6
- #define **HECMW_MAX_EDGE_TET2** 12
- #define **HECMW_MAX_EDGE_PRI1** 9
- #define **HECMW_MAX_EDGE_PRI2** 18
- #define **HECMW_MAX_EDGE_HEX1** 12
- #define **HECMW_MAX_EDGE_HEX2** 24
- #define **HECMW_MAX_EDGE_PYR1** 8
- #define **HECMW_MAX_EDGE_PYR2** 16
- #define **HECMW_MAX_EDGE_MST1** 6
- #define **HECMW_MAX_EDGE_MST2** 9
- #define **HECMW_MAX_EDGE_MSQ1** 8
- #define **HECMW_MAX_EDGE_MSQ2** 12
- #define **HECMW_MAX_EDGE_JTT1** 9
- #define **HECMW_MAX_EDGE_JTT2** 15
- #define **HECMW_MAX_EDGE_JTQ1** 12

- #define **HECMW_MAX_EDGE_JTQ2** 18
- #define **HECMW_MAX_EDGE_BEM1** 1
- #define **HECMW_MAX_EDGE_BEM2** 2
- #define **HECMW_MAX_EDGE_SHT1** 3
- #define **HECMW_MAX_EDGE_SHT2** 6
- #define **HECMW_MAX_EDGE_SHQ1** 4
- #define **HECMW_MAX_EDGE_SHQ2** 8
- #define **HECMW_MAX_EDGE_LN11** 1
- #define **HECMW_MAX_EDGE_LN12** 1
- #define **HECMW_MAX_EDGE_LN13** 1
- #define **HECMW_MAX_EDGE_LN14** 1
- #define **HECMW_MAX_EDGE_LN15** 1
- #define **HECMW_MAX_EDGE_LN16** 1
- #define **HECMW_MAX_EDGE_LN21** 1
- #define **HECMW_MAX_EDGE_LN22** 1
- #define **HECMW_MAX_EDGE_LN23** 1
- #define **HECMW_MAX_EDGE_LN24** 1
- #define **HECMW_MAX_EDGE_LN25** 1
- #define **HECMW_MAX_EDGE_LN26** 1
- #define **HECMW_MAX_EDGE_LN31** 1
- #define **HECMW_MAX_EDGE_LN32** 1
- #define **HECMW_MAX_EDGE_LN33** 1
- #define **HECMW_MAX_EDGE_LN34** 1
- #define **HECMW_MAX_EDGE_LN35** 1
- #define **HECMW_MAX_EDGE_LN36** 1
- #define **HECMW_MAX_EDGE_LN41** 1
- #define **HECMW_MAX_EDGE_LN42** 1
- #define **HECMW_MAX_EDGE_LN43** 1
- #define **HECMW_MAX_EDGE_LN44** 1
- #define **HECMW_MAX_EDGE_LN45** 1
- #define **HECMW_MAX_EDGE_LN46** 1
- #define **HECMW_MAX_EDGE_LN51** 1
- #define **HECMW_MAX_EDGE_LN52** 1
- #define **HECMW_MAX_EDGE_LN53** 1
- #define **HECMW_MAX_EDGE_LN54** 1
- #define **HECMW_MAX_EDGE_LN55** 1

- #define **HECMW_MAX_EDGE_LN56** 1
- #define **HECMW_MAX_EDGE_LN61** 1
- #define **HECMW_MAX_EDGE_LN62** 1
- #define **HECMW_MAX_EDGE_LN63** 1
- #define **HECMW_MAX_EDGE_LN64** 1
- #define **HECMW_MAX_EDGE_LN65** 1
- #define **HECMW_MAX_EDGE_LN66** 1
- #define **HECMW_MAX_SURF_MAX** 6
- #define **HECMW_MAX_SURF_PNT** 0
- #define **HECMW_MAX_SURF_ROD1** 0
- #define **HECMW_MAX_SURF_ROD2** 0
- #define **HECMW_MAX_SURF_TRI1** 3
- #define **HECMW_MAX_SURF_TRI2** 3
- #define **HECMW_MAX_SURF_QUA1** 4
- #define **HECMW_MAX_SURF_QUA2** 4
- #define **HECMW_MAX_SURF_TET1** 4
- #define **HECMW_MAX_SURF_TET2** 4
- #define **HECMW_MAX_SURF_PRI1** 5
- #define **HECMW_MAX_SURF_PRI2** 5
- #define **HECMW_MAX_SURF_HEX1** 6
- #define **HECMW_MAX_SURF_HEX2** 6
- #define **HECMW_MAX_SURF_PYR1** 5
- #define **HECMW_MAX_SURF_PYR2** 5
- #define **HECMW_MAX_SURF_MST1** 1
- #define **HECMW_MAX_SURF_MST2** 1
- #define **HECMW_MAX_SURF_MSQ1** 1
- #define **HECMW_MAX_SURF_MSQ2** 1
- #define **HECMW_MAX_SURF_JTT1** 2
- #define **HECMW_MAX_SURF_JTT2** 2
- #define **HECMW_MAX_SURF_JTQ1** 2
- #define **HECMW_MAX_SURF_JTQ2** 2
- #define **HECMW_MAX_SURF_BEM1** 0
- #define **HECMW_MAX_SURF_BEM2** 0
- #define **HECMW_MAX_SURF_SHT1** 2
- #define **HECMW_MAX_SURF_SHT2** 2
- #define **HECMW_MAX_SURF_SHQ1** 2

- #define **HECMW_MAX_SURF_SHQ2** 2
- #define **HECMW_MAX_SURF_LN11** 0
- #define **HECMW_MAX_SURF_LN12** 0
- #define **HECMW_MAX_SURF_LN13** 0
- #define **HECMW_MAX_SURF_LN14** 0
- #define **HECMW_MAX_SURF_LN15** 0
- #define **HECMW_MAX_SURF_LN16** 0
- #define **HECMW_MAX_SURF_LN21** 0
- #define **HECMW_MAX_SURF_LN22** 0
- #define **HECMW_MAX_SURF_LN23** 0
- #define **HECMW_MAX_SURF_LN24** 0
- #define **HECMW_MAX_SURF_LN25** 0
- #define **HECMW_MAX_SURF_LN26** 0
- #define **HECMW_MAX_SURF_LN31** 0
- #define **HECMW_MAX_SURF_LN32** 0
- #define **HECMW_MAX_SURF_LN33** 0
- #define **HECMW_MAX_SURF_LN34** 0
- #define **HECMW_MAX_SURF_LN35** 0
- #define **HECMW_MAX_SURF_LN36** 0
- #define **HECMW_MAX_SURF_LN41** 0
- #define **HECMW_MAX_SURF_LN42** 0
- #define **HECMW_MAX_SURF_LN43** 0
- #define **HECMW_MAX_SURF_LN44** 0
- #define **HECMW_MAX_SURF_LN45** 0
- #define **HECMW_MAX_SURF_LN46** 0
- #define **HECMW_MAX_SURF_LN51** 0
- #define **HECMW_MAX_SURF_LN52** 0
- #define **HECMW_MAX_SURF_LN53** 0
- #define **HECMW_MAX_SURF_LN54** 0
- #define **HECMW_MAX_SURF_LN55** 0
- #define **HECMW_MAX_SURF_LN56** 0
- #define **HECMW_MAX_SURF_LN61** 0
- #define **HECMW_MAX_SURF_LN62** 0
- #define **HECMW_MAX_SURF_LN63** 0
- #define **HECMW_MAX_SURF_LN64** 0
- #define **HECMW_MAX_SURF_LN65** 0

- #define **HECMW_MAX_SURF_LN66** 0
- #define **HECMW_MAX_TSUF_MAX** 4
- #define **HECMW_MAX_TSUF_PNT** 0
- #define **HECMW_MAX_TSUF_ROD1** 0
- #define **HECMW_MAX_TSUF_ROD2** 0
- #define **HECMW_MAX_TSUF_TRI1** 0
- #define **HECMW_MAX_TSUF_TRI2** 0
- #define **HECMW_MAX_TSUF_QUA1** 0
- #define **HECMW_MAX_TSUF_QUA2** 0
- #define **HECMW_MAX_TSUF_TET1** 4
- #define **HECMW_MAX_TSUF_TET2** 4
- #define **HECMW_MAX_TSUF_PRI1** 2
- #define **HECMW_MAX_TSUF_PRI2** 2
- #define **HECMW_MAX_TSUF_HEX1** 0
- #define **HECMW_MAX_TSUF_HEX2** 0
- #define **HECMW_MAX_TSUF_PYR1** 4
- #define **HECMW_MAX_TSUF_PYR2** 4
- #define **HECMW_MAX_TSUF_MST1** 1
- #define **HECMW_MAX_TSUF_MST2** 1
- #define **HECMW_MAX_TSUF_MSQ1** 0
- #define **HECMW_MAX_TSUF_MSQ2** 0
- #define **HECMW_MAX_TSUF_JTT1** 2
- #define **HECMW_MAX_TSUF_JTT2** 2
- #define **HECMW_MAX_TSUF_JTQ1** 0
- #define **HECMW_MAX_TSUF_JTQ2** 0
- #define **HECMW_MAX_TSUF_BEM1** 0
- #define **HECMW_MAX_TSUF_BEM2** 0
- #define **HECMW_MAX_TSUF_SHT1** 2
- #define **HECMW_MAX_TSUF_SHT2** 2
- #define **HECMW_MAX_TSUF_SHQ1** 0
- #define **HECMW_MAX_TSUF_SHQ2** 0
- #define **HECMW_MAX_TSUF_LN11** 0
- #define **HECMW_MAX_TSUF_LN12** 0
- #define **HECMW_MAX_TSUF_LN13** 0
- #define **HECMW_MAX_TSUF_LN14** 0
- #define **HECMW_MAX_TSUF_LN15** 0

- #define **HECMW_MAX_TSUF_LN16** 0
- #define **HECMW_MAX_TSUF_LN21** 0
- #define **HECMW_MAX_TSUF_LN22** 0
- #define **HECMW_MAX_TSUF_LN23** 0
- #define **HECMW_MAX_TSUF_LN24** 0
- #define **HECMW_MAX_TSUF_LN25** 0
- #define **HECMW_MAX_TSUF_LN26** 0
- #define **HECMW_MAX_TSUF_LN31** 0
- #define **HECMW_MAX_TSUF_LN32** 0
- #define **HECMW_MAX_TSUF_LN33** 0
- #define **HECMW_MAX_TSUF_LN34** 0
- #define **HECMW_MAX_TSUF_LN35** 0
- #define **HECMW_MAX_TSUF_LN36** 0
- #define **HECMW_MAX_TSUF_LN41** 0
- #define **HECMW_MAX_TSUF_LN42** 0
- #define **HECMW_MAX_TSUF_LN43** 0
- #define **HECMW_MAX_TSUF_LN44** 0
- #define **HECMW_MAX_TSUF_LN45** 0
- #define **HECMW_MAX_TSUF_LN46** 0
- #define **HECMW_MAX_TSUF_LN51** 0
- #define **HECMW_MAX_TSUF_LN52** 0
- #define **HECMW_MAX_TSUF_LN53** 0
- #define **HECMW_MAX_TSUF_LN54** 0
- #define **HECMW_MAX_TSUF_LN55** 0
- #define **HECMW_MAX_TSUF_LN56** 0
- #define **HECMW_MAX_TSUF_LN61** 0
- #define **HECMW_MAX_TSUF_LN62** 0
- #define **HECMW_MAX_TSUF_LN63** 0
- #define **HECMW_MAX_TSUF_LN64** 0
- #define **HECMW_MAX_TSUF_LN65** 0
- #define **HECMW_MAX_TSUF_LN66** 0
- #define **HECMW_MAX_QSUF_MAX** 6
- #define **HECMW_MAX_QSUF_PNT** 0
- #define **HECMW_MAX_QSUF_ROD1** 0
- #define **HECMW_MAX_QSUF_ROD2** 0
- #define **HECMW_MAX_QSUF_TRI1** 0

- #define **HECMW_MAX_QSUF_TRI2** 0
- #define **HECMW_MAX_QSUF_QUA1** 0
- #define **HECMW_MAX_QSUF_QUA2** 0
- #define **HECMW_MAX_QSUF_TET1** 0
- #define **HECMW_MAX_QSUF_TET2** 0
- #define **HECMW_MAX_QSUF_PRI1** 3
- #define **HECMW_MAX_QSUF_PRI2** 3
- #define **HECMW_MAX_QSUF_HEX1** 6
- #define **HECMW_MAX_QSUF_HEX2** 6
- #define **HECMW_MAX_QSUF_PYR1** 1
- #define **HECMW_MAX_QSUF_PYR2** 1
- #define **HECMW_MAX_QSUF_MST1** 0
- #define **HECMW_MAX_QSUF_MST2** 0
- #define **HECMW_MAX_QSUF_MSQ1** 1
- #define **HECMW_MAX_QSUF_MSQ2** 1
- #define **HECMW_MAX_QSUF_JTT1** 0
- #define **HECMW_MAX_QSUF_JTT2** 0
- #define **HECMW_MAX_QSUF_JTQ1** 2
- #define **HECMW_MAX_QSUF_JTQ2** 2
- #define **HECMW_MAX_QSUF_BEM1** 0
- #define **HECMW_MAX_QSUF_BEM2** 0
- #define **HECMW_MAX_QSUF_SHT1** 0
- #define **HECMW_MAX_QSUF_SHT2** 0
- #define **HECMW_MAX_QSUF_SHQ1** 2
- #define **HECMW_MAX_QSUF_SHQ2** 2
- #define **HECMW_MAX_QSUF_LN11** 0
- #define **HECMW_MAX_QSUF_LN12** 0
- #define **HECMW_MAX_QSUF_LN13** 0
- #define **HECMW_MAX_QSUF_LN14** 0
- #define **HECMW_MAX_QSUF_LN15** 0
- #define **HECMW_MAX_QSUF_LN16** 0
- #define **HECMW_MAX_QSUF_LN21** 0
- #define **HECMW_MAX_QSUF_LN22** 0
- #define **HECMW_MAX_QSUF_LN23** 0
- #define **HECMW_MAX_QSUF_LN24** 0
- #define **HECMW_MAX_QSUF_LN25** 0

- #define **HECMW_MAX_QSUF_LN26** 0
- #define **HECMW_MAX_QSUF_LN31** 0
- #define **HECMW_MAX_QSUF_LN32** 0
- #define **HECMW_MAX_QSUF_LN33** 0
- #define **HECMW_MAX_QSUF_LN34** 0
- #define **HECMW_MAX_QSUF_LN35** 0
- #define **HECMW_MAX_QSUF_LN36** 0
- #define **HECMW_MAX_QSUF_LN41** 0
- #define **HECMW_MAX_QSUF_LN42** 0
- #define **HECMW_MAX_QSUF_LN43** 0
- #define **HECMW_MAX_QSUF_LN44** 0
- #define **HECMW_MAX_QSUF_LN45** 0
- #define **HECMW_MAX_QSUF_LN46** 0
- #define **HECMW_MAX_QSUF_LN51** 0
- #define **HECMW_MAX_QSUF_LN52** 0
- #define **HECMW_MAX_QSUF_LN53** 0
- #define **HECMW_MAX_QSUF_LN54** 0
- #define **HECMW_MAX_QSUF_LN55** 0
- #define **HECMW_MAX_QSUF_LN56** 0
- #define **HECMW_MAX_QSUF_LN61** 0
- #define **HECMW_MAX_QSUF_LN62** 0
- #define **HECMW_MAX_QSUF_LN63** 0
- #define **HECMW_MAX_QSUF_LN64** 0
- #define **HECMW_MAX_QSUF_LN65** 0
- #define **HECMW_MAX_QSUF_LN66** 0
- #define **HECMW_MESH_NDOFGRP_MAX** 3
- #define **HECMW_MESH_DOF_MAX** 6
- #define **HECMW_MESH_DOF_TOT** 3
- #define **HECMW_MESH_DOF_TWO** 2
- #define **HECMW_MESH_DOF_THREE** 3
- #define **HECMW_MESH_DOF_SIX** 6

説明

要素情報の定数定義

日付:

2004年2月16日

作者:

坂根 一彰

マクロ定義

```
#define HECMW_COMMON_E_ALLOCATION 1111111
```

```
#define HECMW_COMMON_E_OUT_OF_RANGE 1111112
```

```
#define HECMW_COMMON_W_NO_EQN_BLOCK 1111113
```

```
#define HECMW_ETYPE_MAX 966
```

```
#define HECMW_ETYPE_ROD1 111
```

```
#define HECMW_ETYPE_ROD2 112
```

```
#define HECMW_ETYPE_TRI1 231
```

```
#define HECMW_ETYPE_TRI2 232
```


#define HECMW_ETYPE_QUA1 241

#define HECMW_ETYPE_QUA2 242

#define HECMW_ETYPE_TET1 341

#define HECMW_ETYPE_TET2 342

#define HECMW_ETYPE_PRI1 351

#define HECMW_ETYPE_PRI2 352

#define HECMW_ETYPE_HEX1 361

#define HECMW_ETYPE_HEX2 362

#define HECMW_ETYPE_PYR1 371

#define HECMW_ETYPE_PYR2 372

#define HECMW_ETYPE_MST1 431

#define HECMW_ETYPE_MST2 432

#define HECMW_ETYPE_MSQ1 441

#define HECMW_ETYPE_MSQ2 442

#define HECMW_ETYPE_JTT1 531

#define HECMW_ETYPE_JTT2 532

#define HECMW_ETYPE_JTQ1 541

#define HECMW_ETYPE_JTQ2 542

#define HECMW_ETYPE_BEM1 611

#define HECMW_ETYPE_BEM2 612

#define HECMW_ETYPE_SHT1 731

#define HECMW_ETYPE_SHT2 732

#define HECMW_ETYPE_SHQ1 741

#define HECMW_ETYPE_SHQ2 742

#define HECMW_ETYPE_LN11 911

#define HECMW_ETYPE_LN12 912

#define HECMW_ETYPE_LN13 913

#define HECMW_ETYPE_LN14 914

#define HECMW_ETYPE_LN15 915

#define HECMW_ETYPE_LN16 916

#define HECMW_ETYPE_LN21 921

#define HECMW_ETYPE_LN22 922

#define HECMW_ETYPE_LN23 923

#define HECMW_ETYPE_LN24 924

#define HECMW_ETYPE_LN25 925

#define HECMW_ETYPE_LN26 926

#define HECMW_ETYPE_LN31 931

#define HECMW_ETYPE_LN32 932

#define HECMW_ETYPE_LN33 933

#define HECMW_ETYPE_LN34 934

#define HECMW_ETYPE_LN35 935

#define HECMW_ETYPE_LN36 936

#define HECMW_ETYPE_LN41 941

#define HECMW_ETYPE_LN42 942

#define HECMW_ETYPE_LN43 943

#define HECMW_ETYPE_LN44 944

#define HECMW_ETYPE_LN45 945

#define HECMW_ETYPE_LN46 946

#define HECMW_ETYPE_LN51 951

#define HECMW_ETYPE_LN52 952

#define HECMW_ETYPE_LN53 953

#define HECMW_ETYPE_LN54 954

#define HECMW_ETYPE_LN55 955

#define HECMW_ETYPE_LN56 956

#define HECMW_ETYPE_LN61 961

#define HECMW_ETYPE_LN62 962

#define HECMW_ETYPE_LN63 963

#define HECMW_ETYPE_LN64 964

#define HECMW_ETYPE_LN65 965

#define HECMW_ETYPE_LN66 966

#define HECMW_GEOFEM_ETYPE_MAX 722

#define HECMW_GEOFEM_ETYPE_ROD1 111

#define HECMW_GEOFEM_ETYPE_ROD2 112

#define HECMW_GEOFEM_ETYPE_TRI1 211

#define HECMW_GEOFEM_ETYPE_TRI2 212

#define HECMW_GEOFEM_ETYPE_QUA1 221

#define HECMW_GEOFEM_ETYPE_QUA2 222

#define HECMW_GEOFEM_ETYPE_TET1 311

#define HECMW_GEOFEM_ETYPE_TET2 312

#define HECMW_GEOFEM_ETYPE_PRI1 321

#define HECMW_GEOFEM_ETYPE_PRI2 322

#define HECMW_GEOFEM_ETYPE_HEX1 331

#define HECMW_GEOFEM_ETYPE_HEX2 332

#define HECMW_GEOFEM_ETYPE_MST1 411

#define HECMW_GEOFEM_ETYPE_MST2 412

#define HECMW_GEOFEM_ETYPE_MSQ1 421

#define HECMW_GEOFEM_ETYPE_MSQ2 422

#define HECMW_GEOFEM_ETYPE_JTT1 511

#define HECMW_GEOFEM_ETYPE_JTT2 512

#define HECMW_GEOFEM_ETYPE_JTQ1 521

#define HECMW_GEOFEM_ETYPE_JTQ2 522

#define HECMW_GEOFEM_ETYPE_BEM1 611

#define HECMW_GEOFEM_ETYPE_BEM2 612

#define HECMW_GEOFEM_ETYPE_SHT1 711

#define HECMW_GEOFEM_ETYPE_SHT2 712

#define HECMW_GEOFEM_ETYPE_SHQ1 721

#define HECMW_GEOFEM_ETYPE_SHQ2 722

#define HECMW_MESH_ETYPE_MAX 64

#define HECMW_MESH_ETYPE_PNT 0

#define HECMW_MESH_ETYPE_ROD1 1

#define HECMW_MESH_ETYPE_ROD2 2

#define HECMW_MESH_ETYPE_TRI1 3

#define HECMW_MESH_ETYPE_TRI2 4

#define HECMW_MESH_ETYPE_QUA1 5

#define HECMW_MESH_ETYPE_QUA2 6

#define HECMW_MESH_ETYPE_TET1 7

#define HECMW_MESH_ETYPE_TET2 8

#define HECMW_MESH_ETYPE_PRI1 9

#define HECMW_MESH_ETYPE_PRI2 10

#define HECMW_MESH_ETYPE_HEX1 11

#define HECMW_MESH_ETYPE_HEX2 12

#define HECMW_MESH_ETYPE_PYR1 13

#define HECMW_MESH_ETYPE_PYR2 14

#define HECMW_MESH_ETYPE_MST1 15

#define HECMW_MESH_ETYPE_MST2 16

#define HECMW_MESH_ETYPE_MSQ1 17

#define HECMW_MESH_ETYPE_MSQ2 18

#define HECMW_MESH_ETYPE_JTT1 19

#define HECMW_MESH_ETYPE_JTT2 20

#define HECMW_MESH_ETYPE_JTQ1 21

#define HECMW_MESH_ETYPE_JTQ2 22

#define HECMW_MESH_ETYPE_BEM1 23

#define HECMW_MESH_ETYPE_BEM2 24

#define HECMW_MESH_ETYPE_SHT1 25

#define HECMW_MESH_ETYPE_SHT2 26

#define HECMW_MESH_ETYPE_SHQ1 27

#define HECMW_MESH_ETYPE_SHQ2 28

#define HECMW_MESH_ETYPE_LN11 29

#define HECMW_MESH_ETYPE_LN12 30

#define HECMW_MESH_ETYPE_LN13 31

#define HECMW_MESH_ETYPE_LN14 32

#define HECMW_MESH_ETYPE_LN15 33

#define HECMW_MESH_ETYPE_LN16 34

#define HECMW_MESH_ETYPE_LN21 35

#define HECMW_MESH_ETYPE_LN22 36

#define HECMW_MESH_ETYPE_LN23 37

#define HECMW_MESH_ETYPE_LN24 38

#define HECMW_MESH_ETYPE_LN25 39

#define HECMW_MESH_ETYPE_LN26 40

#define HECMW_MESH_ETYPE_LN31 41

#define HECMW_MESH_ETYPE_LN32 42

#define HECMW_MESH_ETYPE_LN33 43

#define HECMW_MESH_ETYPE_LN34 44

#define HECMW_MESH_ETYPE_LN35 45

#define HECMW_MESH_ETYPE_LN36 46

#define HECMW_MESH_ETYPE_LN41 47

#define HECMW_MESH_ETYPE_LN42 48

#define HECMW_MESH_ETYPE_LN43 49

#define HECMW_MESH_ETYPE_LN44 50

#define HECMW_MESH_ETYPE_LN45 51

#define HECMW_MESH_ETYPE_LN46 52

#define HECMW_MESH_ETYPE_LN51 53

#define HECMW_MESH_ETYPE_LN52 54

#define HECMW_MESH_ETYPE_LN53 55

#define HECMW_MESH_ETYPE_LN54 56

#define HECMW_MESH_ETYPE_LN55 57

#define HECMW_MESH_ETYPE_LN56 58

#define HECMW_MESH_ETYPE_LN61 59

#define HECMW_MESH_ETYPE_LN62 60

#define HECMW_MESH_ETYPE_LN63 61

#define HECMW_MESH_ETYPE_LN64 62

#define HECMW_MESH_ETYPE_LN65 63

#define HECMW_MESH_ETYPE_LN66 64

#define HECMW_UCD_LABEL_PNT "pt"

#define HECMW_UCD_LABEL_ROD1 "line"

#define HECMW_UCD_LABEL_ROD2 "line2"

#define HECMW_UCD_LABEL_TRI1 "tri"

#define HECMW_UCD_LABEL_TRI2 "tri2"

#define HECMW_UCD_LABEL_QUA1 "quad"

#define HECMW_UCD_LABEL_QUA2 "quad2"

#define HECMW_UCD_LABEL_TET1 "tet"

#define HECMW_UCD_LABEL_TET2 "tet2"

#define HECMW_UCD_LABEL_PRI1 "prism"

#define HECMW_UCD_LABEL_PRI2 "prism2"

#define HECMW_UCD_LABEL_HEX1 "hex"

#define HECMW_UCD_LABEL_HEX2 "hex2"

#define HECMW_UCD_LABEL_PYR1 "pyr"

#define HECMW_UCD_LABEL_PYR2 "pyr2"

#define HECMW_UCD_LABEL_MST1 "tet"

#define HECMW_UCD_LABEL_MST2 "tet2"

#define HECMW_UCD_LABEL_MSQ1 "pyr"

#define HECMW_UCD_LABEL_MSQ2 "pyr2"

#define HECMW_UCD_LABEL_JTT1 "pri"

#define HECMW_UCD_LABEL_JTT2 "pri2"

#define HECMW_UCD_LABEL_JTQ1 "hex"

#define HECMW_UCD_LABEL_JTQ2 "hex2"

#define HECMW_UCD_LABEL_BEM1 "line"

#define HECMW_UCD_LABEL_BEM2 "line2"

#define HECMW_UCD_LABEL_SHT1 "tri"

#define HECMW_UCD_LABEL_SHT2 "tri2"

#define HECMW_UCD_LABEL_SHQ1 "quad"

#define HECMW_UCD_LABEL_SHQ2 "quad2"

#define HECMW_UCD_LABEL_LN11 "line"

#define HECMW_UCD_LABEL_LN12 "line"

#define HECMW_UCD_LABEL_LN13 "line"

#define HECMW_UCD_LABEL_LN14 "line"

#define HECMW_UCD_LABEL_LN15 "line"

#define HECMW_UCD_LABEL_LN16 "line"

#define HECMW_UCD_LABEL_LN21 "line"

#define HECMW_UCD_LABEL_LN22 "line"

#define HECMW_UCD_LABEL_LN23 "line"

#define HECMW_UCD_LABEL_LN24 "line"

#define HECMW_UCD_LABEL_LN25 "line"

#define HECMW_UCD_LABEL_LN26 "line"

#define HECMW_UCD_LABEL_LN31 "line"

#define HECMW_UCD_LABEL_LN32 "line"

#define HECMW_UCD_LABEL_LN33 "line"

#define HECMW_UCD_LABEL_LN34 "line"

#define HECMW_UCD_LABEL_LN35 "line"

#define HECMW_UCD_LABEL_LN36 "line"

#define HECMW_UCD_LABEL_LN41 "line"

#define HECMW_UCD_LABEL_LN42 "line"

#define HECMW_UCD_LABEL_LN43 "line"

#define HECMW_UCD_LABEL_LN44 "line"

#define HECMW_UCD_LABEL_LN45 "line"

#define HECMW_UCD_LABEL_LN46 "line"

#define HECMW_UCD_LABEL_LN51 "line"

#define HECMW_UCD_LABEL_LN52 "line"

#define HECMW_UCD_LABEL_LN53 "line"

#define HECMW_UCD_LABEL_LN54 "line"

#define HECMW_UCD_LABEL_LN55 "line"

#define HECMW_UCD_LABEL_LN56 "line"

#define HECMW_UCD_LABEL_LN61 "line"

#define HECMW_UCD_LABEL_LN62 "line"

#define HECMW_UCD_LABEL_LN63 "line"

#define HECMW_UCD_LABEL_LN64 "line"

#define HECMW_UCD_LABEL_LN65 "line"

#define HECMW_UCD_LABEL_LN66 "line"

#define HECMW_MAX_NODE_MAX 20

#define HECMW_MAX_NODE_PNT 1

#define HECMW_MAX_NODE_ROD1 2

#define HECMW_MAX_NODE_ROD2 3

#define HECMW_MAX_NODE_TRI1 3

#define HECMW_MAX_NODE_TRI2 6

#define HECMW_MAX_NODE_QUA1 4

#define HECMW_MAX_NODE_QUA2 8

#define HECMW_MAX_NODE_TET1 4

#define HECMW_MAX_NODE_TET2 10

#define HECMW_MAX_NODE_PRI1 6

#define HECMW_MAX_NODE_PRI2 15

#define HECMW_MAX_NODE_HEX1 8

#define HECMW_MAX_NODE_HEX2 20

#define HECMW_MAX_NODE_PYR1 5

#define HECMW_MAX_NODE_PYR2 13

#define HECMW_MAX_NODE_MST1 4

#define HECMW_MAX_NODE_MST2 7

#define HECMW_MAX_NODE_MSQ1 5

#define HECMW_MAX_NODE_MSQ2 9

#define HECMW_MAX_NODE_JTT1 6

#define HECMW_MAX_NODE_JTT2 12

#define HECMW_MAX_NODE_JTQ1 8

#define HECMW_MAX_NODE_JTQ2 16

#define HECMW_MAX_NODE_BEM1 2

#define HECMW_MAX_NODE_BEM2 3

#define HECMW_MAX_NODE_SHT1 3

#define HECMW_MAX_NODE_SHT2 6

#define HECMW_MAX_NODE_SHQ1 4

#define HECMW_MAX_NODE_SHQ2 8

#define HECMW_MAX_NODE_LN11 2

#define HECMW_MAX_NODE_LN12 2

#define HECMW_MAX_NODE_LN13 2

#define HECMW_MAX_NODE_LN14 2

#define HECMW_MAX_NODE_LN15 2

#define HECMW_MAX_NODE_LN16 2

#define HECMW_MAX_NODE_LN21 2

#define HECMW_MAX_NODE_LN22 2

#define HECMW_MAX_NODE_LN23 2

#define HECMW_MAX_NODE_LN24 2

#define HECMW_MAX_NODE_LN25 2

#define HECMW_MAX_NODE_LN26 2

#define HECMW_MAX_NODE_LN31 2

#define HECMW_MAX_NODE_LN32 2

#define HECMW_MAX_NODE_LN33 2

#define HECMW_MAX_NODE_LN34 2

#define HECMW_MAX_NODE_LN35 2

#define HECMW_MAX_NODE_LN36 2

#define HECMW_MAX_NODE_LN41 2

#define HECMW_MAX_NODE_LN42 2

#define HECMW_MAX_NODE_LN43 2

#define HECMW_MAX_NODE_LN44 2

#define HECMW_MAX_NODE_LN45 2

#define HECMW_MAX_NODE_LN46 2

#define HECMW_MAX_NODE_LN51 2

#define HECMW_MAX_NODE_LN52 2

#define HECMW_MAX_NODE_LN53 2

#define HECMW_MAX_NODE_LN54 2

#define HECMW_MAX_NODE_LN55 2

#define HECMW_MAX_NODE_LN56 2

#define HECMW_MAX_NODE_LN61 2

#define HECMW_MAX_NODE_LN62 2

#define HECMW_MAX_NODE_LN63 2

#define HECMW_MAX_NODE_LN64 2

#define HECMW_MAX_NODE_LN65 2

#define HECMW_MAX_NODE_LN66 2

#define HECMW_MAX_EDGE_MAX 24

#define HECMW_MAX_EDGE_PNT 0

#define HECMW_MAX_EDGE_ROD1 1

#define HECMW_MAX_EDGE_ROD2 2

#define HECMW_MAX_EDGE_TRI1 3

#define HECMW_MAX_EDGE_TRI2 6

#define HECMW_MAX_EDGE_QUA1 4

#define HECMW_MAX_EDGE_QUA2 8

#define HECMW_MAX_EDGE_TET1 6

#define HECMW_MAX_EDGE_TET2 12

#define HECMW_MAX_EDGE_PRI1 9

#define HECMW_MAX_EDGE_PRI2 18

#define HECMW_MAX_EDGE_HEX1 12

#define HECMW_MAX_EDGE_HEX2 24

#define HECMW_MAX_EDGE_PYR1 8

#define HECMW_MAX_EDGE_PYR2 16

#define HECMW_MAX_EDGE_MST1 6

#define HECMW_MAX_EDGE_MST2 9

#define HECMW_MAX_EDGE_MSQ1 8

#define HECMW_MAX_EDGE_MSQ2 12

#define HECMW_MAX_EDGE_JTT1 9

#define HECMW_MAX_EDGE_JTT2 15

#define HECMW_MAX_EDGE_JTQ1 12

#define HECMW_MAX_EDGE_JTQ2 18

#define HECMW_MAX_EDGE_BEM1 1

#define HECMW_MAX_EDGE_BEM2 2

#define HECMW_MAX_EDGE_SHT1 3

#define HECMW_MAX_EDGE_SHT2 6

#define HECMW_MAX_EDGE_SHQ1 4

#define HECMW_MAX_EDGE_SHQ2 8

#define HECMW_MAX_EDGE_LN11 1

#define HECMW_MAX_EDGE_LN12 1

#define HECMW_MAX_EDGE_LN13 1

#define HECMW_MAX_EDGE_LN14 1

#define HECMW_MAX_EDGE_LN15 1

#define HECMW_MAX_EDGE_LN16 1

#define HECMW_MAX_EDGE_LN21 1

#define HECMW_MAX_EDGE_LN22 1

#define HECMW_MAX_EDGE_LN23 1

#define HECMW_MAX_EDGE_LN24 1

#define HECMW_MAX_EDGE_LN25 1

#define HECMW_MAX_EDGE_LN26 1

#define HECMW_MAX_EDGE_LN31 1

#define HECMW_MAX_EDGE_LN32 1

#define HECMW_MAX_EDGE_LN33 1

#define HECMW_MAX_EDGE_LN34 1

#define HECMW_MAX_EDGE_LN35 1

#define HECMW_MAX_EDGE_LN36 1

#define HECMW_MAX_EDGE_LN41 1

#define HECMW_MAX_EDGE_LN42 1

#define HECMW_MAX_EDGE_LN43 1

#define HECMW_MAX_EDGE_LN44 1

#define HECMW_MAX_EDGE_LN45 1

#define HECMW_MAX_EDGE_LN46 1

#define HECMW_MAX_EDGE_LN51 1

#define HECMW_MAX_EDGE_LN52 1

#define HECMW_MAX_EDGE_LN53 1

#define HECMW_MAX_EDGE_LN54 1

#define HECMW_MAX_EDGE_LN55 1

#define HECMW_MAX_EDGE_LN56 1

#define HECMW_MAX_EDGE_LN61 1

#define HECMW_MAX_EDGE_LN62 1

#define HECMW_MAX_EDGE_LN63 1

#define HECMW_MAX_EDGE_LN64 1

#define HECMW_MAX_EDGE_LN65 1

#define HECMW_MAX_EDGE_LN66 1

#define HECMW_MAX_SURF_MAX 6

#define HECMW_MAX_SURF_PNT 0

#define HECMW_MAX_SURF_ROD1 0

#define HECMW_MAX_SURF_ROD2 0

#define HECMW_MAX_SURF_TRI1 3

#define HECMW_MAX_SURF_TRI2 3

#define HECMW_MAX_SURF_QUA1 4

#define HECMW_MAX_SURF_QUA2 4

#define HECMW_MAX_SURF_TET1 4

#define HECMW_MAX_SURF_TET2 4

#define HECMW_MAX_SURF_PRI1 5

#define HECMW_MAX_SURF_PRI2 5

#define HECMW_MAX_SURF_HEX1 6

#define HECMW_MAX_SURF_HEX2 6

#define HECMW_MAX_SURF_PYR1 5

#define HECMW_MAX_SURF_PYR2 5

#define HECMW_MAX_SURF_MST1 1

#define HECMW_MAX_SURF_MST2 1

#define HECMW_MAX_SURF_MSQ1 1

#define HECMW_MAX_SURF_MSQ2 1

#define HECMW_MAX_SURF_JTT1 2

#define HECMW_MAX_SURF_JTT2 2

#define HECMW_MAX_SURF_JTQ1 2

#define HECMW_MAX_SURF_JTQ2 2

#define HECMW_MAX_SURF_BEM1 0

#define HECMW_MAX_SURF_BEM2 0

#define HECMW_MAX_SURF_SHT1 2

#define HECMW_MAX_SURF_SHT2 2

#define HECMW_MAX_SURF_SHQ1 2

#define HECMW_MAX_SURF_SHQ2 2

#define HECMW_MAX_SURF_LN11 0

#define HECMW_MAX_SURF_LN12 0

#define HECMW_MAX_SURF_LN13 0

#define HECMW_MAX_SURF_LN14 0

#define HECMW_MAX_SURF_LN15 0

#define HECMW_MAX_SURF_LN16 0

#define HECMW_MAX_SURF_LN21 0

#define HECMW_MAX_SURF_LN22 0

#define HECMW_MAX_SURF_LN23 0

#define HECMW_MAX_SURF_LN24 0

#define HECMW_MAX_SURF_LN25 0

#define HECMW_MAX_SURF_LN26 0

#define HECMW_MAX_SURF_LN31 0

#define HECMW_MAX_SURF_LN32 0

#define HECMW_MAX_SURF_LN33 0

#define HECMW_MAX_SURF_LN34 0

#define HECMW_MAX_SURF_LN35 0

#define HECMW_MAX_SURF_LN36 0

#define HECMW_MAX_SURF_LN41 0

#define HECMW_MAX_SURF_LN42 0

#define HECMW_MAX_SURF_LN43 0

#define HECMW_MAX_SURF_LN44 0

#define HECMW_MAX_SURF_LN45 0

#define HECMW_MAX_SURF_LN46 0

#define HECMW_MAX_SURF_LN51 0

#define HECMW_MAX_SURF_LN52 0

#define HECMW_MAX_SURF_LN53 0

#define HECMW_MAX_SURF_LN54 0

#define HECMW_MAX_SURF_LN55 0

#define HECMW_MAX_SURF_LN56 0

#define HECMW_MAX_SURF_LN61 0

#define HECMW_MAX_SURF_LN62 0

#define HECMW_MAX_SURF_LN63 0

#define HECMW_MAX_SURF_LN64 0

#define HECMW_MAX_SURF_LN65 0

#define HECMW_MAX_SURF_LN66 0

#define HECMW_MAX_TSUF_MAX 4

#define HECMW_MAX_TSUF_PNT 0

#define HECMW_MAX_TSUF_ROD1 0

#define HECMW_MAX_TSUF_ROD2 0

#define HECMW_MAX_TSUF_TRI1 0

#define HECMW_MAX_TSUF_TRI2 0

#define HECMW_MAX_TSUF_QUA1 0

#define HECMW_MAX_TSUF_QUA2 0

#define HECMW_MAX_TSUF_TET1 4

#define HECMW_MAX_TSUF_TET2 4

#define HECMW_MAX_TSUF_PRI1 2

#define HECMW_MAX_TSUF_PRI2 2

#define HECMW_MAX_TSUF_HEX1 0

#define HECMW_MAX_TSUF_HEX2 0

#define HECMW_MAX_TSUF_PYR1 4

#define HECMW_MAX_TSUF_PYR2 4

#define HECMW_MAX_TSUF_MST1 1

#define HECMW_MAX_TSUF_MST2 1

#define HECMW_MAX_TSUF_MSQ1 0

#define HECMW_MAX_TSUF_MSQ2 0

#define HECMW_MAX_TSUF_JTT1 2

#define HECMW_MAX_TSUF_JTT2 2

#define HECMW_MAX_TSUF_JTQ1 0

#define HECMW_MAX_TSUF_JTQ2 0

#define HECMW_MAX_TSUF_BEM1 0

#define HECMW_MAX_TSUF_BEM2 0

#define HECMW_MAX_TSUF_SHT1 2

#define HECMW_MAX_TSUF_SHT2 2

#define HECMW_MAX_TSUF_SHQ1 0

#define HECMW_MAX_TSUF_SHQ2 0

#define HECMW_MAX_TSUF_LN11 0

#define HECMW_MAX_TSUF_LN12 0

#define HECMW_MAX_TSUF_LN13 0

#define HECMW_MAX_TSUF_LN14 0

#define HECMW_MAX_TSUF_LN15 0

#define HECMW_MAX_TSUF_LN16 0

#define HECMW_MAX_TSUF_LN21 0

#define HECMW_MAX_TSUF_LN22 0

#define HECMW_MAX_TSUF_LN23 0

#define HECMW_MAX_TSUF_LN24 0

#define HECMW_MAX_TSUF_LN25 0

#define HECMW_MAX_TSUF_LN26 0

#define HECMW_MAX_TSUF_LN31 0

#define HECMW_MAX_TSUF_LN32 0

#define HECMW_MAX_TSUF_LN33 0

#define HECMW_MAX_TSUF_LN34 0

#define HECMW_MAX_TSUF_LN35 0

#define HECMW_MAX_TSUF_LN36 0

#define HECMW_MAX_TSUF_LN41 0

#define HECMW_MAX_TSUF_LN42 0

#define HECMW_MAX_TSUF_LN43 0

#define HECMW_MAX_TSUF_LN44 0

#define HECMW_MAX_TSUF_LN45 0

#define HECMW_MAX_TSUF_LN46 0

#define HECMW_MAX_TSUF_LN51 0

#define HECMW_MAX_TSUF_LN52 0

#define HECMW_MAX_TSUF_LN53 0

#define HECMW_MAX_TSUF_LN54 0

#define HECMW_MAX_TSUF_LN55 0

#define HECMW_MAX_TSUF_LN56 0

#define HECMW_MAX_TSUF_LN61 0

#define HECMW_MAX_TSUF_LN62 0

#define HECMW_MAX_TSUF_LN63 0

#define HECMW_MAX_TSUF_LN64 0

#define HECMW_MAX_TSUF_LN65 0

#define HECMW_MAX_TSUF_LN66 0

#define HECMW_MAX_QSUF_MAX 6

#define HECMW_MAX_QSUF_PNT 0

#define HECMW_MAX_QSUF_ROD1 0

#define HECMW_MAX_QSUF_ROD2 0

#define HECMW_MAX_QSUF_TRI1 0

#define HECMW_MAX_QSUF_TRI2 0

#define HECMW_MAX_QSUF_QUA1 0

#define HECMW_MAX_QSUF_QUA2 0

#define HECMW_MAX_QSUF_TET1 0

#define HECMW_MAX_QSUF_TET2 0

#define HECMW_MAX_QSUF_PRI1 3

#define HECMW_MAX_QSUF_PRI2 3

#define HECMW_MAX_QSUF_HEX1 6

#define HECMW_MAX_QSUF_HEX2 6

#define HECMW_MAX_QSUF_PYR1 1

#define HECMW_MAX_QSUF_PYR2 1

#define HECMW_MAX_QSUF_MST1 0

#define HECMW_MAX_QSUF_MST2 0

#define HECMW_MAX_QSUF_MSQ1 1

#define HECMW_MAX_QSUF_MSQ2 1

#define HECMW_MAX_QSUF_JTT1 0

#define HECMW_MAX_QSUF_JTT2 0

#define HECMW_MAX_QSUF_JTQ1 2

#define HECMW_MAX_QSUF_JTQ2 2

#define HECMW_MAX_QSUF_BEM1 0

#define HECMW_MAX_QSUF_BEM2 0

#define HECMW_MAX_QSUF_SHT1 0

#define HECMW_MAX_QSUF_SHT2 0

#define HECMW_MAX_QSUF_SHQ1 2

#define HECMW_MAX_QSUF_SHQ2 2

#define HECMW_MAX_QSUF_LN11 0

#define HECMW_MAX_QSUF_LN12 0

#define HECMW_MAX_QSUF_LN13 0

#define HECMW_MAX_QSUF_LN14 0

#define HECMW_MAX_QSUF_LN15 0

#define HECMW_MAX_QSUF_LN16 0

#define HECMW_MAX_QSUF_LN21 0

#define HECMW_MAX_QSUF_LN22 0

#define HECMW_MAX_QSUF_LN23 0

#define HECMW_MAX_QSUF_LN24 0

#define HECMW_MAX_QSUF_LN25 0

#define HECMW_MAX_QSUF_LN26 0

#define HECMW_MAX_QSUF_LN31 0

#define HECMW_MAX_QSUF_LN32 0

#define HECMW_MAX_QSUF_LN33 0

#define HECMW_MAX_QSUF_LN34 0

#define HECMW_MAX_QSUF_LN35 0

#define HECMW_MAX_QSUF_LN36 0

#define HECMW_MAX_QSUF_LN41 0

#define HECMW_MAX_QSUF_LN42 0

#define HECMW_MAX_QSUF_LN43 0

#define HECMW_MAX_QSUF_LN44 0

#define HECMW_MAX_QSUF_LN45 0

#define HECMW_MAX_QSUF_LN46 0

#define HECMW_MAX_QSUF_LN51 0

#define HECMW_MAX_QSUF_LN52 0

#define HECMW_MAX_QSUF_LN53 0

#define HECMW_MAX_QSUF_LN54 0

#define HECMW_MAX_QSUF_LN55 0

#define HECMW_MAX_QSUF_LN56 0

#define HECMW_MAX_QSUF_LN61 0

#define HECMW_MAX_QSUF_LN62 0

#define HECMW_MAX_QSUF_LN63 0

#define HECMW_MAX_QSUF_LN64 0

#define HECMW_MAX_QSUF_LN65 0

#define HECMW_MAX_QSUF_LN66 0

#define HECMW_MESH_NDOFGRP_MAX 3

#define HECMW_MESH_DOF_MAX 6

#define HECMW_MESH_DOF_TOT 3

#define HECMW_MESH_DOF_TWO 2


```
#define HECMW_MESH_DOF_THREE 3
```

```
#define HECMW_MESH_DOF_SIX 6
```

hecmw_config.h

HEC-MWで使用する定数等の定義.

```
#include "mpi.h"
```

マクロ定義

- `#define HECMW_INT ((HECMW_Datatype)10001)`
*MPI_INT*をラップする.
- `#define HECMW_DOUBLE ((HECMW_Datatype)10002)`
*MPI_DOUBLE*をラップする.
- `#define HECMW_CHAR ((HECMW_Datatype)10003)`
*MPI_CHAR*をラップする.
- `#define HECMW_MIN ((HECMW_Op)20001)`
*MPI_MIN*をラップする.
- `#define HECMW_MAX ((HECMW_Op)20002)`
*MPI_MAX*をラップする.
- `#define HECMW_SUM ((HECMW_Op)20003)`
*MPI_SUM*をラップする.
- `#define HECMW_EXIT_SUCCESS 0`
正常終了時の終了コード
- `#define HECMW_EXIT_ERROR 1`
異常終了時の終了コード

- `#define HECMW_HEADER_LEN 127`
ヘッダの最大長('¥0'は含まない)
- `#define HECMW_NAME_LEN 63`
名前の最大長('¥0'は含まない)
- `#define HECMW_FILENAME_LEN 1023`
ファイル名(パス)の最大長('¥0'は含まない)
- `#define HECMW_MSG_LEN 255`
メッセージの最大長('¥0'は含まない)

型定義

- `typedef MPI_Comm HECMW_Comm`
MPI コミュニケータをラップする.
- `typedef MPI_Group HECMW_Group`
MPI グループをラップする.
- `typedef MPI_Request HECMW_Request`
MPI リクエストハンドルをラップする.
- `typedef MPI_Status HECMW_Status`
MPI ステータスオブジェクトをラップする.
- `typedef MPI_Datatype HECMW_Datatype`
MPI データタイプをラップする.
- `typedef MPI_Op HECMW_Op`

MPI オペレーションをラップする.

- `typedef MPI_Fint HECMW_Fint`
MPI Fortran コミュニケータをラップする.

説明

HEC-MW で使用される定数等の定義.

日付:

2004年2月16日

作者:

坂根 一彰

マクロ定義

#define HECMW_INT ((HECMW_Datatype)10001)

MPI_INT をラップする.

#define HECMW_DOUBLE ((HECMW_Datatype)10002)

MPI_DOUBLE をラップする.

#define HECMW_CHAR ((HECMW_Datatype)10003)

MPI_CHAR をラップする.

#define HECMW_MIN ((HECMW_Op)20001)

MPI_MINをラップする.

#define HECMW_MAX ((HECMW_Op)20002)

MPI_MAXをラップする.

#define HECMW_SUM ((HECMW_Op)20003)

MPI_SUMをラップする.

#define HECMW_EXIT_SUCCESS 0

正常終了時の終了コード

#define HECMW_EXIT_ERROR 1

異常終了時の終了コード

#define HECMW_HEADER_LEN 127

ヘッダの最大長(¥0'は含まない)

#define HECMW_NAME_LEN 63

名前の最大長(¥0は含まない)

#define HECMW_FILENAME_LEN 1023

ファイル名(パス)の最大長(¥0は含まない)

#define HECMW_MSG_LEN 255

メッセージの最大長(¥0は含まない)

型定義

typedef MPI_Comm HECMW_Comm

MPIコミュニケーターをラップする.

typedef MPI_Group HECMW_Group

MPIグループをラップする.

typedef MPI_Request HECMW_Request

MPIリクエストハンドルをラップする.

typedef MPI_Status HECMW_Status

MPIステータスオブジェクトをラップする.

typedef MPI_Datatype HECMW_Datatype

MPIデータタイプをラップする.

typedef MPI_Op HECMW_Op

MPIオペレーションをラップする.

typedef MPI_Fint HECMW_Fint

MPI Fortran コミュニケータをラップする.

hecmw_control.h

全体制御ファイルの読み込みと，その情報取得を行う

データ構造

- struct **hecmw_ctrl_meshfile**
全体制御ファイルから取得したメッシュファイルの情報を格納する
- struct **hecmw_ctrl_meshfiles**
メッシュ情報構造体

マクロ定義

- #define **HECMW_CTRL_FILE** "hecmw_ctrl.dat"
デフォルトの全体制御ファイル名
- #define **HECMW_CTRL_FTYPE_HECMW_DIST** 1
TYPE=HECMW-DIST.
- #define **HECMW_CTRL_FTYPE_HECMW_ENTIRE** 2
TYPE=HECMW-ENTIRE.
- #define **HECMW_CTRL_FTYPE_GEOFEM** 3
TYPE=GeoFE<.
- #define **HECMW_CTRL_FTYPE_ABAQUS** 4
TYPE=ABAQUS.

- `#define HECMW_CTRL_FTYPE_NASTRAN 5`
`TYPE=NASTRAN.`
- `#define HECMW_CTRL_FTYPE_FEMAP 6`
`TYPE=FEMAP.`
- `#define HECMW_CTRL_FILE_IO_IN 1`
`IO=IN.`
- `#define HECMW_CTRL_FILE_IO_OUT 2`
`IO=OUT.`
- `#define HECMW_CTRL_FILE_IO_INOUT 4`
`IO=INOUT.`

関数

- `int HECMW_ctrl_init (void)`
全体制御ファイルの読み込みを行う
- `int HECMW_ctrl_finalize (void)`
全体制御データの終了処理を行う
- `hecmw_ctrl_meshfiles * HECMW_ctrl_get_meshfiles (char *name_ID)`
メッシュ情報を返す
- `hecmw_ctrl_meshfiles * HECMW_ctrl_get_meshfiles_header (char *name_ID)`
メッシュ情報を返す
- `void HECMW_ctrl_free_meshfiles (struct hecmw_ctrl_meshfiles *meshfiles)`
取得したメッシュ情報を解放する

- `char * HECMW_ctrl_get_restart_file (char *name_ID, char *buf, int bufsize)`
リスタートファイル名を返す
- `char * HECMW_ctrl_get_restart_file_by_io (int io, char *buf, int bufsize)`
リスタートファイル名を返す
- `char * HECMW_ctrl_get_result_file (char *name_ID, char *buf, int bufsize)`
結果ファイル名を返す
- `char * HECMW_ctrl_get_result_fileheader (char *name_ID, char *buf, int bufsize)`
結果ファイル名を返す
- `char * HECMW_ctrl_get_result_file_by_io (int io, char *buf, int bufsize)`
結果ファイル名を返す
- `char * HECMW_ctrl_get_result_fileheader_by_io (int io, char *buf, int bufsize)`
結果ファイル名を返す
- `char * HECMW_ctrl_get_control_file (char *name_ID, char *buf, int bufsize)`
制御ファイル名を返す

説明

全体制御ファイルの読み込みと、その情報取得を行う

全体制御ファイルはパラメータを外部から容易に与えることを可能とする。これにより、実行時に必要な機能の制御を簡単に行うことができる。

全体制御ファイルはプログラムの実行カレントディレクトリに置く必要があり、そのファイル名は'`hecmw_ctrl.dat`'である。また、この読み込みは全体初期化処理によって自動的に行われる。

参照:

全体制御ファイル

日付:

2004年2月16日

作者:

坂根 一彰

マクロ定義

#define HECMW_CTRL_FILE "hecmw_ctrl.dat"

デフォルトの全体制御ファイル名

#define HECMW_CTRL_FTYPE_HECMW_DIST 1

TYPE=HECMW-DIST.

#define HECMW_CTRL_FTYPE_HECMW_ENTIRE 2

TYPE=HECMW-ENTIRE.

#define HECMW_CTRL_FTYPE_GEOFEM 3

TYPE=GeoFE<.

#define HECMW_CTRL_FTYPE_ABAQUS 4

TYPE=ABAQUS.

#define HECMW_CTRL_FTYPE_NASTRAN 5

TYPE=NASTRAN.

#define HECMW_CTRL_FTYPE_FEMAP 6

TYPE=FEMAP.

#define HECMW_CTRL_FILE_IO_IN 1

IO=IN.

#define HECMW_CTRL_FILE_IO_OUT 2

IO=OUT.

#define HECMW_CTRL_FILE_IO_INOUT 4

IO=INOUT.

関数

int HECMW_ctrl_init (void)

全体制御ファイルの読み込みを行う

全体制御ファイルから全体制御データが読み込まれ、メモリ上に保持される。読み込まれる全体制御ファイルは、**HECMW_CTRL_FILE**で定義されるファイルである。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_ctrl_finalize (void)

全体制御データの終了処理を行う

メモリ上に保持されている全体制御データを解放する

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmw_ctrl_meshfiles* HECMW_ctrl_get_meshfiles (char * *name_ID*)

メッシュ情報を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!MESHまたは!MESH GROUPの情報を取得して返す。メッシュファイル名には、必要に応じてランク番号(.rank)が自動的に付加される。ここで取得した情報は、必要でなくなった時点でHECMW_free_meshfiles()によって解放することができる。

引数:

name_ID 取得したい!MESHまたは!MESH GROUPのNAMEパラメータ

戻り値:

成功すれば取得できたメッシュ情報を返す。失敗すればNULLを返し、
HECMW_set_error()によってエラー情報がセットされる。

参照:

HECMW_free_meshfiles()

**struct hecmw_ctrl_meshfiles* HECMW_ctrl_get_meshfiles_header (char *
name_ID)**

メッシュ情報を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!MESHまたは!MESH GROUP
の情報を取得して返す。メッシュファイル名にランク番号は付加されない。ここで取得し
た情報は、必要でなくなった時点でHECMW_free_meshfiles()によって解放することができる。

引数:

name_ID 取得したい!MESHまたは!MESH GROUPのNAMEパラメータ

戻り値:

成功すれば取得できたメッシュ情報を返す。失敗すればNULLを返し、
HECMW_set_error()によってエラー情報がセットされる。

参照:

HECMW_free_meshfiles()

void HECMW_ctrl_free_meshfiles (struct hecmw_ctrl_meshfiles * *meshfiles*)

取得したメッシュ情報を解放する

引数:

meshfiles 解放したいメッシュ情報へのポインタ

char* HECMW_ctrl_get_restart_file (char * *name_ID*, char * *buf*, int *bufsize*)

リスタートファイル名を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!RESTARTのファイル名を取得して返す。ファイル名には、ランク番号(.rank)が自動的に付加される。

引数:

name_ID 取得したい!RESTARTのNAMEパラメータ

buf 取得したファイル名を格納するバッファ領域。最大で

HECMW_FILENAME_LEN+1の大きさが必要である。 NULLを指定すると動的にメモリを確保しそれを格納先に使用する。

bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できたリスタートファイル名へのポインタを返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

char* HECMW_ctrl_get_restart_file_by_io (int *io*, char * *buf*, int *bufsize*)

リスタートファイル名を返す

全体制御データから、引数で示されるIOパラメータを持つ!RESTARTのうち、最初に見つかったもののファイル名を返す。ファイル名には、ランク番号(.rank)が自動的に付加される。

引数:

io 取得したい!RESTARTのIOパラメータ。対象とするIOパラメータが複数ある場合は、その論理和を指定すればよい。

buf 取得したファイル名を格納するバッファ領域。最大で

HECMW_FILENAME_LEN+1の大きさが必要である。 NULLを指定すると動的にメモリを確保しそれを格納先に使用する。

bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できたリスタートファイル名へのポインタを返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

char* HECMW_ctrl_get_result_file (char * *name_ID*, char * *buf*, int *bufsize*)

結果ファイル名を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!RESULTのファイル名を取得して返す。ファイル名には、ランク番号(.rank)が自動的に付加される。

引数:

name_ID 取得したい!RESULTのNAMEパラメータ

buf 取得したファイル名を格納するバッファ領域。最大で

HECMW_FILENAME_LEN+1の大きさが必要である。NULLを指定すると動的にメモリを確保しそれを格納先に使用する。

bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できた結果ファイル名へのポインタを返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

char* HECMW_ctrl_get_result_fileheader (char * *name_ID*, char * *buf*, int *bufsize*)

結果ファイル名を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!RESULTの情報を取得して返す。ファイル名にランク番号は付加されない。

引数:

name_ID 取得したい!RESULTのNAMEパラメータ

buf 取得したファイル名を格納するバッファ領域。 最大で **HECMW_FILENAME_LEN+1**の大きさが必要である。 NULLを指定すると動的にメモリを確保しそれを格納先に使用する。
bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できた結果ファイル名へのポインタを返す。 失敗すればNULLを返し、**HECMW_set_error()**によってエラー情報がセットされる。

char* HECMW_ctrl_get_result_file_by_io (int *io*, char * *buf*, int *bufsize*)

結果ファイル名を返す

全体制御データから、引数で示されるIOパラメータを持つ!RESULTのうち、 最初に見つかったもののファイル名を返す。 ファイル名には、ランク番号(.rank)が自動的に付加される。

引数:

io 取得したい!RESULTのIOパラメータ。 対象とするIOパラメータが複数ある場合は、その論理和を指定すればよい。

buf 取得したファイル名を格納するバッファ領域。 最大で **HECMW_FILENAME_LEN+1**の大きさが必要である。 NULLを指定すると動的にメモリを確保しそれを格納先に使用する。
bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できた結果ファイル名へのポインタを返す。 失敗すればNULLを返し、**HECMW_set_error()**によってエラー情報がセットされる。

char* HECMW_ctrl_get_result_fileheader_by_io (int *io*, char * *buf*, int *bufsize*)

結果ファイル名を返す

全体制御データから、引数で示されるIOパラメータを持つ!RESULTのうち、最初に見つかったもののファイル名を返す。ファイル名にランク番号は付加されない。

引数:

io 取得したい!RESULTのIOパラメータ。対象とするIOパラメータが複数ある場合は、その論理和を指定すればよい。

buf 取得したファイル名を格納するバッファ領域。最大で

HECMW_FILENAME_LEN+1の大きさが必要である。NULLを指定すると動的にメモリを確保しそれを格納先に使用する。

bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できた結果ファイル名へのポインタを返す。失敗すればNULLを返し、**HECMW_set_error()**によってエラー情報がセットされる。

char* HECMW_ctrl_get_control_file (char * *name_ID*, char * *buf*, int *bufsize*)

制御ファイル名を返す

全体制御データから、引数で示されるNAMEパラメータを持つ!CONTROLのファイル名を取得して返す。ファイル名には、ランク番号(.rank)が自動的に付加される。

引数:

name_ID 取得したい!CONTROLのNAMEパラメータ

buf 取得したファイル名を格納するバッファ領域。最大で

HECMW_FILENAME_LEN+1の大きさが必要である。NULLを指定すると動的にメモリを確保しそれを格納先に使用する。

bufsize *buf*のサイズ。 *buf*がNULLの場合は何でもよい。

戻り値:

成功すれば取得できた制御ファイル名へのポインタを返す。失敗すればNULLを返し、**HECMW_set_error()**によってエラー情報がセットされる。

hecmw_ctrllex.h

全体制御データの字句解析ルーチン

```
#include <stdio.h>
```

列挙型

- enum { **HECMW_CTRLLEX_NL** = 1000, **HECMW_CTRLLEX_INT**,
HECMW_CTRLLEX_DOUBLE, **HECMW_CTRLLEX_NAME**,
HECMW_CTRLLEX_FILENAME, **HECMW_CTRLLEX_H_CONTROL** = 2000,
HECMW_CTRLLEX_H_MESH, **HECMW_CTRLLEX_H_MESH_GROUP**,
HECMW_CTRLLEX_H_RESULT, **HECMW_CTRLLEX_H_RESTART**,
HECMW_CTRLLEX_H_COUPLEMESH, **HECMW_CTRLLEX_H_COUPLE**,
HECMW_CTRLLEX_H_PROG1, **HECMW_CTRLLEX_H_PROG2**,
HECMW_CTRLLEX_K_ABAQUS = 3000, **HECMW_CTRLLEX_K_DIR**,
HECMW_CTRLLEX_K_FEMAP, **HECMW_CTRLLEX_K_GEOFEM**,
HECMW_CTRLLEX_K_HECMW_DIST, **HECMW_CTRLLEX_K_HECMW_ENTIRE**,
HECMW_CTRLLEX_K_IN, **HECMW_CTRLLEX_K_INOUT**, **HECMW_CTRLLEX_K_IO**,
HECMW_CTRLLEX_K_NAME, **HECMW_CTRLLEX_K_NASTRAN**,
HECMW_CTRLLEX_K_OUT, **HECMW_CTRLLEX_K_TYPE**,
HECMW_CTRLLEX_K_NPROC, **HECMW_CTRLLEX_K_MXN**,
HECMW_CTRLLEX_K_MAXMN }

トークン番号定義

関数

- double **HECMW_ctrllex_get_number** (void)
直前に読み込んだトークンが数値であった場合、その値を返す
- char * **HECMW_ctrllex_get_text** (void)
直前に読み込んだトークンの文字列表記を返す

- **int HECMW_ctrllex_get_lineno** (void)
読み込んでいるファイルの現在の行番号を返す
 - **int HECMW_ctrllex_next_token** (void)
次のトークンを返す
 - **int HECMW_ctrllex_next_token_skip** (int skip_token)
次のトークンを返す
 - **int HECMW_ctrllex_set_input** (FILE *fp)
字句解析ルーチンで読み込むファイルのファイルポインタをセットする
 - **int HECMW_ctrllex_skip_line** (void)
現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す
 - **int HECMW_ctrllex_unput_token** (void)
直前に読み込んだトークンを押し戻す
-

説明

全体制御データの字句解析ルーチン

日付:

2004年2月16日

作者:

坂根 一彰

列挙型

anonymous enum

トークン番号定義

列挙型の値:

HECMW_CTRLLEX_NL 改行文字
HECMW_CTRLLEX_INT 整数
HECMW_CTRLLEX_DOUBLE 浮動小数点
HECMW_CTRLLEX_NAME 名前
HECMW_CTRLLEX_FILENAME ファイル名
HECMW_CTRLLEX_H_CONTROL !CONTROL
HECMW_CTRLLEX_H_MESH !MESH
HECMW_CTRLLEX_H_MESH_GROUP !MESH GROUP
HECMW_CTRLLEX_H_RESULT !RESULT
HECMW_CTRLLEX_H_RESTART !RESTART
HECMW_CTRLLEX_H_COUPLEMESH !COUPLEMESH
HECMW_CTRLLEX_H_COUPLE !COUPLE
HECMW_CTRLLEX_H_PROG1 !PROG1
HECMW_CTRLLEX_H_PROG2 !PROG2
HECMW_CTRLLEX_K_ABAQUS ABAQUS.
HECMW_CTRLLEX_K_DIR DIR.
HECMW_CTRLLEX_K_FEMAP FEMAP.
HECMW_CTRLLEX_K_GEOFEM GeoFEM.
HECMW_CTRLLEX_K_HECMW_DIST HECMW-DIST.
HECMW_CTRLLEX_K_HECMW_ENTIRE HECMW-ENTIRE.
HECMW_CTRLLEX_K_IN IN.
HECMW_CTRLLEX_K_INOUT INOUT.
HECMW_CTRLLEX_K_IO IO.
HECMW_CTRLLEX_K_NAME NAME.
HECMW_CTRLLEX_K_NASTRAN NASTRAN.
HECMW_CTRLLEX_K_OUT OUT.
HECMW_CTRLLEX_K_TYPE TYPE.
HECMW_CTRLLEX_K_NPROC NPROC.
HECMW_CTRLLEX_K_MXN MXN.

HECMW_CTRLLEX_K_MAXMN MAXMN.

関数

double HECMW_ctrllex_get_number (void)

直前に読み込んだトークンが数値であった場合、その値を返す

戻り値:

直前に読み込まれた数値

char* HECMW_ctrllex_get_text (void)

直前に読み込んだトークンの文字列表記を返す

戻り値:

直前に読み込まれたトークンの文字列表記

int HECMW_ctrllex_get_lineno (void)

読み込んでいるファイルの現在の行番号を返す

戻り値:

行番号

int HECMW_ctrllex_next_token (void)

次のトークンを返す

EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_ctrllex_next_token_skip (int *skip_token*)

次のトークンを返す

*skip_token*で指定されたトークンの連続は読み飛ばされる。それ以外のトークンが現れたらそれを返す。 EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_ctrllex_set_input (FILE * *fp*)

字句解析ルーチンで読み込むファイルのファイルポインタをセットする

引数:

fp 字句解析対象のファイルへのファイルポインタ

戻り値:

成功すれば0を返す。 引数が無効なら-1を返す。

int HECMW_ctrllex_skip_line (void)

現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す

戻り値:

現在の行の最後のトークン番号. EOFに達した場合は0を返す.

int HECMW_ctrllex_unput_token (void)

直前に読み込んだトークンを押し戻す

次回取得したトークンは, 押し戻したトークンとなる.

戻り値:

常に0を返す.

hecmw_debug_write_dist.h

分散メッシュデータのDEBUG出力

```
#include "hecmw_struct.h"
```

マクロ定義

- #define **HECMW_dbg_n_node**(mesh)
- #define **HECMW_dbg_nn_internal**(mesh)
- #define **HECMW_dbg_n_dof**(mesh)
- #define **HECMW_dbg_n_dof_grp**(mesh)
- #define **HECMW_dbg_node**(mesh)
- #define **HECMW_dbg_node_id**(mesh)
- #define **HECMW_dbg_node_id_lid**(mesh)
- #define **HECMW_dbg_node_id_domain**(mesh)
- #define **HECMW_dbg_global_node_id**(mesh)
- #define **HECMW_dbg_node_dof_item**(mesh)
- #define **HECMW_dbg_node_dof_index**(mesh)
- #define **HECMW_dbg_node_init_val_item**(mesh)
- #define **HECMW_dbg_node_init_val_index**(mesh)
- #define **HECMW_dbg_n_elem**(mesh)
- #define **HECMW_dbg_ne_internal**(mesh)
- #define **HECMW_dbg_n_elem_type**(mesh)
- #define **HECMW_dbg_elem_type_item**(mesh)
- #define **HECMW_dbg_elem_type_index**(mesh)
- #define **HECMW_dbg_elem_type**(mesh)
- #define **HECMW_dbg_elem_node_item**(mesh)
- #define **HECMW_dbg_elem_node_index**(mesh)
- #define **HECMW_dbg_elem_id**(mesh)
- #define **HECMW_dbg_elem_id_lid**(mesh)
- #define **HECMW_dbg_elem_id_domain**(mesh)
- #define **HECMW_dbg_global_elem_id**(mesh)
- #define **HECMW_dbg_section_id**(mesh)

- #define **HECMW_dbg_n_elem_mat_id**(mesh)
- #define **HECMW_dbg_elem_mat_id_item**(mesh)
- #define **HECMW_dbg_elem_mat_id_index**(mesh)
- #define **HECMW_dbg_elem_internal_list**(mesh)
- #define **HECMW_dbg_n_neighbor_pe**(mesh)
- #define **HECMW_dbg_neighbor_pe**(mesh)
- #define **HECMW_dbg_import_item**(mesh)
- #define **HECMW_dbg_import_index**(mesh)
- #define **HECMW_dbg_export_item**(mesh)
- #define **HECMW_dbg_export_index**(mesh)
- #define **HECMW_dbg_shared_item**(mesh)
- #define **HECMW_dbg_shared_index**(mesh)
- #define **HECMW_dbg_n_sect**(mesh)
- #define **HECMW_dbg_section**(mesh)
- #define **HECMW_dbg_n_mat**(mesh)
- #define **HECMW_dbg_material**(mesh)
- #define **HECMW_dbg_n_mpc**(mesh)
- #define **HECMW_dbg_mpc**(mesh)
- #define **HECMW_dbg_n_amp**(mesh)
- #define **HECMW_dbg_amp**(mesh)
- #define **HECMW_dbg_n_node_group**(mesh)
- #define **HECMW_dbg_node_group**(mesh)
- #define **HECMW_dbg_n_elem_group**(mesh)
- #define **HECMW_dbg_elem_group**(mesh)
- #define **HECMW_dbg_n_surf_group**(mesh)
- #define **HECMW_dbg_surf_group**(mesh)

関数

- void **HECMW_dbg_n_node_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_nn_internal_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_dof_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_dof_grp_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_id_lid_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)

- void **HECMW_dbg_node_id_domain_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_global_node_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_dof_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_dof_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_init_val_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_init_val_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_elem_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_ne_internal_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_elem_type_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_type_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_type_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_type_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_node_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_node_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_id_lid_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_id_domain_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_global_elem_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_internal_list_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_section_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_elem_mat_id_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)

- void **HECMW_dbg_elem_mat_id_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_mat_id_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_neighbor_pe_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_neighbor_pe_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_import_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_import_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_export_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_export_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_shared_item_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_shared_index_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_sect_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_section_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_mat_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_material_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_mpc_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_mpc_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_amp_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_amp_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_node_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_node_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_elem_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_elem_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_n_surf_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)
- void **HECMW_dbg_surf_group_** (struct **hecmwST_local_mesh** *local_mesh, char *file, int line)

説明

分散メッシュデータの DEBUG 出力

マクロ定義

```
#define HECMW_dbg_n_node(mesh)
```

```
#define HECMW_dbg_nn_internal(mesh)
```

```
#define HECMW_dbg_n_dof(mesh)
```

```
#define HECMW_dbg_n_dof_grp(mesh)
```

```
#define HECMW_dbg_node(mesh)
```

```
#define HECMW_dbg_node_id(mesh)
```

```
#define HECMW_dbg_node_id_lid(mesh)
```

```
#define HECMW_dbg_node_id_domain(mesh)
```

```
#define HECMW_dbg_global_node_id(mesh)
```

```
#define HECMW_dbg_node_dof_item(mesh)
```

#define HECMW_dbg_node_dof_index(mesh)

#define HECMW_dbg_node_init_val_item(mesh)

#define HECMW_dbg_node_init_val_index(mesh)

#define HECMW_dbg_n_elem(mesh)

#define HECMW_dbg_ne_internal(mesh)

#define HECMW_dbg_n_elem_type(mesh)

#define HECMW_dbg_elem_type_item(mesh)

#define HECMW_dbg_elem_type_index(mesh)

#define HECMW_dbg_elem_type(mesh)

#define HECMW_dbg_elem_node_item(mesh)

#define HECMW_dbg_elem_node_index(mesh)

#define HECMW_dbg_elem_id(mesh)

#define HECMW_dbg_elem_id_lid(mesh)

#define HECMW_dbg_elem_id_domain(mesh)

#define HECMW_dbg_global_elem_id(mesh)

#define HECMW_dbg_section_id(mesh)

#define HECMW_dbg_n_elem_mat_id(mesh)

#define HECMW_dbg_elem_mat_id_item(mesh)

#define HECMW_dbg_elem_mat_id_index(mesh)

#define HECMW_dbg_elem_internal_list(mesh)

#define HECMW_dbg_n_neighbor_pe(mesh)

#define HECMW_dbg_neighbor_pe(mesh)

#define HECMW_dbg_import_item(mesh)

#define HECMW_dbg_import_index(mesh)

#define HECMW_dbg_export_item(mesh)

#define HECMW_dbg_export_index(mesh)

#define HECMW_dbg_shared_item(mesh)

#define HECMW_dbg_shared_index(mesh)

#define HECMW_dbg_n_sect(mesh)

#define HECMW_dbg_section(mesh)

#define HECMW_dbg_n_mat(mesh)

#define HECMW_dbg_material(mesh)

#define HECMW_dbg_n_mpc(mesh)

#define HECMW_dbg_mpc(mesh)

#define HECMW_dbg_n_amp(mesh)

```
#define HECMW_dbg_amp(mesh)
```

```
#define HECMW_dbg_n_node_group(mesh)
```

```
#define HECMW_dbg_node_group(mesh)
```

```
#define HECMW_dbg_n_elem_group(mesh)
```

```
#define HECMW_dbg_elem_group(mesh)
```

```
#define HECMW_dbg_n_surf_group(mesh)
```

```
#define HECMW_dbg_surf_group(mesh)
```

関数

```
void HECMW_dbg_n_node_ (struct hecmwST_local_mesh * local_mesh, char *  
file, int line)
```

```
void HECMW_dbg_nn_internal_ (struct hecmwST_local_mesh * local_mesh, char  
* file, int line)
```

**void HECMW_dbg_n_dof_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_n_dof_grp_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_node_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_node_id_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_node_id_lid_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

**void HECMW_dbg_node_id_domain_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_global_node_id_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_node_dof_item_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_node_dof_index_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_node_init_val_item_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_node_init_val_index_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_n_elem_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_ne_internal_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

**void HECMW_dbg_n_elem_type_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_type_item_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_type_index_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_type_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_elem_node_item_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_node_index_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_elem_id_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_elem_id_lid_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

**void HECMW_dbg_elem_id_domain_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_global_elem_id_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_internal_list_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_section_id_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_n_elem_mat_id_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_mat_id_item_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_elem_mat_id_index_ (struct hecmwST_local_mesh *
local_mesh, char * *file*, int *line*)**

**void HECMW_dbg_n_neighbor_pe_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_neighbor_pe_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_import_item_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_import_index_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_export_item_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

**void HECMW_dbg_export_index_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_shared_item_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_shared_index_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_n_sect_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_section_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_n_mat_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_material_ (struct hecmwST_local_mesh * *local_mesh*, char *
file, int *line*)**

**void HECMW_dbg_n_mpc_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_mpc_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_n_amp_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_amp_ (struct hecmwST_local_mesh * *local_mesh*, char * *file*,
int *line*)**

**void HECMW_dbg_n_node_group_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_node_group_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_n_elem_group_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_elem_group_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

**void HECMW_dbg_n_surf_group_ (struct hecmwST_local_mesh * *local_mesh*,
char * *file*, int *line*)**

**void HECMW_dbg_surf_group_ (struct hecmwST_local_mesh * *local_mesh*, char
* *file*, int *line*)**

hecmw_dist.h

メッシュデータ構造体に関する処理

```
#include "hecmw_struct.h"
```

関数

- `int HECMW_dist_get_mat_id (const struct hecmwST_material *mat, const char *name)`
材料情報構造体に含まれる材料のうち、材料名が`name`である材料のIDを返す.
 - `int HECMW_dist_get_egrp_id (const struct hecmwST_elem_grp *egrp, const char *name)`
要素グループ情報構造体に含まれる材料のうち、要素グループ名が`name`である要素グループのIDを返す.
 - `int HECMW_dist_gid2lid_node (const struct hecmwST_local_mesh *mesh, int gid)`
グローバル節点番号をローカル節点番号に変換する
 - `int HECMW_dist_gid2lid_elem (const struct hecmwST_local_mesh *mesh, int gid)`
グローバル要素番号をローカル要素番号に変換する
-

説明

メッシュデータ構造体に関する処理

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_dist_get_mat_id (const struct hecmwST_material * *mat*, const char * *name*)

材料情報構造体に含まれる材料のうち、材料名が*name*である材料のIDを返す。

材料IDは、1から始まる通し番号とする。

引数:

mat 材料情報構造体

name 材料名

戻り値:

成功すれば、材料IDを返す。 引数が無効、あるいは適切な材料が見付からなければ-1を返す。

int HECMW_dist_get_egrp_id (const struct hecmwST_elem_grp * *egrp*, const char * *name*)

要素グループ情報構造体に含まれる材料のうち、要素グループ名が*name*である要素グループのIDを返す。

要素グループIDは、1から始まる通し番号とする。

引数:

egrp 要素グループ情報構造体

name 要素グループ名

戻り値:

成功すれば、要素グループIDを返す。 引数が無効、あるいは適切な要素グループが見付からなければ-1を返す。

int HECMW_dist_gid2lid_node (const struct hecmwST_local_mesh * *mesh*, int *gid*)

グローバル節点番号をローカル節点番号に変換する

引数:

mesh メッシュデータ構造体

gid グローバル節点番号

戻り値:

成功すれば、ローカル節点番号を返す。 引数が無効、あるいは適切な節点が見付からなければ-1を返す。

int HECMW_dist_gid2lid_elem (const struct hecmwST_local_mesh * *mesh*, int *gid*)

グローバル要素番号をローカル要素番号に変換する

引数:

mesh メッシュデータ構造体

gid グローバル要素番号

戻り値:

成功すれば、ローカル要素番号を返す。 引数が無効、あるいは適切な要素が見付からなければ-1を返す。

hecmw_dist_copy_f2c.h

FortranからCへメッシュデータをコピーする.

関数

- `int HECMW_dist_copy_f2c_init (struct hecmwST_local_mesh *local_mesh)`
コピールーチンの初期化を行う
- `int HECMW_dist_copy_f2c_finalize (void)`
コピールーチンの終了処理を行う

説明

Fortran から C へメッシュデータをコピーする.

日付:

2004年2月16日

作者:

坂根 一彰

関数

`int HECMW_dist_copy_f2c_init (struct hecmwST_local_mesh * local_mesh)`

コピールーチンの初期化を行う

引数:

local_mesh コピー先メッシュデータ構造体. 構造体メンバは全て領域が確保されている必要がある. その他の構造体メンバはコピー時に確保される.

戻り値:

常に0を返す.

int HECMW_dist_copy_f2c_finalize (void)

コピールーチンの終了処理を行う

hecmw_dist_free.h

メッシュデータ構造体のメモリ領域を解放する.

```
#include "hecmw_struct.h"
```

関数

- void **HECMW_dist_free_section** (struct **hecmwST_section** *section)
セクション情報のメモリ領域を解放する
- void **HECMW_dist_free_material** (struct **hecmwST_material** *material)
材料情報のメモリ領域を解放する
- void **HECMW_dist_free_mpc** (struct **hecmwST_mpc** *mpc)
拘束情報のメモリ領域を解放する
- void **HECMW_dist_free_amplitude** (struct **hecmwST_amplitude** *amp)
AMPLITUDE情報のメモリ領域を解放する.
- void **HECMW_dist_free_ngrp** (struct **hecmwST_node_grp** *grp)
節点グループ情報のメモリ領域を解放する
- void **HECMW_dist_free_egrp** (struct **hecmwST_elem_grp** *grp)
要素グループ情報のメモリ領域を解放する
- void **HECMW_dist_free_sgrp** (struct **hecmwST_surf_grp** *grp)
面グループ情報のメモリ領域を解放する
- void **HECMW_dist_free** (struct **hecmwST_local_mesh** *mesh)
メッシュデータ構造体のメモリ領域を解放する

説明

メッシュデータ構造体のメモリ領域を解放する.

日付:

2004年2月16日

作者:

坂根 一彰

関数

void HECMW_dist_free_section (struct hecmwST_section * *section*)

セクション情報のメモリ領域を解放する

引数:

section セクション情報

void HECMW_dist_free_material (struct hecmwST_material * *material*)

材料情報のメモリ領域を解放する

引数:

material 材料情報

void HECMW_dist_free_mpc (struct hecmwST_mpc * *mpc*)

拘束情報のメモリ領域を解放する

引数:

mpc 拘束情報

void HECMW_dist_free_amplitude (struct hecmwST_amplitude * *amp*)

AMPLITUDE情報のメモリ領域を解放する.

引数:

amp AMPLITUDE情報

void HECMW_dist_free_ngrp (struct hecmwST_node_grp * *grp*)

節点グループ情報のメモリ領域を解放する

引数:

grp 節点グループ情報

void HECMW_dist_free_egrp (struct hecmwST_elem_grp * *grp*)

要素グループ情報のメモリ領域を解放する

引数:

grp 要素グループ情報

void HECMW_dist_free_sgrp (struct hecmwST_surf_grp * *grp*)

面グループ情報のメモリ領域を解放する

引数:

grp 面グループ情報

void HECMW_dist_free (struct hecmwST_local_mesh * *mesh*)

メッシュデータ構造体のメモリ領域を解放する

引数:

mesh メッシュデータ構造体

hecmw_dist_print.h

メッシュデータ構造体の内容出力する

```
#include <stdio.h>
#include "hecmw_struct.h"
```

関数

- void **HECMW_dist_print_flags** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
フラグを出力する
- void **HECMW_dist_print_header** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
headerを出力する
- void **HECMW_dist_print_gridfile** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
gridfileを出力する
- void **HECMW_dist_print_files** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
filesを出力する
- void **HECMW_dist_print_zero_temp** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
zero_tempを出力する
- void **HECMW_dist_print_node** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
節点情報を出力する
- void **HECMW_dist_print_elem** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
要素情報を出力する
- void **HECMW_dist_print_pe** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)

PE情報を出力する.

- void **HECMW_dist_print_adapt** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
階層情報を出力する
- void **HECMW_dist_print_section** (const struct **hecmwST_section** *sect, FILE *fp)
セクション情報を出力する
- void **HECMW_dist_print_material** (const struct **hecmwST_material** *material, FILE *fp)
材料情報を出力する
- void **HECMW_dist_print_mpc** (const struct **hecmwST_mpc** *mpc, FILE *fp)
拘束情報を出力する
- void **HECMW_dist_print_amp** (const struct **hecmwST_amplitude** *amp, FILE *fp)
AMPLITUDE情報を出力する.
- void **HECMW_dist_print_ngrp** (const struct **hecmwST_node_grp** *ngrp, FILE *fp)
節点グループ情報を出力する
- void **HECMW_dist_print_egrp** (const struct **hecmwST_elem_grp** *egrp, FILE *fp)
要素グループ情報を出力する
- void **HECMW_dist_print_sgrp** (const struct **hecmwST_surf_grp** *sgrp, FILE *fp)
面グループ情報を出力する
- void **HECMW_dist_print** (const struct **hecmwST_local_mesh** *mesh, FILE *fp)
メッシュデータの全ての情報を出力する

説明

メッシュデータ構造体の内容出力する

日付:

2004年2月16日

作者:

坂根 一彰

関数

void HECMW_dist_print_flags (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

フラグを出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_header (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

headerを出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_gridfile (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

gridfileを出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_files (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

filesを出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_zero_temp (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

zero_tempを出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_node (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

節点情報を出力する

引数:

mesh メッシュデータ構造体
fp 出力先ファイルポインタ

void HECMW_dist_print_elem (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

要素情報を出力する

引数:

mesh メッシュデータ構造体
fp 出力先ファイルポインタ

void HECMW_dist_print_pe (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

PE情報を出力する.

引数:

mesh メッシュデータ構造体
fp 出力先ファイルポインタ

void HECMW_dist_print_adapt (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

階層情報を出力する

引数:

mesh メッシュデータ構造体
fp 出力先ファイルポインタ

void HECMW_dist_print_section (const struct hecmwST_section * *sect*, FILE * *fp*)

セクション情報を出力する

引数:

sect セクション情報構造体

fp 出力先ファイルポインタ

**void HECMW_dist_print_material (const struct hecmwST_material * *material*,
FILE * *fp*)**

材料情報を出力する

引数:

material 材料情報構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_mpc (const struct hecmwST_mpc * *mpc*, FILE * *fp*)

拘束情報を出力する

引数:

mpc 拘束情報構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_amp (const struct hecmwST_amplitude * *amp*, FILE * *fp*)

AMPLITUDE情報を出力する.

引数:

amp AMPLITUDE情報構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_ngrp (const struct hecmwST_node_grp * *ngrp*, FILE * *fp*)

節点グループ情報を出力する

引数:

ngrp 節点グループ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_egrp (const struct hecmwST_elem_grp * *egrp*, FILE * *fp*)

要素グループ情報を出力する

引数:

egrp 要素グループ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print_sgrp (const struct hecmwST_surf_grp * *sgrp*, FILE * *fp*)

面グループ情報を出力する

引数:

sgrp 面グループ構造体

fp 出力先ファイルポインタ

void HECMW_dist_print (const struct hecmwST_local_mesh * *mesh*, FILE * *fp*)

メッシュデータの全ての情報を出力する

引数:

mesh メッシュデータ構造体

fp 出力先ファイルポインタ

hecmw_error.h

エラーが発生した場合、その情報を保存し、他の関数から取得可能とする。

```
#include <stdarg.h>
```

関数

- **int HECMW_set_verror** (int errorno, const char *fmt, va_list ap)
エラー情報をセットする。
- **int HECMW_set_error** (int errorno, const char *fmt,...)
エラー情報をセットする。
- **int HECMW_get_error** (char **errmsg)
エラー情報を取得する。
- **int HECMW_get_errno** (void)
エラー番号を取得する。
- **char * HECMW_get_errmsg** (void)
エラーメッセージを取得する。

説明

エラーが発生した場合、その情報を保存し、他の関数から取得可能とする。
保存される情報は以下のとおりである。

- エラー番号
- エラーメッセージ

これらの情報は、エラー情報をセットする関数によってセットされ、次の呼び出しまで保存される。

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_set_verror (int *errorno*, const char * *fmt*, va_list *ap*)

エラー情報をセットする。

ここでセットされた情報は次の呼び出しまで保存される。

引数:

errorno セットするエラー番号

fmt エラーメッセージを示す可変引数のフォーマット

ap エラーメッセージを示す可変引数リスト

戻り値:

エラー情報の設定に成功すれば0、失敗すれば0以外を返す。

int HECMW_set_error (int *errorno*, const char * *fmt*, ...)

エラー情報をセットする。

ここでセットされた情報は次の呼び出しまで保存される。

引数:

errorno セットするエラー番号

fmt エラーメッセージを示す可変引数のフォーマット
... エラーメッセージ

戻り値:

エラー情報の設定に成功すれば0, 失敗すれば-1を返す.

int HECMW_get_error (char ** *errmsg*)

エラー情報を取得する.

現在保存されているエラー番号と, それを示すエラーメッセージを取得する.

引数:

errmsg 現在保存されているエラーメッセージへのポインタへのポインタ

戻り値:

エラー番号

int HECMW_get_errno (void)

エラー番号を取得する.

現在保存されているエラー番号を取得する.

戻り値:

エラー番号

char* HECMW_get_errmsg (void)

エラーメッセージを取得する.

現在保存されているエラーメッセージを取得する.

戻り値:

エラーメッセージ

hecmw_etype.h

要素情報を提供する

```
#include "hecmw_util.h"
```

関数

- int **HECMW_get_etype_UTIL2HECMW** (int etype)
- int **HECMW_get_etype_HECMW2UTIL** (int etype)
- int **HECMW_get_etype_GeoFEM2HECMW** (int etype)
- int **HECMW_get_max_node** (int etype)
- int **HECMW_get_max_edge** (int etype)
- int **HECMW_get_max_surf** (int etype)
- int **HECMW_get_max_tsuf** (int etype)
- int **HECMW_get_max_qsuf** (int etype)
- char * **HECMW_get_ucd_label** (int etype)
- int **HECMW_is_etype_rod** (int etype)

etype がロッド要素かどうかを返す

- int **HECMW_is_etype_surface** (int etype)
- etype* がサーフェイス要素かどうかを返す

- int **HECMW_is_etype_solid** (int etype)
- etype* がソリッド要素かどうかを返す
- int **HECMW_is_etype_interface** (int etype)
- etype* がインタフェース要素かどうかを返す

- int **HECMW_is_etype_beam** (int etype)
- etype* がビーム要素かどうかを返す

- **int HECMW_is_etype_shell** (int *etype*)

etype がシェル要素かどうかを返す

- **int HECMW_is_etype_link** (int *etype*)

etype がリンク要素かどうかを返す

説明

要素情報を提供する

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_get_etype_UTIL2HECMW (int *etype*)

int HECMW_get_etype_HECMW2UTIL (int *etype*)

int HECMW_get_etype_GeoFEM2HECMW (int *etype*)

int HECMW_get_max_node (int *etype*)

int HECMW_get_max_edge (int *etype*)

int HECMW_get_max_surf (int *etype*)

int HECMW_get_max_tsuf (int *etype*)

int HECMW_get_max_qsuf (int *etype*)

char* HECMW_get_ucd_label (int *etype*)

int HECMW_is_etype_rod (int *etype*)

*etype*がロッド要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

ロッド要素なら1を返し，そうでなければ0を返す

int HECMW_is_etype_surface (int *etype*)

*etype*がサーフェイス要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

サーフェイス要素なら1を返し，そうでなければ0を返す

int HECMW_is_etype_solid (int etype)

etypeがソリッド要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

ソリッド要素なら1を返し、そうでなければ0を返す

int HECMW_is_etype_interface (int etype)

etypeがインタフェース要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

インタフェース要素なら1を返し、そうでなければ0を返す

int HECMW_is_etype_beam (int etype)

etypeがビーム要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

ビーム要素なら1を返し，そうでなければ0を返す

int HECMW_is_etype_shell (int etype)

etypeがシェル要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

シェル要素なら1を返し，そうでなければ0を返す

int HECMW_is_etype_link (int etype)

etypeがリンク要素かどうかを返す

引数:

etype HEC-MW要素番号

戻り値:

リンク要素なら1を返し，そうでなければ0を返す

hecmw_finalize.h

HEC-MWの全体終了処理を行う。

関数

- `int HECMW_finalize (void)`
HEC-MWの全体終了処理を行う。

説明

HEC-MW の全体終了処理を行う。
全体終了処理では、以下の処理が行われる。

- 全体制御ファイル終了処理
- MPI終了処理

HEC-MW 利用プログラムは、この全体終了処理を実行してプログラムを終了しなければならない。

日付:

2004年2月16日

作者:

坂根 一彰

関数

`int HECMW_finalize (void)`

HEC-MWの全体終了処理を行う.

戻り値:

成功すれば0を返す. 失敗すれば-1を返し, `HECMW_set_error()`によってエラー情報がセットされる.

hecmw_geometric.h

幾何学的計算・変換処理を行う

データ構造

- struct **hecmw_coord**
座標を表す構造体

関数

- double **HECMW_degree_to_radian** (double deg)
角度からラジアンへの変換を行う
 - double **HECMW_radian_to_degree** (double rad)
ラジアンから角度への変換を行う
 - int **HECMW_cylindrical_to_cartesian** (const struct **hecmw_coord** *coord, struct **hecmw_coord** *result)
円筒座標から直行デカルト座標への変換を行う
-

説明

幾何学的計算・変換処理を行う

日付:

2004年2月16日

作者:

坂根 一彰

関数

double HECMW_degree_to_radian (double *deg*)

角度からラジアンへの変換を行う

引数:

deg 角度

戻り値:

ラジアン

double HECMW_radian_to_degree (double *rad*)

ラジアンから角度への変換を行う

引数:

rad ラジアン

戻り値:

角度

int HECMW_cylindrical_to_cartesian (const struct hecmw_coord * *coord*, struct hecmw_coord * *result*)

円筒座標から直交デカルト座標への変換を行う

引数:

coord 変換元の円筒座標

result 変換後のデカルト座標格納領域

戻り値:

hecmw_gflex.h

GeoFEMメッシュデータの字句解析ルーチン.

```
#include <stdio.h>
```

列挙型

- enum { **HECMW_GFLEX_NL** = 1000, **HECMW_GFLEX_INT**,
HECMW_GFLEX_DOUBLE, **HECMW_GFLEX_NAME** }
トークン番号定義

関数

- int **HECMW_gflex_get_lineno** (void)
読み込んでいるファイルの現在の行番号を返す
- double **HECMW_gflex_get_number** (void)
直前に読み込んだトークンが数値であった場合、その値を返す
- char * **HECMW_gflex_get_text** (void)
直前に読み込んだトークンの文字列表記を返す
- int **HECMW_gflex_next_token** (void)
次のトークンを返す
- int **HECMW_gflex_next_token_skip** (int skip_token)
次のトークンを返す
- int **HECMW_gflex_set_input** (FILE *fp)
字句解析ルーチンで読み込むファイルのファイルポインタをセットする

- **int HECMW_gflex_skip_line** (void)

現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す

説明

GeoFEM メッシュデータの字句解析ルーチン.

日付:

2004年2月16日

作者:

坂根 一彰

列挙型

anonymous enum

トークン番号定義

列挙型の値:

HECMW_GFLEX_NL 改行文字

HECMW_GFLEX_INT 整数

HECMW_GFLEX_DOUBLE 浮動小数点

HECMW_GFLEX_NAME 名前

関数

int HECMW_gflex_get_lineno (void)

読み込んでいるファイルの現在の行番号を返す

戻り値:

行番号

double HECMW_gflex_get_number (void)

直前に読み込んだトークンが数値であった場合、その値を返す

戻り値:

直前に読み込まれた数値

char* HECMW_gflex_get_text (void)

直前に読み込んだトークンの文字列表記を返す

戻り値:

直前に読み込まれたトークンの文字列表記

int HECMW_gflex_next_token (void)

次のトークンを返す

EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_gflex_next_token_skip (int *skip_token*)

次のトークンを返す

*skip_token*で指定されたトークンの連続は読み飛ばされる。それ以外のトークンが現れたらそれを返す。EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_gflex_set_input (FILE * *fp*)

字句解析ルーチンで読み込むファイルのファイルポインタをセットする

引数:

fp 字句解析対象のファイルへのファイルポインタ

戻り値:

成功すれば0を返す。引数が無効なら-1を返す。

int HECMW_gflex_skip_line (void)

現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す

戻り値:

現在の行の最後のトークン番号。EOFに達した場合は0を返す。

hecmw_heclex.h

HEC-MW単一領域メッシュデータの字句解析ルーチン.

```
#include <stdio.h>
```

列挙型

- enum { **HECMW_HECLEX_NL** = 1000, **HECMW_HECLEX_INT**,
HECMW_HECLEX_DOUBLE, **HECMW_HECLEX_NAME**,
HECMW_HECLEX_FILENAME, **HECMW_HECLEX_HEADER**,
HECMW_HECLEX_H_AMPLITUDE = 2000, **HECMW_HECLEX_H_ECOPY**,
HECMW_HECLEX_H_EGEN, **HECMW_HECLEX_H_EGROUP**,
HECMW_HECLEX_H_ELEMENT, **HECMW_HECLEX_H_END**,
HECMW_HECLEX_H_EQUATION, **HECMW_HECLEX_H_HEADER**,
HECMW_HECLEX_H_INCLUDE, **HECMW_HECLEX_H_INITIAL**,
HECMW_HECLEX_H_ITEM, **HECMW_HECLEX_H_MATERIAL**,
HECMW_HECLEX_H_NCOPY, **HECMW_HECLEX_H_NFILL**,
HECMW_HECLEX_H_NGEN, **HECMW_HECLEX_H_NGROUP**,
HECMW_HECLEX_H_NODE, **HECMW_HECLEX_H_SECTION**,
HECMW_HECLEX_H_SGROUP, **HECMW_HECLEX_H_SYSTEM**,
HECMW_HECLEX_H_ZERO, **HECMW_HECLEX_K_ABSOLUTE** = 3000,
HECMW_HECLEX_K_BEAM, **HECMW_HECLEX_K_COMPOSITE**,
HECMW_HECLEX_K_DEFINITION, **HECMW_HECLEX_K_EGRP**,
HECMW_HECLEX_K_GENERATE, **HECMW_HECLEX_K_INPUT**,
HECMW_HECLEX_K_INTERFACE, **HECMW_HECLEX_K_ITEM**,
HECMW_HECLEX_K_MATERIAL, **HECMW_HECLEX_K_MATITEM**,
HECMW_HECLEX_K_NAME, **HECMW_HECLEX_K_NGRP**,
HECMW_HECLEX_K_RELATIVE, **HECMW_HECLEX_K_SECOPT**,
HECMW_HECLEX_K_SECTION, **HECMW_HECLEX_K_SGRP**,
HECMW_HECLEX_K_SHELL, **HECMW_HECLEX_K_SOLID**,
HECMW_HECLEX_K_STEP_TIME, **HECMW_HECLEX_K_SUBITEM**,
HECMW_HECLEX_K_SYSTEM, **HECMW_HECLEX_K_TABLE**,
HECMW_HECLEX_K_TABULAR, **HECMW_HECLEX_K_TEMPERATURE**,

**HECMW_HECLEX_K_TIME, HECMW_HECLEX_K_TYPE,
HECMW_HECLEX_K_VALUE }**

トークン番号定義

関数

- **double HECMW_heclex_get_number (void)**
直前に読み込んだトークンが数値であった場合、その値を返す
- **char * HECMW_heclex_get_text (void)**
直前に読み込んだトークンの文字列表記を返す
- **int HECMW_heclex_get_lineno (void)**
読み込んでいるファイルの現在の行番号を返す
- **int HECMW_heclex_next_token (void)**
次のトークンを返す
- **int HECMW_heclex_next_token_skip (int skip_token)**
次のトークンを返す
- **int HECMW_heclex_set_input (FILE *fp)**
字句解析ルーチンで読み込むファイルのファイルポインタをセットする
- **int HECMW_heclex_skip_line (void)**
現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す
- **int HECMW_heclex_switch_to_include (const char *filename)**
現在読み込み中の字句解析ルーチンを中断し、新たなファイル(インクルードファイル)を解析対象とする

- **int HECMW_heclex_unput_token** (void)
直前に読み込んだトークンを押し戻す
- **int HECMW_heclex_is_including** (void)
現在解析中のファイルがインクルードファイルか否かを返す

説明

HEC-MW 単一領域メッシュデータの字句解析ルーチン.

日付:

2004年2月16日

作者:

坂根 一彰

列挙型

anonymous enum

トークン番号定義

列挙型の値:

HECMW_HECLEX_NL 改行文字

HECMW_HECLEX_INT 整数

HECMW_HECLEX_DOUBLE 浮動小数点

HECMW_HECLEX_NAME 名前

HECMW_HECLEX_FILENAME ファイル名

HECMW_HECLEX_HEADER ヘッダデータ

HECMW_HECLEX_H_AMPLITUDE !AMPLITUDE

HECMW_HECLEX_H_ECOPY !ECOPY
HECMW_HECLEX_H_EGEN !EGEN
HECMW_HECLEX_H_EGROUP !EGROUP
HECMW_HECLEX_H_ELEMENT !ELEMENT
HECMW_HECLEX_H_END !END
HECMW_HECLEX_H_EQUATION !EQUATION
HECMW_HECLEX_H_HEADER !HEADER
HECMW_HECLEX_H_INCLUDE !INCLUDE
HECMW_HECLEX_H_INITIAL !INITIAL CONDITION
HECMW_HECLEX_H_ITEM !ITEM
HECMW_HECLEX_H_MATERIAL !MATERIAL
HECMW_HECLEX_H_NCOPY !NCOPY
HECMW_HECLEX_H_NFILL !NFILL
HECMW_HECLEX_H_NGEN !NGEN
HECMW_HECLEX_H_NGROUP !NGROUP
HECMW_HECLEX_H_NODE !NODE
HECMW_HECLEX_H_SECTION !SECTION
HECMW_HECLEX_H_SGROUP !SGROUP
HECMW_HECLEX_H_SYSTEM !SYSTEM
HECMW_HECLEX_H_ZERO !ZERO
HECMW_HECLEX_K_ABSOLUTE ABSOLUTE.
HECMW_HECLEX_K_BEAM BEAM.
HECMW_HECLEX_K_COMPOSITE COMPOSITE.
HECMW_HECLEX_K_DEFINITION DEFINITION.
HECMW_HECLEX_K_EGRP EGRP.
HECMW_HECLEX_K_GENERATE GENERATE.
HECMW_HECLEX_K_INPUT INPUT.
HECMW_HECLEX_K_INTERFACE INTERFACE.
HECMW_HECLEX_K_ITEM ITEM.
HECMW_HECLEX_K_MATERIAL MATERIAL.
HECMW_HECLEX_K_MATITEM MATITEM.
HECMW_HECLEX_K_NAME NAME.
HECMW_HECLEX_K_NGRP NGRP.
HECMW_HECLEX_K_RELATIVE RELATIVE.
HECMW_HECLEX_K_SECOPT SECOPT.
HECMW_HECLEX_K_SECTION SECTION.

HECMW_HECLEX_K_SGRP SGRP.
HECMW_HECLEX_K_SHELL SHELL.
HECMW_HECLEX_K_SOLID SOLID.
HECMW_HECLEX_K_STEP_TIME STEP TIME.
HECMW_HECLEX_K_SUBITEM SUBITEM.
HECMW_HECLEX_K_SYSTEM SYSTEM.
HECMW_HECLEX_K_TABLE TABLE.
HECMW_HECLEX_K_TABULAR TABULAR.
HECMW_HECLEX_K_TEMPERATURE TEMPERATURE.
HECMW_HECLEX_K_TIME TIME.
HECMW_HECLEX_K_TYPE TYPE.
HECMW_HECLEX_K_VALUE VALUE.

関数

double HECMW_heclex_get_number (void)

直前に読み込んだトークンが数値であった場合、その値を返す

戻り値:

直前に読み込まれた数値

char* HECMW_heclex_get_text (void)

直前に読み込んだトークンの文字列表記を返す

戻り値:

直前に読み込まれたトークンの文字列表記

int HECMW_hecllex_get_lineno (void)

読み込んでいるファイルの現在の行番号を返す

戻り値:

行番号

int HECMW_hecllex_next_token (void)

次のトークンを返す

EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_hecllex_next_token_skip (int *skip_token*)

次のトークンを返す

*skip_token*で指定されたトークンの連続は読み飛ばされる。 それ以外のトークンが現れたらそれを返す。 EOFに達した場合は0を返す

戻り値:

次のトークン番号

int HECMW_hecllex_set_input (FILE * *fp*)

字句解析ルーチンで読み込むファイルのファイルポインタをセットする

引数:

fp 字句解析対象のファイルへのファイルポインタ

戻り値:

成功すれば0を返す。 引数が無効なら-1を返す。

int HECMW_heclex_skip_line (void)

現在読み込み中の行を読み飛ばし、その行の最後のトークン番号を返す

戻り値:

現在の行の最後のトークン番号。 EOFに達した場合は0を返す。

int HECMW_heclex_switch_to_include (const char * *filename*)

現在読み込み中の字句解析ルーチンを中断し、新たなファイル(インクルードファイル)を解析対象とする

解析中のファイルの情報は保存される。 インクルードファイルへの切替えは、この関数の呼び出し直後から有効になり、 次に返されるトークンはインクルードファイルのものとなる。

インクルードファイルのトークンがなくなると、自動的に保存されていた 状態に切り替わり、 前のファイルの解析が再開される。 インクルードファイルの解析が終了したことを通知することはないが、 **HECMW_heclex_is_including()**によって、現在インクルードファイルを 読んでいるのか否かを知ることができる。

引数:

filename インクルードファイル名

戻り値:

成功すれば0を返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_hecllex_unput_token (void)

直前に読み込んだトークンを押し戻す

次回取得したトークンは、押し戻したトークンとなる.

戻り値:

常に0を返す.

int HECMW_hecllex_is_including (void)

現在解析中のファイルがインクルードファイルか否かを返す

戻り値:

インクルードファイル解析中なら1を返し、そうでなければ0を返す

hecmw_init.h

HEC-MWの全体初期化処理を行う。

関数

- `int HECMW_init (int *argc, char ***argv)`

HEC-MWの初期化処理を行う。

説明

HEC-MW の全体初期化処理を行う。

全体初期化処理では、以下の処理が行われる。

- MPI初期化処理
- 全体制御ファイル入力処理

HEC-MW のライブラリは、MPI を利用した並列処理を行うように作成されているため、プログラムの最初にこの初期化処理を必ず行う必要がある。

また、全体制御ファイル入力処理によって、HEC-MW の実行制御に必要な 情報が HEC-MW 全体制御ファイルから取得され、保持される。

この初期化処理を行った場合、プログラムの最後で `HECMW_finalize()` による終了処理を行わなくてはならない。

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_init (int * *argc*, char * *argv*)**

HEC-MWの初期化処理を行う.

この関数によって初期化が正常終了して初めてHEC-MWのライブラリが 正常に動作するようになる.

参照:

HECMW_finalize()

引数:

argc main関数の*argc*引数へのポインタ

argv main関数の*argv*引数へのポインタ

戻り値:

成功すれば0を返す. 失敗した場合は0以外の値を返し, `hecmw_set_error()`によってエラー情報がセットされる.

hecmw_io.h

IO関連のヘッダをインクルード.

```
#include "hecmw_geometric.h"
#include "hecmw_ablex.h"
#include "hecmw_gflex.h"
#include "hecmw_dist.h"
#include "hecmw_dist_copy_f2c.h"
#include "hecmw_dist_free.h"
#include "hecmw_dist_print.h"
#include "hecmw_hecllex.h"
#include "hecmw_io_dist.h"
#include "hecmw_io_geofem.h"
#include "hecmw_io_get_mesh.h"
#include "hecmw_io_hec.h"
#include "hecmw_io_abaqus.h"
#include "hecmw_io_mesh.h"
#include "hecmw_io_put_mesh.h"
#include "hecmw_io_struct.h"
#include "hecmw_restart.h"
#include "hecmw_result.h"
#include "hecmw_result_copy_c2f.h"
#include "hecmw_system.h"
```

説明

IO 関連のヘッダをインクルード.

削除予定

hecmw_io_abaqus.h

ABAQUSメッシュデータ入力ルーチン.

```
#include <stdio.h>
```

```
#include "hecmw_struct.h"
```

関数

- **int HECMW_read_abaqus_mesh** (const char *filename)
ABAQUSメッシュデータを読み込む.
- **hecmwST_local_mesh * HECMW_get_abaqus_mesh** (const char *filename)
ABAQUSメッシュデータを読み込み, メッシュデータ構造体を返す.

説明

ABAQUSメッシュデータ入力ルーチン.

ABAQUSメッシュデータを読み込む.

参照:

ABQUSメッシュデータ入力機能

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_read_abaqus_mesh (const char * *filename*)

ABAQUSメッシュデータを読み込む。

入力処理を行うのみで、メッシュデータ構造体は生成しない。入力されたメッシュデータは保持されたままである。

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmwST_local_mesh* HECMW_get_abaqus_mesh (const char * *filename*)

ABAQUSメッシュデータを読み込み、メッシュデータ構造体を返す。

入力処理を行い、その結果得られたデータからメッシュデータ構造体を生成して返す。

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すればメッシュデータ構造体を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

hecmw_io_dist.h

HEC-MW分散メッシュデータ入出力ルーチン.

```
#include "hecmw_struct.h"
```

関数

- **hecmwST_local_mesh * HECMW_get_dist_mesh** (char *fname)
HEC-MW分散メッシュデータを読み込む.
- **int HECMW_put_dist_mesh** (const struct **hecmwST_local_mesh** *mesh, const char *fname)
HEC-MW分散メッシュデータを出力する.

説明

HEC-MW 分散メッシュデータ入出力ルーチン.

HEC-MW 分散メッシュデータを入力・出力する.

日付:

2004年2月16日

作者:

坂根 一彰

関数

struct hecmwST_local_mesh* HECMW_get_dist_mesh (char * *fname*)

HEC-MW分散メッシュデータを読み込む.

入力処理を行い，その結果得られたデータからメッシュデータ構造体を生成して返す．

引数:

fname メッシュファイル名(パス含む)

戻り値:

成功すればメッシュデータ構造体を返す．失敗すればNULLを返し，HECMW_set_error()によってエラー情報がセットされる．

int HECMW_put_dist_mesh (const struct hecmwST_local_mesh * *mesh*, const char * *fname*)

HEC-MW分散メッシュデータを出力する．

メッシュデータ構造体の内容をファイルに出力する．

引数:

mesh 出力するメッシュデータ構造体

fname メッシュファイル名(パス含む)

戻り値:

成功すれば0を返す．失敗すれば-1を返し，HECMW_set_error()によってエラー情報がセットされる．

hecmw_io_geofem.h

GeoFEMメッシュデータ入力ルーチン.

```
#include <stdio.h>
```

```
#include "hecmw_util.h"
```

関数

- **int HECMW_read_geofem_mesh** (const char *filename)
GeoFEMメッシュデータを読み込む.
- **hecmwST_local_mesh * HECMW_get_geofem_mesh** (const char *filename)
GeoFEMメッシュデータを読み込み, メッシュデータ構造体を返す.

説明

GeoFEMメッシュデータ入力ルーチン.

GeoFEMメッシュデータを読み込む. 対応するのは初期メッシュデータのみで, 分散メッシュデータには対応しない.

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_read_geofem_mesh (const char * *filename*)

GeoFEMメッシュデータを読み込む。

入力処理を行うのみで、メッシュデータ構造体は生成しない。 入力されたメッシュデータは保持されたままである。

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すれば0を返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmwST_local_mesh* HECMW_get_geofem_mesh (const char * *filename*)

GeoFEMメッシュデータを読み込み、メッシュデータ構造体を返す。

入力処理を行い、その結果得られたデータからメッシュデータ構造体を生成して返す。

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すればメッシュデータ構造体を返す。 失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

hecmw_io_get_mesh.h

メッシュデータ取得ルーチン

```
#include "hecmw_struct.h"
```

関数

- **hecmwST_local_mesh * HECMW_get_mesh** (char *name_ID)
メッシュデータを読み込み, メッシュデータ構造体に格納して返す

説明

メッシュデータ取得ルーチン

メッシュファイルを読み込み, メッシュデータを取得する. 取得したメッシュデータは, メッシュデータ構造体(**hecmwST_local_mesh**)に 格納されて返される.

メッシュファイルの定義は, 全体制御ファイルにて行う. メッシュデータ取得ルーチンは, 全体制御ファイルで定義される情報をもとに 読み込むべきメッシュファイルを決するため, その定義は必須である.

全体制御ファイルでメッシュファイルを定義するには, **!MESH** と **!MESH GROUP** を使用する. それぞれの定義において **NAME** パラメータの定義が必須になっており, それが読み込み対象とするメッシュファイルの判別に使用される.

メッシュデータ取得ルーチンによって読み込み可能なメッシュのタイプを以下に示す.

- HEC-MW分散メッシュデータ
- HEC-MW単一領域メッシュデータ
- GeoFEM単一領域メッシュデータ
- ABAQUSメッシュデータ
- NASTRANメッシュデータ(現版では未対応)
- FEMAPメッシュデータ(現版では未対応)

それぞれのメッシュタイプの定義は、全体制御ファイルにて行う。メッシュデータ取得ルーチンは、読み込むメッシュのタイプを!MESH の定義から自動的に判別するため、そのルーチン使用時にメッシュタイプを考慮する必要はない。また、これらのメッシュデータは、同一のメッシュデータ取得ルーチンによって取得可能であり、取得した結果得られる構造体は、全てメッシュデータ構造体が使用される。このように、メッシュタイプによって、メッシュの取得方法や、得られるデータ構造が異なることはなく、全て統一的に操作が可能である。グループ化されたメッシュを読み込み対象として指定した場合、メッシュデータ取得ルーチンは、その定義によってグループ化されている複数のファイルを同時に読み込み、一つのメッシュデータ構造体を生成する。これは、メッシュファイルが複数に分割されている場合や、異なるタイプのメッシュを混在させたい場合に便利である。

参照:

HECMW_get_mesh()

日付:

2004年2月16日

作者:

坂根 一彰

関数

struct hecmwST_local_mesh* HECMW_get_mesh (char * *name_ID*)

メッシュデータを読み込み、メッシュデータ構造体に格納して返す

読み込み対象となるメッシュファイル名は、全体制御ファイルから取得する。

引数:

name_ID 読み込み対象となるメッシュを識別するための、!MESHまたは!MESH GROUPにおけるNAMEパラメータ

戻り値:

成功すれば、読み込んだメッシュデータを格納したメッシュデータ構造体を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

hecmw_io_hec.h

HEC-MW単一領域メッシュデータ入力ルーチン.

```
#include <stdio.h>
#include "hecmw_struct.h"
```

関数

- **int HECMW_read_entire_mesh** (const char *filename)
HEC-MW単一領域メッシュデータを読み込む.
- **hecmwST_local_mesh * HECMW_get_entire_mesh** (const char *filename)
HEC-MW単一領域メッシュデータを読み込み, メッシュデータ構造体を返す.

説明

HEC-MW 単一領域メッシュデータ入力ルーチン.
HEC-MW 単一領域メッシュデータを読み込む.

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_read_entire_mesh (const char * *filename*)

HEC-MW単一領域メッシュデータを読み込む.

入力処理を行うのみで、メッシュデータ構造体は生成しない。 入力されたメッシュデータは保持されたままである.

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すれば0を返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる.

struct hecmwST_local_mesh* HECMW_get_entire_mesh (const char * *filename*)

HEC-MW単一領域メッシュデータを読み込み、メッシュデータ構造体を返す.

入力処理を行い、その結果得られたデータからメッシュデータ構造体を生成して返す.

引数:

filename メッシュファイル名(パス含む)

戻り値:

成功すればメッシュデータ構造体を返す。 失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる.

hecmw_io_mesh.h

全体メッシュデータの取得とメッシュデータ構造体生成

```
#include <stdio.h>
#include "hecmw_util.h"
#include "hecmw_io_struct.h"
#include "hecmw_system.h"
```

関数

- **int HECMW_io_get_version** (void)
#HECMW_FLAG_VERSIONを返す
- **int HECMW_io_init** (void)
初期化処理を行う.
- **int HECMW_io_finalize** (void)
終了処理を行う
- **void HECMW_io_print_all** (FILE *fp)
現在貯め込んでいるメッシュ情報をfpに出力する
- **int HECMW_io_free_all** (void)
現在貯め込んでいるメッシュ情報のメモリ領域を解放する
- **int HECMW_io_set_gridfile** (char *gridfile)
gridfileをセットする
- **hecmw_io_amplitude * HECMW_io_add_amp** (const char *name, int definition, int time, int value, double val, double t)
AMPLITUDE情報を追加する.

- **hecmw_io_initial * HECMW_io_get_initial** (int node)
節点の初期条件を取得する
- **hecmw_io_initial * HECMW_io_add_initial** (int type, int node, const char *ngrp, double val)
初期条件を追加する
- **hecmw_io_element * HECMW_io_get_elem** (int id)
要素情報を取得する
- **int HECMW_io_get_n_elem** (void)
現在の保持されている要素数を取得する
- **int HECMW_io_get_elem_max_id** (void)
最大のグローバル要素番号を取得する
- **hecmw_io_element * HECMW_io_add_elem** (int id, int type, int *node, int nmatitem, double *matitem)
要素を追加する
- **hecmw_io_id_array * HECMW_io_get_elem_in_egrp** (const char *name)
要素グループに含まれる要素を取得する
- **hecmw_io_egrp * HECMW_io_get_egrp** (const char *name)
要素グループを取得する
- **int HECMW_io_add_egrp** (const char *name, int nelem, int *elem)
要素グループを追加する
- **hecmw_io_node * HECMW_io_get_node** (int id)
節点を取得する

- **int HECMW_io_get_n_node** (void)
現在の節点数を取得する
- **hecmw_io_node * HECMW_io_add_node** (int id, double x, double y, double z)
節点を追加する
- **int HECMW_io_get_nnode_in_ngrp** (const char *name)
節点グループに含まれる節点数を取得する
- **int HECMW_io_remove_node** (int id)
節点を削除する
- **hecmw_io_ngrp * HECMW_io_get_ngrp** (const char *name)
節点グループを取得する
- **hecmw_io_id_array * HECMW_io_get_node_in_ngrp** (const char *name)
節点グループに含まれる節点を取得する
- **int HECMW_io_add_ngrp** (const char *name, int nnode, int *node)
節点グループを追加する
- **int HECMW_io_add_sgrp** (const char *name, int n, int *elem, int *surf)
面グループを追加する
- **hecmw_io_mpc * HECMW_io_add_mpc** (int neq, const struct hecmw_io_mpcitem *mpcitem)
拘束グループ情報を追加する
- **hecmw_io_section * HECMW_io_add_sect** (struct hecmw_io_section *sect)
セクション情報を追加する
- **hecmw_io_material * HECMW_io_get_mat** (const char *name)

材料情報を取得する

- **hecmw_io_material * HECMW_io_add_mat** (struct **hecmw_io_material** *mat)

材料情報を追加する

- **void HECMW_io_set_header** (struct **hecmw_io_header** *header)

ヘッダをセットする

- **hecmw_system_param * HECMW_io_get_system** (void)

局所座量から全体座標への変換パラメータを取得する

- **void HECMW_io_set_system** (struct **hecmw_system_param** *system)

局所座量から全体座標への変換パラメータをセットする

- **void HECMW_io_set_zero** (struct **hecmw_io_zero** *zero)

絶対零度をセットする

- **int HECMW_io_post_process** (void)

入力終了後の後処理を行う

- **int HECMW_io_pre_process** (void)

入力ルーチンの前処理を行う

- **int HECMW_io_check_mpc_dof** (int dof)

拘束グループの自由度をチェックする

- **int HECMW_io_is_reserved_name** (const char *name)

名前が予約済みでないかどうかをチェックする

- **hecmwST_local_mesh * HECMW_io_make_local_mesh** (void)

メッシュデータ構造体を生成する

説明

全体メッシュデータの取得とメッシュデータ構造体生成
読み込まれた全体メッシュデータは本ルーチンを通じて蓄えられ、メッシュデータ構造体へと変換される。
HECMW_io_get_*(),HECMW_io_set_*(),HECMW_io_add_*() は全て一時領域のメッシュデータに対するものである。

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_io_get_version (void)

#HECMW_FLAG_VERSIONを返す

戻り値:

#HECMW_FLAG_VERSION

int HECMW_io_init (void)

初期化処理を行う。

既にデータが蓄えられていれば、それらを解放し、初期化する。

戻り値:

成功すれば0を返す。失敗すれば-1を返す。

int HECMW_io_finalize (void)

終了処理を行う

蓄えられているデータのメモリ領域は解放される。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

void HECMW_io_print_all (FILE * fp)

現在貯め込んでいるメッシュ情報をfpに出力する

引数:

fp 出力先ファイルポインタ

int HECMW_io_free_all (void)

現在貯め込んでいるメッシュ情報のメモリ領域を解放する

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_io_set_gridfile (char * *gridfile*)

gridfileをセットする

引数:

gridfile ファイル名. NULLは空文字列とみなされる.

戻り値:

常に成功し, 0を返す.

struct hecmw_io_amplitude* HECMW_io_add_amp (const char * *name*, int *definition*, int *time*, int *value*, double *val*, double *t*)

AMPLITUDE情報を追加する.

引数:

name AMPLITUDE名

definition AMPLITUDEタイプ

time 時刻の種類

value 値の種類

val 時刻*t*における値

t 時刻*t*

戻り値:

成功すれば, 追加したAMPLITUDE情報を返す. 失敗すればNULLを返し,
HECMW_set_error()によってエラー情報がセットされる.

struct hecmw_io_initial* HECMW_io_get_initial (int *node*)

節点の初期条件を取得する

引数:

node グローバル節点番号

戻り値:

節点の初期条件が見付かればその情報を返す。 見付からない場合はNULLを返す。

**struct hecmw_io_initial* HECMW_io_add_initial (int *type*, int *node*, const char *
ngrp, double *val*)**

初期条件を追加する

節点の指定方法は、グローバル節点番号を指定する場合と、 節点グループを指定してそれに含まれる全ての節点を指定する場合がある。 引数`ngrp`がNULLでなければ節点グループ名による指定とみなされ、 `ngrp`がNULLの場合は、節点番号による指定とみなされる。

引数:

type 初期条件の種類

node 初期条件を与える節点のグローバル節点番号

ngrp 初期条件を与える節点の節点グループ名

val 初期条件の値

戻り値:

成功すれば追加した初期条件情報を返す。 失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmw_io_element* HECMW_io_get_elem (int *id*)

要素情報を取得する

引数で指定されたグローバル要素番号を持つ要素の情報を取得する

引数:

id グローバル要素番号

戻り値:

グローバル要素番号`id`を持つ要素が見付かれればその情報を返す. 見付からなければ
NULLを返す.

int HECMW_io_get_n_elem (void)

現在の保持されている要素数を取得する

int HECMW_io_get_elem_max_id (void)

最大のグローバル要素番号を取得する

**struct hecmw_io_element* HECMW_io_add_elem (int *id*, int *type*, int * *node*, int
nmatitem, double * *matitem*)**

要素を追加する

引数:

id グローバル要素番号

type 要素タイプ

node コネクティビティ

nmatitem 材料物性値の個数

matitem 材料物性値

戻り値:

成功すれば, 追加した要素情報を返す. 失敗すればNULLを返し, HECMW_set_error()
によってエラー情報がセットされる.

struct hecmw_io_id_array* HECMW_io_get_elem_in_egrp (const char * *name*)

要素グループに含まれる要素を取得する

引数:

name 要素グループ名

戻り値:

成功すれば、要素グループに含まれる全要素のグローバル要素番号を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。 *name*に対応する要素グループが存在しない場合もNULLを返すが、エラー情報をセットしない。区別がつかないので、要修正。

struct hecmw_io_egrp* HECMW_io_get_egrp (const char * *name*)

要素グループを取得する

引数:

name 要素グループ名

戻り値:

成功すれば、*name*で示される名前の要素グループ情報を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_io_add_egrp (const char * *name*, int *nelem*, int * *elem*)

要素グループを追加する

*name*で示される名前の要素グループを追加し、それに含まれる要素を登録する。 *name*に対応する要素グループが存在しない場合は要素グループが新規に作成され、既に存在する場合は、既存グループに要素が追加される。

引数:

name 要素グループ名

nelem 追加する要素数

elem 追加する要素のグローバル要素番号

戻り値:

成功すれば追加した要素数を返す. 失敗すれば-1を返し, `HECMW_set_error()`によってエラー情報がセットされる.

struct hecmw_io_node* HECMW_io_get_node (int *id*)

節点を取得する

引数:

id グローバル節点番号

戻り値:

対応する節点があれば, その情報を返す. そうでなければNULLを返す.

int HECMW_io_get_n_node (void)

現在の節点数を取得する

戻り値:

現在の節点数

struct hecmw_io_node* HECMW_io_add_node (int *id*, double *x*, double *y*, double *z*)

節点を追加する

引数:

id グローバル節点番号
x X座標値
y Y座標値
z Z座標値

戻り値:

成功すれば追加した節点情報を返す. 失敗すればNULLを返し, `HECMW_set_error()`によってエラー情報がセットされる.

int HECMW_io_get_nnode_in_ngrp (const char * *name*)

節点グループに含まれる節点数を取得する

引数:

name 節点グループ名

戻り値:

成功すれば, 節点グループに含まれる要素の個数を返す. 失敗すれば-1を返し, `HECMW_set_error()`によってエラー情報がセットされる.

int HECMW_io_remove_node (int *id*)

節点を削除する

引数:

id 削除する節点のグローバル節点番号

戻り値:

成功すれば, 削除した件数(1)を返す. 削除する節点が見付からなければ0を返す.

struct hecmw_io_ngrp* HECMW_io_get_ngrp (const char * *name*)

節点グループを取得する

引数:

name 節点グループ名

戻り値:

*name*で指定された名前を持つ節点グループ情報を返す。見付からなければNULLを返す。

struct hecmw_io_id_array* HECMW_io_get_node_in_ngrp (const char * *name*)

節点グループに含まれる節点を取得する

引数:

name 節点グループ名

戻り値:

*name*に対応する節点グループに含まれる全節点のグローバル節点番号を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。*name*に対応する節点グループが存在しない場合もNULLを返すが、エラー情報をセットしない。区別がつかないので、要修正。

int HECMW_io_add_ngrp (const char * *name*, int *nnode*, int * *node*)

節点グループを追加する

*name*で示される名前の節点グループを追加し、それに含まれる節点を登録する。*name*に対応する節点グループが存在しない場合は節点グループが新規に作成され、既に存在する場合は、既存グループに節点が追加される。

引数:

name 節点グループ名

nnode 追加する節点数

node 追加する節点のグローバル節点番号

戻り値:

成功すれば追加した節点数を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_io_add_sgrp (const char * *name*, int *n*, int * *elem*, int * *surf*)

面グループを追加する

*name*で示される名前の面グループを追加し、それに含まれる要素と面番号のペアを登録する。
*name*に対応する面グループが存在しない場合は面グループが新規に作成され、既に存在する場合は、既存グループに要素と面番号のペアが追加される。

引数:

name 面グループ名

n 追加する要素と面番号のペア

elem 追加する要素のグローバル要素番号

surf 追加する要素の面番号

戻り値:

成功すれば追加した節点数を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmw_io_mpc* HECMW_io_add_mpc (int *neq*, const struct hecmw_io_mpcitem * *mpcitem*)

拘束グループ情報を追加する

引数:

neq 方程式の項数

mpcitem 拘束グループ情報. 必要な情報はここからコピーされる.

戻り値:

成功すれば, 追加した拘束グループ情報を返す. 失敗すればNULLを返し,

HECMW_set_error()によってエラー情報がセットされる.

struct hecmw_io_section* HECMW_io_add_sect (struct hecmw_io_section * sect)

セクション情報を追加する

引数:

sect 追加するセクション情報. 必要な情報はここからコピーされる.

成功すれば, 追加したセクション情報を返す. 失敗すればNULLを返し,

HECMW_set_error()によってエラー情報がセットされる.

struct hecmw_io_material* HECMW_io_get_mat (const char * name)

材料情報を取得する

引数:

name 材料名

戻り値:

*name*に対応する材料情報が見付かれればその情報を返す. 見付からなければNULLを返す.

**struct hecmw_io_material* HECMW_io_add_mat (struct hecmw_io_material *
mat)**

材料情報を追加する

引数:

mat 材料情報. この情報はそのまま追加されるため、領域を解放したり内容を変更してはならない.

戻り値:

成功すれば、追加した材料情報を返す. 失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる.

void HECMW_io_set_header (struct hecmw_io_header * *header*)

ヘッダをセットする

引数:

header ヘッダ情報. この情報はそのまま追加されるため、領域を解放したり内容を変更してはならない.

struct hecmw_system_param* HECMW_io_get_system (void)

局所座量から全体座標への変換パラメータを取得する

戻り値:

局所座量から全体座標への変換パラメータを返す. なければNULLを返す.

void HECMW_io_set_system (struct hecmw_system_param * *system*)

局所座量から全体座標への変換パラメータをセットする

引数:

system 局所座量から全体座標への変換パラメータ この情報はそのまま追加されるため、領域を解放したり内容を変更してはならない.

void HECMW_io_set_zero (struct hecmw_io_zero * zero)

絶対零度をセットする

引数:

zero 絶対零度情報 この情報はそのまま追加されるため、領域を解放したり内容を変更してはならない.

int HECMW_io_post_process (void)

入力終了後の後処理を行う

入力が終了したものととして、現在保持されている情報に対し後処理を実行する.

int HECMW_io_pre_process (void)

入力ルーチンの前処理を行う

入力を実行する前に実行しなければならない. ただし、現在は何も行わない.

int HECMW_io_check_mpc_dof (int dof)

拘束グループの自由度をチェックする

引数:

dof チェックする自由度

戻り値:

正しければ0を返し，そうでなければ-1を返す.

int HECMW_io_is_reserved_name (const char * *name*)

名前が予約済みでないかどうかをチェックする

HECMWで始まる名前は予約されている.

引数:

name チェックする名前

戻り値:

予約済であれば1を返し，そうでなければ0を返す.

struct hecmwST_local_mesh* HECMW_io_make_local_mesh (void)

メッシュデータ構造体を生成する

戻り値:

成功すればメッシュデータ構造体を返す. 失敗すればNULLを返し, HECMW_set_error() によってエラー情報がセットされる.

hecmw_io_put_mesh.h

メッシュデータをファイルに出力する

```
#include "hecmw_struct.h"
```

関数

- **int HECMW_put_mesh** (struct **hecmwST_local_mesh** *mesh, char *name_ID)
メッシュデータをファイルに出力する

説明

メッシュデータをファイルに出力する

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_put_mesh (struct **hecmwST_local_mesh** * *mesh*, char * *name_ID*)

メッシュデータをファイルに出力する

メッシュデータを分散メッシュデータとして出力する。領域が1つであっても、分散メッシュデータとして出力される。出力されるファイルは、全体制御ファイルの!MESHのうち、NAMEがname_IDのものとなる。出力されるファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなる。

引数:

mesh 出力メッシュデータ

name_ID メッシュファイル識別子

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

hecmw_io_struct.h

メッシュデータ入力時に使用するデータ構造

```
#include "hecmw_config.h"
#include "hecmw_set_int.h"
```

データ構造

- struct **hecmw_io_id_array**
int型IDの配列を表す
- struct **hecmw_io_id**
int型IDの単方向リンクリストを表す
- struct **hecmw_io_header**
ヘッダ情報
- struct **hecmw_io_zero**
絶対零度情報
- struct **hecmw_io_node**
節点情報
- struct **hecmw_io_element**
要素情報
- struct **hecmw_io_ngrp**
節点グループ情報
- struct **hecmw_io_egrp**

要素グループ情報

- struct **hecmw_io_elem_surf**
面グループアイテム
- struct **hecmw_io_sgrp**
面グループ情報
- struct **hecmw_io_mpc**
拘束グループ情報
- struct **hecmw_io_mpc::hecmw_io_mpcitem**
拘束グループアイテム
- struct **hecmw_io_amplitude**
AMPLITUDE 情報.
- struct **hecmw_io_amplitude::hecmw_io_amplitude_item**
AMPLITUDE アイテム.
- struct **hecmw_io_initial**
初期条件情報
- struct **hecmw_io_material**
材料情報
- struct **hecmw_io_material::hecmw_io_matitem**
材料ITEM
- struct **hecmw_io_material::hecmw_io_matitem::hecmw_io_matsubitem**
材料SUBITEM

- struct **hecmw_io_section**
セクション情報
- union **hecmw_io_section::hecmw_io_section_item**
セクションアイテム
- struct **hecmw_io_section::hecmw_io_section_item::hecmw_io_section_solid**
ソリッドセクション
- struct **hecmw_io_section::hecmw_io_section_item::hecmw_io_section_shell**
シェルセクション
- struct **hecmw_io_section::hecmw_io_section_item::hecmw_io_section_interface**
インタフェースセクション

マクロ定義

- #define **HECMW_INITIAL_TYPE_TEMPERATURE** 1

説明

メッシュデータ入力時に使用するデータ構造

マクロ定義

- #define **HECMW_INITIAL_TYPE_TEMPERATURE** 1

hecmw_lib_fc.h

FortranとC間のデータ変換ユーティリティ.

関数

- char * **HECMW_strepy_f2c** (const char *fstr, int flen)
Fortran 文字列をC文字列に変換する.
- char * **HECMW_strepy_f2c_r** (const char *fstr, int flen, char *buf, int bufsize)
Fortran 文字列をC文字列に変換する.
- int **HECMW_strepy_c2f** (const char *cstr, char *fstr, int flen)
C文字列をFortran文字列列に変換する.

説明

Fortran と C 間のデータ変換ユーティリティ.

日付:

2004年2月16日

作者:

坂根 一彰

関数

char* **HECMW_strcpy_f2c** (const char * *fstr*, int *flen*)

Fortran文字列をC文字列に変換する.

変換された文字列の領域は動的に割り当てられる.

引数:

fstr Fortran文字列

flen Fortran文字列の長さ

戻り値:

C文字列

char* HECMW_strcpy_f2c_r (const char * *fstr*, int *flen*, char * *buf*, int *bufsize*)

Fortran文字列をC文字列に変換する.

変換された文字列の領域は引数で与える

引数:

fstr Fortran文字列

flen Fortran文字列の長さ

buf C文字列格納用領域

bufsize bufの大きさ

戻り値:

C文字列

int HECMW_strcpy_c2f (const char * *cstr*, char * *fstr*, int *flen*)

C文字列をFortran文字列に変換する.

引数:

cstr C文字列

fstr Fortran文字列

flen Fortran文字列の長さ

戻り値:

変換後のFortran文字列の長さ

hecmw_log.h

ログ出力を可能とするための関数ライブラリ

```
#include <stdarg.h>
```

マクロ定義

- `#define HECMW_LOG_MAX 10`
指定可能なログファイルの上限
- `#define HECMW_LOG_NONE 0`
ログレベルを指定しない
- `#define HECMW_LOG_ERROR 1`
エラーを示すログレベル
- `#define HECMW_LOG_WARN 2`
警告を示すログレベル
- `#define HECMW_LOG_INFO 4`
情報を示すログレベル
- `#define HECMW_LOG_DEBUG 8`
デバッグを示すログレベル
- `#define HECMW_LOG_ALL (HECMW_LOG_ERROR|HECMW_LOG_WARN|HECMW_LOG_INFO|HECMW_LOG_DEBUG)`
全てのログレベルを包含する

- `#define HECMW_LOG_PERROR 1`
標準エラー出力に出力する
- `#define HECMW_LOG_OPTALL (HECMW_LOG_PERROR)`
全てのオプションを包含する

関数

- `int HECMW_openlog (const char *logfile, int loglv, int options)`
ログファイルを開く
- `int HECMW_closelog (const char *logfile)`
ログファイルを閉じる
- `int HECMW_vlog (int loglv, const char *fmt, va_list ap)`
ログを出力する
- `int HECMW_log (int loglv, const char *fmt,...)`
ログを出力する
- `void HECMW_setloglv (int loglv)`
出力ログレベルの指定

説明

ログ出力を可能とするための関数ライブラリ
システムやアプリケーションが必要とするさまざまな情報を出力可能とする。
ログは、複数のファイルに出力することが可能である。出力先は基本的にファイルであるが、オプションにより標準エラー出力を追加することができる。また、出力先を全く指定しない場合は標準エラー出力に出力される。

ログには以下の 4 つのログレベルがあり、出力先ごとにそのファイルが受け付けるログレベルを指定可能である。

- **HECMW_LOG_ERROR**
- **HECMW_LOG_WARN**
- **HECMW_LOG_INFO**
- **HECMW_LOG_DEBUG**

また、次のログレベルは定義可能なログレベル全てを含んでいる。

- **HECMW_LOG_ALL**

ログレベルは **HECMW_LOG_ERROR** が一番高く、**HECMW_LOG_WARN**,**HECMW_LOG_INFO**,**HECMW_LOG_DEBUG** の順に低くなる。

ログレベルは、

- ログファイルに出力するメッセージ
- ログファイル
- ログ出力レベル

の 3 つに対して設定する。

メッセージに対するログレベル設定では、メッセージそのもののレベルを定義する。

一方、ログファイルに対するレベル設定では、出力先のログファイルがどのレベルのログを受け付けるかを指定する。このログレベルには複数のレベルを設定することが可能であり、必要なレベルのログだけを受け付けるようにすることができる。

また、ログ出力レベルでは、ログ出力関数が出力を許可するログレベルを設定する。指定するレベルは、出力を許可するレベルのうち、最も低いレベルである。これは、前述の 2 つのログレベルより優先される。従って、ログ出力レベルで許可されていないレベルのログが出力されることはない。デフォルトの出力レベルは **HECMW_LOG_INFO** である。そのため、**HECMW_LOG_DEBUG** レベルのログはデフォルトでは出力されない。また、特殊なログレベル **HECMW_LOG_NONE** を指定すると、全てのログ出力を抑制できる。出力レベルは動的に変更可能である。

ログの出力先が全く指定されていない場合、標準エラー出力へ送られる。この場合、出力されるログレベルは、ログ出力ルーチンのログレベル設定によって決まる。また、ログファイルに対するロックは行わないため、同一ログファ

イルに同時に書き込みが行われた場合、その動作は不定である。

ログの出力形式は、

```
month day hh:mm:ss loglv: message
```

となる。

注意事項として、現状のログ操作関数の実装では、並列環境における排他制御を行っていないため、同一ログファイルに複数プロセスが同時アクセスした場合の結果は不定である。

日付:

2004年2月16日

作者:

坂根 一彰

マクロ定義

#define HECMW_LOG_MAX 10

指定可能なログファイルの上限

#define HECMW_LOG_NONE 0

ログレベルを指定しない

#define HECMW_LOG_ERROR 1

エラーを示すログレベル

#define HECMW_LOG_WARN 2

警告を示すログレベル

#define HECMW_LOG_INFO 4

情報を示すログレベル

#define HECMW_LOG_DEBUG 8

デバッグを示すログレベル

#define

**HECMW_LOG_ALL (HECMW_LOG_ERROR|HECMW_LOG_WARN|HECMW_LOG
_INFO|HECMW_LOG_DEBUG)**

全てのログレベルを包含する

#define HECMW_LOG_PERROR 1

標準エラー出力に出力する

#define HECMW_LOG_OPTALL (HECMW_LOG_PERROR)

全てのオプションを包含する

関数

int HECMW_openlog (const char * *logfile*, int *loglv*, int *options*)

ログファイルを開く

引数:

logfile ログファイル名

loglv このログファイルに対して出力を許可するログレベルの論理和

options オプション

戻り値:

成功すれば0, 失敗すれば-1を返す.

int HECMW_closelog (const char * *logfile*)

ログファイルを閉じる

指定されたログファイルを閉じる.

引数:

logfile ログファイル名

戻り値:

成功すれば0, 失敗すれば-1を返す.

int HECMW_vlog (int *loglv*, const char * *fmt*, va_list *ap*)

ログを出力する

ログファイルがオープンされていない場合, 標準エラー出力に出力される.

引数:

loglv このメッセージのログレベル

fmt フォーマット

ap 可変引数リスト

戻り値:

成功すれば0, 失敗すれば-1を返す.

int HECMW_log (int *loglv*, const char * *fmt*, ...)

ログを出力する

ログファイルがオープンされていない場合, 標準エラー出力に出力される.

引数:

loglv このメッセージのログレベル

fmt フォーマット

... 可変引数

戻り値:

成功すれば0, 失敗すれば-1を返す.

void HECMW_setloglv (int *loglv*)

出力ログレベルの指定

引数で指定されたレベルを持つログのみが出力対象となる. `HECMW_LOG_NONE`を指定すると, すべてのログ出力を抑制できる. なお, デフォルトで許可されているログレベルは以下のとおりである.

- `HECMW_LOG_ERROR`
- `HECMW_LOG_WARN`
- `HECMW_LOG_INFO`

引数:

loglv 出力を許可するログレベルの論理和

hecmw_malloc.h

メモリリーク検出ライブラリ

```
#include <stdlib.h>
```

```
#include <string.h>
```

マクロ定義

- #define **HECMW_malloc**(size) malloc(size)
- #define **HECMW_calloc**(nmemb, size) calloc(nmemb, size)
- #define **HECMW_realloc**(ptr, size) realloc(ptr, size)
- #define **HECMW_strdup**(s) strdup(s)
- #define **HECMW_free**(ptr) free(ptr)

関数

- void * **HECMW_malloc_** (size_t size, char *file, int line)
malloc() のラップ関数
- void * **HECMW_calloc_** (size_t nmemb, size_t size, char *file, int line)
calloc() のラップ関数
- void * **HECMW_realloc_** (void *ptr, size_t size, char *file, int line)
realloc() のラップ関数
- char * **HECMW_strdup_** (const char *s, char *file, int line)
strdup() のラップ関数
- void **HECMW_free_** (void *ptr, char *file, int line)
free() のラップ関数

- **int HECMW_check_memleak** (void)
メモリリークをチェックを実行する
 - **long HECMW_get_memsizes** (void)
現在動的に確保されているメモリサイズの合計を返す
 - **void HECMW_set_autocheck_memleak** (int flag)
呼び出された時点から、自動メモリリークチェックを有効／無効化する
 - **int HECMW_list_meminfo** (FILE *fp)
現在確保されているメモリの情報をfpに出力する
-

説明

メモリリーク検出ライブラリ

機能概要

デバッグ時におけるメモリ領域に関する情報を監視することによりメモリリークの検出を可能にする.

そのため、ANSI-C 標準ライブラリ関数のラップ関数を用意し、メモリ確保・解放には以下の関数を利用するようにする.

- **HECMW_malloc**()
- **HECMW_calloc**()
- **HECMW_realloc**()
- **HECMW_strdup**()
- **HECMW_free**()

これらのラップ関数は実際はマクロとして実装されており、デバッグシンボルの有無によって、呼び出し関数を変更する. ラップ関数が呼び出されるのは `hecmw_malloc.h` 読み込み時に、マクロ `DEBUG` が定義されている場合のみであり、それ以外の場合は、ANSI-C 標準ライブラリ関数が呼び出される. これにより、リリース時のパフォーマンス低下を防いでいる.

また、メモリリークチェック関数は、解放し忘れていた領域を検出する。このチェック関数は、デフォルトではプログラム終了時に自動的に実行されるが、これを行わないようにすることもできる。さらに、任意のタイミングでチェックすることも可能である。

マクロ定義

```
#define HECMW_malloc(size) malloc(size)
```

```
#define HECMW_calloc(nmemb, size) calloc(nmemb, size)
```

```
#define HECMW_realloc(ptr, size) realloc(ptr, size)
```

```
#define HECMW_strdup(s) strdup(s)
```

```
#define HECMW_free(ptr) free(ptr)
```

関数

```
void* HECMW_malloc_ (size_t size, char * file, int line)
```

malloc()のラップ関数

このラップ関数は、マクロ**HECMW_malloc()**より呼び出されるため、直接呼び出す必要はない。

引数:

size 確保するサイズ

file この関数の呼び出し位置(ファイル名)

line この関数の呼び出し位置(行番号)

戻り値:

成功すれば確保されたメモリ領域へのポインタを返す。失敗すればNULLを返し, `errno` にエラー番号がセットされる。

void* HECMW_calloc_ (size_t *nmemb*, size_t *size*, char * *file*, int *line*)

`calloc()`のラップ関数

このラップ関数は, マクロ**HECMW_calloc()**より呼び出されるため, 直接呼び出す必要はない。

引数:

nmemb sizeバイトの要素の個数

size 要素1個のサイズ

file この関数の呼び出し位置(ファイル名)

line この関数の呼び出し位置(行番号)

戻り値:

成功すれば確保されたメモリ領域へのポインタを返す。失敗すればNULLを返し, `errno` にエラー番号がセットされる。

void* HECMW_realloc_ (void * *ptr*, size_t *size*, char * *file*, int *line*)

`realloc()`のラップ関数

このラップ関数は, マクロ**HECMW_realloc()**より呼び出されるため, 直接呼び出す必要はない。

引数:

ptr 再配置するメモリ領域へのポインタ
size 確保するサイズ
file この関数の呼び出し位置(ファイル名)
line この関数の呼び出し位置(行番号)

戻り値:

成功すれば再配置されたメモリ領域へのポインタを返す。失敗すればNULLを返し、
`errno`にエラー番号がセットされる。

`char* HECMW_strdup_ (const char * s, char * file, int line)`

`strdup()`のラップ関数

このラップ関数は、マクロ**`HECMW_strdup()`**より呼び出されるため、直接呼び出す必要はない。

引数:

s 複製する文字列へのポインタ
file この関数の呼び出し位置(ファイル名)
line この関数の呼び出し位置(行番号)

戻り値:

成功すれば、複製した文字列へのポインタを返す。失敗すればNULLを返し、`errno`にエラー番号がセットされる。

`void HECMW_free_ (void * ptr, char * file, int line)`

`free()`のラップ関数

このラップ関数は、マクロ**`HECMW_free()`**より呼び出されるため、直接呼び出す必要はない。

引数:

ptr 解放するメモリ領域へのポインタ
file この関数の呼び出し位置(ファイル名)
line この関数の呼び出し位置(行番号)

int HECMW_check_memleak (void)

メモリリークをチェックを実行する

メモリリークチェックが行われ、メモリリークがあれば標準エラー出力に メッセージを出力する.

戻り値:

メモリリーク件数

long HECMW_get_memsize (void)

現在動的に確保されているメモリサイズの合計を返す

戻り値:

確保されているメモリサイズの合計

void HECMW_set_autocheck_memleak (int *flag*)

呼び出された時点から、自動メモリリークチェックを有効／無効化する

デフォルトで有効になっているプログラム終了時の自動メモリリークチェック を行わないようにすることができる.

引数:

flag 0で無効, 0以外で有効にする.

int HECMW_list_meminfo (FILE * *fp*)

現在確保されているメモリの情報を`fp`に出力する

引数:

fp 出力先のファイルポインタ. NULLなら標準出力に出力される.

戻り値:

現在確保されている件数

hecmw_map_int.h

整数をキーとするマップ

データ構造

- struct **hecmw_map_int_value**
キーと値のペアを格納する
- struct **hecmw_map_int_pair**
キーと内部インデックスのペアを格納する
- struct **hecmw_map_int**
整数をキーとするマップ構造体

関数

- int **HECMW_map_int_init** (struct **hecmw_map_int** *map, void (*free_fnc)(void *))
マップを初期化する
- void **HECMW_map_int_finalize** (struct **hecmw_map_int** *map)
マップのメモリ領域を解放する
- int **HECMW_map_int_nval** (struct **hecmw_map_int** *map)
データの個数を取得する
- int **HECMW_map_int_add** (struct **hecmw_map_int** *map, int key, void *value)
マップにキーと値のペアを追加する

- **int HECMW_map_int_check_dup** (struct **hecmw_map_int** *map)
マップにキーの重複がないか調べる
- **int HECMW_map_int_key2local** (struct **hecmw_map_int** *map, int key)
指定されたキーに対応する内部インデックスを取得する
- **void * HECMW_map_int_get** (struct **hecmw_map_int** *map, int key)
指定されたキーに対応するデータを取得する
- **void HECMW_map_int_iter_init** (struct **hecmw_map_int** *map)
連続してデータを取り出す際の初期化
- **int HECMW_map_int_iter_next** (struct **hecmw_map_int** *map, int *key, void **value)
連続してデータを取り出す際の次のデータの取り出し
- **int HECMW_map_int_mark_init** (struct **hecmw_map_int** *map)
データのマーキングのための初期化
- **int HECMW_map_int_mark** (struct **hecmw_map_int** *map, int key)
指定したキーに対応するデータをマークする
- **int HECMW_map_int_iter_next_unmarked** (struct **hecmw_map_int** *map, int *key, void **value)
連続してデータを取り出す際の次のマークされていないデータの取り出し
- **int HECMW_map_int_del_unmarked** (struct **hecmw_map_int** *map)
マークされていないデータすべてを削除・メモリ開放する

説明

整数をキーとするマップ

日付:

2007年11月22日

作者:

後藤 和哉

関数

int HECMW_map_int_init (struct hecmw_map_int *map, void (*free_fnc)(void *))

マップを初期化する

引数:

map マップ

free_fnc データ開放用関数へのポインタ

戻り値:

常に1を返す.

void HECMW_map_int_finalize (struct hecmw_map_int *map)

マップのメモリ領域を解放する

引数:

map マップ

int HECMW_map_int_nval (struct hecmw_map_int *map)

データの個数を取得する

引数:

map マップ

戻り値:

データの個数を返す.

int HECMW_map_int_add (struct hecmw_map_int *map, int key, void *value)

マップにキーと値のペアを追加する

引数:

map マップ

key キー

value 値

戻り値:

成功すれば1を返す. 失敗すれば0を返す.

int HECMW_map_int_check_dup (struct hecmw_map_int *map)

マップにキーの重複がないか調べる

引数:

map マップ

戻り値:

重複したデータの個数を返す.

int HECMW_map_int_key2local (struct hecmw_map_int *map, int key)

指定されたキーに対応する内部インデックスを取得する

引数:

map マップ

key キー

戻り値:

キーに対応するデータが見つければ、その内部インデックスを返す。見つからなければ-1を返す。

void * HECMW_map_int_get (struct hecmw_map_int *map, int key)

指定されたキーに対応するデータを取得する

引数:

map マップ

key キー

戻り値:

キーに対応するデータが見つければ、その値を返す。見つからなければNULLを返す。

void HECMW_map_int_iter_init (struct hecmw_map_int *map)

連続してデータを取り出す際の初期化

引数:

map マップ

int HECMW_map_int_iter_next (struct hecmw_map_int *map, int *key, void **value)

連続してデータを取り出す際の次のデータの取り出し

引数:

map マップ

key キー

value 値

戻り値:

次のデータがある場合、*key*と*value*にそれぞれキーと値をセットし、1を返す。 次のデータがない場合、0を返す。

int HECMW_map_int_mark_init (struct hecmw_map_int *map)

データのマーキングのための初期化

引数:

map マップ

戻り値:

成功すれば1を返す。 失敗すれば0を返す。

int HECMW_map_int_mark (struct hecmw_map_int *map, int key)

指定したキーに対応するデータをマークする

引数:

map マップ

key キー

戻り値:

成功すれば1を返す。 キーに対応するデータが見つからず、失敗すれば0を返す。

int HECMW_map_int_iter_next_unmarked (struct hecmw_map_int *map, int *key, void **value)

連続してデータを取り出す際の次のマークされていないデータの取り出し

引数:

map マップ

key キー

value 値

戻り値:

次のデータがある場合、*key*と*value*にそれぞれキーと値をセットし、1を返す。 次のデータがない場合、0を返す。

int HECMW_map_int_del_unmarked (struct hecmw_map_int *map)

マークされていないデータすべてを削除・メモリ開放する

引数:

map マップ

戻り値:

削除されたデータの個数を返す。

hecmw_msg.h

HEC-MWメッセージ情報ライブラリ.

```
#include "hecmw_msgno.h"
```

データ構造

- **struct hecmw_msgent**
HEC-MWメッセージ情報構造体

関数

- **char * HECMW_strmsg (int msgno)**
与えられたメッセージ番号に対応するメッセージ文字列を返す
- **int HECMW_is_syserr (int msgno)**
与えられたメッセージ番号がシステムエラー番号か否かを返す

変数

- **hecmw_msgent hecmw_msg_table []**
HEC-MWメッセージ情報を格納したテーブル.

説明

HEC-MW メッセージ情報ライブラリ.

HEC-MW において取り扱うメッセージは、メッセージ番号と、 それに対応す

るメッセージ文字列によって定義する事ができる。
このライブラリでは、容易にメッセージ情報を取得できる関数を提供する。

日付:

2004年2月16日

作者:

坂根 一彰

関数

char* HECMW_strmsg (int *msgno*)

与えられたメッセージ番号に対応するメッセージ文字列を返す

引数:

msgno メッセージ番号

戻り値:

*msgno*に対応するメッセージ文字列

int HECMW_is_syserr (int *msgno*)

与えられたメッセージ番号がシステムエラー番号か否かを返す

引数:

msgno メッセージ番号

戻り値:

*msgno*がシステムエラー番号なら1, そうでなければ0

変数

struct hecmw_msgent hecmw_msg_table[]

HEC-MWメッセージ情報を格納したテーブル.

hecmw_path.h

パス名に関する処理を行うユーティリティ

関数

- **int HECMW_get_path_separator** (void)
システムのパス区切り文字を返す
- **int HECMW_is_absolute_path** (const char *path)
パス名が絶対パスか否かを返す
- **char * HECMW_dirname** (const char *path)
パス名からディレクトリ名を取り出す
- **char * HECMW_basename** (const char *path)
パス名からディレクトリ名を削除する

説明

パス名に関する処理を行うユーティリティ

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_get_path_separator (void)

システムのパス区切り文字を返す

戻り値:

パス区切り文字

int HECMW_is_absolute_path (const char * *path*)

パス名が絶対パスか否かを返す

引数:

path パス名

戻り値:

絶対パスなら1を返し、そうでなければ0を返す

char* HECMW_dirname (const char * *path*)

パス名からディレクトリ名を取り出す

引数:

path パス名

戻り値:

ディレクトリ名

char* HECMW_basename (const char * *path*)

パス名からディレクトリ名を削除する

引数:

path パス名

戻り値:

ディレクトリ名を削除した部分

hecmw_reorder.h

メッシュ情報の並べ替え実装

```
#include "hecmw_struct.h"
```

関数

- `int HECMW_reorder_node_mpc (struct hecmwST_local_mesh *local_mesh)`
 - `int HECMW_reorder_node_dof (struct hecmwST_local_mesh *local_mesh)`
 - `int HECMW_reorder_elem_type (struct hecmwST_local_mesh *local_mesh)`
 - `int HECMW_reorder (struct hecmwST_local_mesh *local_mesh)`
-

説明

メッシュ情報の並べ替え実装

日付:

2004年2月16日

作者:

坂根 一彰

関数

`int HECMW_reorder_node_mpc (struct hecmwST_local_mesh * local_mesh)`

`int HECMW_reorder_node_dof (struct hecmwST_local_mesh * local_mesh)`

`int HECMW_reorder_elem_type (struct hecmwST_local_mesh * local_mesh)`


```
int HECMW_reorder (struct hecmwST_local_mesh * local_mesh)
```

hecmw_restart.h

リスタートデータの入出力を行う

```
#include <stdio.h>
```

関数

- **int HECMW_restart_open_by_name** (char *name_ID)
リスタートファイルを読み込み用にオープンする
- **int HECMW_restart_open** (void)
リスタートファイルを読み込み用にオープンする
- **int HECMW_restart_close** (void)
リスタートファイルをクローズする
- **void * HECMW_restart_read** (void *addr)
リスタートファイルからリスタートデータを取得する
- **int HECMW_restart_add** (void *data, size_t size)
リスタートファイルに出力するデータを登録する(データ登録処理)
- **int HECMW_restart_add_int** (int *data, int n_data)
リスタートファイルに出力するデータを登録する(データ登録処理)
- **int HECMW_restart_add_double** (double *data, int n_data)
リスタートファイルに出力するデータを登録する(データ登録処理)
- **int HECMW_restart_write_by_name** (char *name_ID)
リスタートデータを出力する

- `int HECMW_restart_write (void)`

リスタートデータを出力する

説明

リスタートデータの入出力を行う

リスタートデータは、HEC-MW が提供する API を通じて入出力することが可能である。

リスタートデータ入出力ルーチンでは、リスタートデータのデータ構造を定めていない。連続したひと固まりの領域であれば、それをリスタートデータとして扱うことができる。リスタートデータ全体は、そのような連続したデータの集合である。これらの連続したデータの意味が何であるかは、利用者が把握しておく必要がある。

リスタートデータの出力方法

リスタートデータを出力するには、まず出力するデータを出力ルーチンに登録する。出力したいデータが複数ある場合は、全てのデータに対して登録を繰り返す。この時点では、まだファイルへの出力は行われず、出力データへの領域を記憶するだけである。したがって、実際に出力が終了するまで登録したデータ領域を変更してはならない。次に示す関数が、登録を行う関数である。

- `HECMW_restart_add()`
- `HECMW_restart_add_int()`
- `HECMW_restart_add_double()`

全てのデータを登録し終えたら、ファイルへの出力を行う。この時、データ登録を行った順に、そらのデータがファイルへ出力される。

- `HECMW_restart_write()`
- `HECMW_restart_write_by_name()`

これで、リスタートファイルへの出力が完了する。

リスタートデータの入力

リスタートデータを入力するには、まずリスタートファイルをオープンする。

- **HECMW_restart_open()**
- **HECMW_restart_open_by_name()**

次に、データの入力を行う。データの入力は、登録したデータ単位で行われる。したがって、リスタートファイルの全データを取得するには、登録した回数と同じだけ入力を行う必要がある。この時、登録時と同じ順序でデータが入力され、そのデータは登録された時のデータと等しくなる。

- **HECMW_restart_read()**

必要なデータを取得し終わったら、ファイルをクローズして完了する。

-HECMW_restart_close()

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_restart_open_by_name (char * *name_ID*)

リスタートファイルを読み込み用にオープンする

*name_ID*を持つ!RESTARTで定義されるリスタートファイルをリスタートデータ読み込み用にオープンする。この時、IOパラメータは無視される。オープンしたリスタートファイルは、HECMW_restart_close()によってクローズしなければならない。

引数:

name_ID リスタートファイル識別子

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_open (void)

リスタートファイルを読み込み用にオープンする

オープンされるファイルは、全体制御ファイルの!RESTARTから自動検索される。検索対象となるのは、IO=INまたはIO=INOUTと定義されている!RESTARTである。それらのうち、全体制御ファイルの先頭から最初に定義されているものが使用される。

オープンしたリスタートファイルは、HECMW_restart_close()によってクローズしなければならない。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_close (void)

リスタートファイルをクローズする

既にオープンされているリスタートファイルをクローズする。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

参照:

HECMW_restart_open()

HECMW_restart_open_by_name()

void* HECMW_restart_read (void * *addr*)

リスタートファイルからリスタートデータを取得する

リスタートファイル内の部分データが読み込まれ、*addr*に格納される。 引数*addr*にデータを格納するためには、その呼び出しで読み込まれるデータのサイズを事前に計算し、十分なメモリ領域を確保した上で、*addr*がそれをさすようにしておく必要がある。 一度の呼び出しで取得するデータサイズはリスタートデータに依存するため、何回目の呼び出しでどのデータが読み込まれるかを把握しておくのはユーザの責任である。 ただし、*addr*にNULLを渡すと入力ルーチンがデータ入力時に自動的に適切なデータサイズを取得し、データ格納用にメモリ領域を動的確保するため、データサイズの心配はしなくてもよい。

引数:

addr リスタートデータ格納先。 NULLを指定すると動的に確保される。

戻り値:

成功すれば、リスタートデータが格納された領域へのポインタを返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

参照:

HECMW_restart_open()
HECMW_restart_open_by_name()
HECMW_restart_add()
HECMW_restart_add_int()
HECMW_restart_add_double()

int HECMW_restart_add (void * *data*, size_t *size*)

リスタートファイルに出力するデータを登録する(データ登録処理)

汎用データ型の出力インタフェース。 *data*から始まる*size*バイトをリスタートデータとして登録する。 この関数の呼び出し時点ではリスタートデータとして出力することを登録しただけであるため、実際には出力が行われるわけではない。 そのため、HECMW_restart_write()

またはHECMW_restart_write_by_name()によって、 ファイルへの出力が行われるまで、 data が指す領域を変更してはならない。

引数:

data リスタートデータ

size リスタートデータのサイズ(バイト)

戻り値:

成功すれば0を返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_add_int (int * *data*, int *n_data*)

リスタートファイルに出力するデータを登録する(データ登録処理)

int型の配列に特化した出力インタフェース。 この呼び出しは、HECMW_restart_add(data, sizeof(int)*n_data)と等価である。

引数:

data リスタートデータ

n_data dataに含まれるデータの個数

戻り値:

成功すれば0を返す。 失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_add_double (double * *data*, int *n_data*)

リスタートファイルに出力するデータを登録する(データ登録処理)

double型の配列に特化した出力インタフェース。 この呼び出しは、HECMW_restart_add(data, sizeof(double)*n_data)と等価である。

引数:

data リスタートデータ

n_data *data*に含まれるデータの個数

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_write_by_name (char * *name_ID*)

リスタートデータを出力する

HECMW_restart_add(),HECMW_restart_add_int(),HECMW_restart_add_double() で登録処理されたリスタートデータをリスタートファイルに出力する。出力されるファイルは、!RESTARTのNAMEが*name_ID*のものである。出力されるファイルのファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなる。

引数:

name_ID リスタートファイル識別子

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_restart_write (void)

リスタートデータを出力する

HECMW_restart_add(),HECMW_restart_add_int(),HECMW_restart_add_double() で登録処理されたリスタートデータをリスタートファイルに出力する。出力されるファイルは、全体制御ファイルの!RESTARTから自動検索される。検索対象となるのは、IO=OUTまたはIO=INOUTと定義されている!RESTARTである。それらのうち、全体制御ファイルの先頭か

ら最初に定義されているものが使用される。出力されるファイルのファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>」を付加したものとなる。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

hecmw_result.h

結果データ 入出力

データ構造

- **struct hecmwST_result_data**
結果データ構造体

関数

- **int HECMW_result_init** (int n_node, int n_elem, int step, char *header)
結果データ出力の初期化を行う
- **int HECMW_result_finalize** (void)
結果データ出力の後始末を行う
- **int HECMW_result_add** (int node_or_elem, int n_dof, char *label, double *ptr)
出力結果データを登録する
- **int HECMW_result_write** (void)
結果データを結果ファイルに出力する
- **int HECMW_result_write_by_name** (char *name_ID)
結果データを結果ファイルに出力する
- **int HECMW_result_write_by_fname** (char *filename)
結果データを結果ファイルに出力する

- **hecmwST_result_data * HECMW_result_read_by_name** (char *name_ID, int tstep)
結果ファイルから結果データを入力する
- **hecmwST_result_data * HECMW_result_read** (int tstep)
結果ファイルから結果データを入力する
- **hecmwST_result_data * HECMW_result_read_by_fname** (char *filename)
結果ファイルから結果データを入力する
- **void HECMW_result_free** (struct hecmwST_result_data *result)
結果ファイル構造体のメモリ領域を解放する
- **int HECMW_result_get_nnode** (void)
Fortran インタフェースが節点数取得のために呼び出す。それ以外は使用しない。 .
- **int HECMW_result_get_nelem** (void)
Fortran インタフェースが要素数取得のために呼び出す。それ以外は使用しない。 .

説明

結果データ 入出力

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_result_init (int *n_node*, int *n_elem*, int *step*, char * *header*)

結果データ出力の初期化を行う

結果データ出力に必要な情報を取得する。ただし、*header*はファイルのコメントとして使用されるだけで、任意の文字列を指定してよい。

引数:

n_node 節点数

n_elem 要素数

step タイムステップ

header ファイルヘッダ

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_result_finalize (void)

結果データ出力の後始末を行う

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_result_add (int *node_or_elem*, int *n_dof*, char * *label*, double * *ptr*)

出力結果データを登録する

出力する結果データを出力ルーチンに伝える。この時点ではまだファイルに出力されていない。出力が完了するまで、データを変更してはならない。

引数:

node_or_elem 指定する値が節点なのか要素なのかを示す(1: 節点, 2: 要素)

n_dof 自由度数

label コンポーネントラベル

ptr 結果データ

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_result_write (void)

結果データを結果ファイルに出力する

出力されるデータは、HECMW_result_add()によって登録されたデータである。出力されるファイルは、全体制御ファイルの!RESULTから自動検索される。対象となるファイルは、IO=OUTの!RESULTのうち、全体制御ファイルの先頭から最初に定義されているものになる。出力されるファイルのファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなる。

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_result_write_by_name (char * *name_ID*)

結果データを結果ファイルに出力する

出力されるデータは、HECMW_result_add()によって登録されたデータである。出力されるファイルは、全体制御ファイルの!RESULTのNAMEが*name_ID*のものとなる。この場合、IOパラメータは無視される。出力されるファイルのファイル名は、全体制御ファイルから取得したファイル名の末尾に「.<ランク番号>.<tstep>」を付加したものとなる。

引数:

name_ID 結果ファイル識別子

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

int HECMW_result_write_by_fname (char * *filename*)

結果データを結果ファイルに出力する

出力されるデータは、HECMW_result_add()によって登録されたデータである。出力されるファイルは、*filename*で指定されたものとなる。

引数:

filename 結果ファイル名

戻り値:

成功すれば0を返す。失敗すれば-1を返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmwST_result_data* HECMW_result_read_by_name (char * *name_ID*, int *tstep*)

結果ファイルから結果データを入力する

結果ファイル内のデータを読み込み、結果データ構造体に格納して返す。入力されるファイルは、全体制御ファイルの!RESULTのNAMEが*name_ID*のものとなる。この場合、IOパラメータは無視される。ファイル名は、全体制御ファイルから取得したファイル名の末尾に、「<ランク番号>.<tstep>」を付加したものとなる。

引数:

name_ID 結果ファイル識別子

tstep タイムステップ

戻り値:

成功すれば結果データ構造体を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmwST_result_data* HECMW_result_read (int *tstep*)

結果ファイルから結果データを入力する

結果ファイル内のデータを読み込み、結果データ構造体に格納して返す。入力されるファイルは、全体制御ファイルの!RESULTから自動検索される。対象となるファイルは、IO=INの!RESULTのうち、全体制御ファイルの先頭から最初に定義されているものになる。ファイル名は、全体制御ファイルから取得したファイル名の末尾に、「.<ランク番号>.<tstep>」を付加したものとなる。

引数:

tstep タイムステップ

戻り値:

成功すれば結果データ構造体を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

struct hecmwST_result_data* HECMW_result_read_by_fname (char * *filename*)

結果ファイルから結果データを入力する

結果ファイル内のデータを読み込み、結果データ構造体に格納して返す。入力されるファイルは、*filename*で指定されたものとなる。

戻り値:

成功すれば結果データ構造体を返す。失敗すればNULLを返し、HECMW_set_error()によってエラー情報がセットされる。

void HECMW_result_free (struct hecmwST_result_data * *result*)

結果ファイル構造体のメモリ領域を解放する

引数:

result 結果データ構造体

int HECMW_result_get_nnode (void)

Fortranインタフェースが節点数取得のために呼び出す。それ以外は使用しない。 .

戻り値:

節点数

int HECMW_result_get_nelem (void)

Fortranインタフェースが要素数取得のために呼び出す。それ以外は使用しない。 .

戻り値:

要素数

hecmw_result_copy_c2f.h

CからFortranへ結果データのコピーを行う.

```
#include "hecmw_result.h"
```

関数

- `int HECMW_result_copy_c2f_init (struct hecmwST_result_data *result_data, int nnode, int nelem)`
コピー元の結果データ構造体をセットする
- `int HECMW_result_copy_c2f_finalize (void)`
コピールーチン終了処理

説明

C から Fortran へ結果データのコピーを行う.

初期化処理によってセットされた結果データ構造体を C から Fortran へコピーする. 初期化処理と終了処理の間に, Fortran からコピー関数を呼ぶことによってコピーを行う. スカラーと文字列以外は, 実際にはコピーではなく, C から Fortran のメモリ領域をポインタで指しているだけである.

日付:

2004年2月16日

作者:

坂根 一彰

関数

int HECMW_result_copy_c2f_init (struct hecmwST_result_data * *result_data*, int *nnode*, int *nelem*)

コピー元の結果データ構造体をセットする

引数:

result_data コピー元の結果データ構造体

nnode 節点数

nelem 要素数

戻り値:

常に0を返す.

int HECMW_result_copy_c2f_finalize (void)

コピールーチン終了処理

コピー元の結果データ構造体へのポインタをクリアする.

戻り値:

常に0を返す.

hecmw_result_copy_f2c.h

FortranからCへ結果データのコピーを行う.

```
#include "hecmw_result.h"
```

関数

- `int HECMW_result_copy_f2c_init (struct hecmwST_result_data *result_data)`
コピーを格納する結果データ構造体をセットする
- `int HECMW_result_copy_f2c_finalize (void)`
コピールーチン終了処理

説明

Fortran から C へ結果データのコピーを行う.

初期化処理によってセットされた結果データ構造体に Fortran からのコピーを格納する. 初期化処理と終了処理の間に, Fortran からコピー関数を呼ぶことによってコピーを行う.

日付:

2004年2月16日

作者:

坂根 一彰

関数

`int HECMW_result_copy_f2c_init (struct hecmwST_result_data * result_data)`

コピーを格納する結果データ構造体をセットする

`result_data`のメンバ領域は、コピー時に割り当てられる。

引数:

result_data コピー格納先の結果データ構造体

int HECMW_result_copy_f2c_finalize (void)

コピールーチン終了処理

コピー先の結果データ構造体へのポインタをクリアする。

hecmw_set_int.h

整数の集合

データ構造

- struct **hecmw_set_int**
整数の集合構造体

関数

- int **HECMW_set_int_init** (struct **hecmw_set_int** *set)
集合を初期化する
- void **HECMW_set_int_finalize** (struct **hecmw_set_int** *set)
集合のメモリ領域を解放する
- int **HECMW_set_int_nval** (struct **hecmw_set_int** *set)
値の個数を取得する
- int **HECMW_set_int_add** (struct **hecmw_set_int** *set, int value)
集合に値を追加する
- int **HECMW_set_int_check_dup** (struct **hecmw_set_int** *set)
値に重複がないか調べる
- int **HECMW_set_int_del** (struct **hecmw_set_int** *set, int value)
値を集合から削除する

- **void HECMW_set_int_iter_init** (struct hecmw_set_int *set)
連続して値を取り出す際の初期化
 - **int HECMW_set_int_iter_next** (struct hecmw_set_int *set, int *value)
連続して値を取り出す際の次の値の取り出し
-

説明

整数の集合

日付:

2007年11月22日

作者:

後藤 和哉

関数

int HECMW_set_int_init (struct hecmw_set_int *set)

集合を初期化する

引数:

set 集合

戻り値:

常に1を返す.

void HECMW_set_int_finalize (struct hecmw_set_int *set)

集合のメモリ領域を解放する

引数:

set 集合

int HECMW_set_int_nval (struct hecmw_set_int *set)

値の個数を取得する

引数:

set 集合

戻り値:

値の個数を返す.

int HECMW_set_int_add (struct hecmw_set_int *set, int value)

集合に値を追加する

引数:

set 集合

value 値

戻り値:

成功すれば1を返す. 失敗すれば0を返す.

int HECMW_set_int_check_dup (struct hecmw_set_int *set)

値に重複がないか調べる

引数:

set 集合

戻り値:

重複したデータの個数を返す.

int HECMW_set_int_del (struct hecmw_set_int *set, int value)

値を集合から削除する

引数:

set 集合

value 値

戻り値:

成功すれば1を返す. 値が存在せず, 失敗すれば0を返す.

void HECMW_set_int_iter_init (struct hecmw_set_int *set)

連続して値を取り出す際の初期化

引数:

set 集合

int HECMW_set_int_iter_next (struct hecmw_set_int *set, int *value)

連続して値を取り出す際の次の値の取り出し

引数:

set 集合

value 値

戻り値:

次のデータがあれば, **value**に値をセットし1を返す. 次のデータがなければ0を返す.

hecmw_struct.h

メッシュデータ構造体定義

```
#include "hecmw_util.h"
```

データ構造

- struct **hecmwST_section**
セクション情報構造体
- struct **hecmwST_material**
材料情報構造体
- struct **hecmwST_mpc**
拘束グループ情報構造体
- struct **hecmwST_amplitude**
AMPLITUDE情報構造体
- struct **hecmwST_node_grp**
節点グループ情報構造体
- struct **hecmwST_elem_grp**
要素グループ情報構造体
- struct **hecmwST_surf_grp**
面グループ情報構造体
- struct **hecmwST_local_mesh**
メッシュデータ構造体

マクロ定義

- #define **HECMW_SECT_TYPE_SOLID** 1
- #define **HECMW_SECT_TYPE_SHELL** 2
- #define **HECMW_SECT_TYPE_BEAM** 3
- #define **HECMW_SECT_TYPE_INTERFACE** 4
- #define **HECMW_SECT_OPT_PSTRESS** 0

セクションオプション.

- #define **HECMW_SECT_OPT_PSTRAIN** 1
- #define **HECMW_SECT_OPT_ASYMMETRY** 2
- #define **HECMW_SECT_OPT_PSTRESS_RI** 10
- #define **HECMW_SECT_OPT_PSTRAIN_RI** 11
- #define **HECMW_SECT_OPT_ASYMMETRY_RI** 12
- #define **HECMW_AMP_TYPEDEF_TABULAR** 1
- #define **HECMW_AMP_TYPETIME_STEP** 1
- #define **HECMW_AMP_TYPEVAL_RELATIVE** 1
- #define **HECMW_AMP_TYPEVAL_ABSOLUTE** 2
- #define **HECMW_BCGRPTYPE_DISPACEMENT** 1
- #define **HECMW_BCGRPTYPE_FLUX** 2
- #define **HECMW_FLAG_PARTTYPE_UNKNOWN** 0
- #define **HECMW_FLAG_PARTTYPE_NODEBASED** 1
- #define **HECMW_FLAG_PARTTYPE_ELEMBASED** 2

説明

メッシュデータ構造体定義

日付:

2004年2月16日

作者:

マクロ定義

#define HECMW_SECT_TYPE_SOLID 1

#define HECMW_SECT_TYPE_SHELL 2

#define HECMW_SECT_TYPE_BEAM 3

#define HECMW_SECT_TYPE_INTERFACE 4

#define HECMW_SECT_OPT_PSTRESS 0

セクションオプション.

サイズ : n_sect

- 0: 指定無し, 平面圧力
- 1: 平面歪
- 2: 軸対象
- 10: 0+次数低減積分

- 11: 1+次数低減積分
- 12: 2+次数低減積分

#define HECMW_SECT_OPT_PSTRAIN 1

#define HECMW_SECT_OPT_ASYMMETRY 2

#define HECMW_SECT_OPT_PSTRESS_RI 10

#define HECMW_SECT_OPT_PSTRAIN_RI 11

#define HECMW_SECT_OPT_ASYMMETRY_RI 12

#define HECMW_AMP_TYPEDEF_TABULAR 1

#define HECMW_AMP_TYPETIME_STEP 1

#define HECMW_AMP_TYPEVAL_RELATIVE 1

#define HECMW_AMP_TYPEVAL_ABSOLUTE 2

#define HECMW_BCGRPTYPE_DISPALCEMENT 1

#define HECMW_BCGRPTYPE_FLUX 2

#define HECMW_FLAG_PARTTYPE_UNKNOWN 0

#define HECMW_FLAG_PARTTYPE_NODEBASED 1

#define HECMW_FLAG_PARTTYPE_ELEMBASED 2

hecmw_system.h

局所座標系から全体座標系への変換を行う

```
#include "hecmw_geometric.h"
```

データ構造

- struct **hecmw_system_param**

座標変換パラメータ

関数

- int **HECMW_system** (struct **hecmw_system_param** *param, struct **hecmw_coord** *coord, struct **hecmw_coord** *result)

局所座標系から全体座標系への変換を行う(未実装)

説明

局所座標系から全体座標系への変換を行う

日付:

2004年2月16日

作者:

坂根 一彰

関数

```
int HECMW_system (struct hecmw_system_param * param, struct hecmw_coord  
* coord, struct hecmw_coord * result)
```

局所座標系から全体座標系への変換を行う(未実装)

引数:

param 座標変換パラメータ

coord 変換元の局所座標

result 変換後の全体座標

hecmw_ucd_print.h

UCDファイルを出力する.

```
#include "hecmw_struct.h"
```

```
#include "hecmw_result.h"
```

関数

- `int HECMW_ucd_print (const struct hecmwST_local_mesh *mesh, const struct hecmwST_result_data *result, const char *ofname)`

説明

UCD ファイルを出力する.

日付:

2004年2月16日

作者:

坂根 一彰

関数

```
int HECMW_ucd_print (const struct hecmwST_local_mesh * mesh, const struct  
hecmwST_result_data * result, const char * ofname)
```

hecmw_util.h

HEC-MWユーティリティ.

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <stdarg.h>
#include "hecmw_config.h"
#include "hecmw_init.h"
#include "hecmw_finalize.h"
#include "hecmw_malloc.h"
#include "hecmw_log.h"
#include "hecmw_msg.h"
#include "hecmw_lib_fc.h"
#include "hecmw_comm.h"
#include "hecmw_control.h"
#include "hecmw_hash.h"
#include "hecmw_hash_int.h"
#include "hecmw_error.h"
#include "mpi.h"
```

マクロ定義

- #define **HECMW_DEBUG**(args) ((void)0)
- #define **HECMW_assert**(cond) ((void)0)
- #define **HECMW_check_condition**(cond, isabort) HECMW_check_condition_((cond)?1:0, #cond, isabort, __FILE__, __LINE__);

関数

- void **HECMW_fprintf** (FILE *fp, char *fmt,...)
メッセージを出力する

- void **HECMW_printerr** (char *fmt,...)
メッセージを標準出力に出力する
- char * **HECMW_get_date** (void)
日付を取得する
- void **HECMW_assert_** (int cond, char *cond_str, char *file, int line)
アサーションを行う
- int **HECMW_check_condition_** (int cond, char *cond_str, int isabort, char *file, int line)
診断式をチェックする.
- void **HECMW_abort** (HECMW_Comm comm)
プログラムを異常終了させる
- char * **HECMW_toupper** (char *s)
小文字から大文字への変換を行う
- char * **HECMW_tolower** (char *s)
大文字から小文字への変換を行う
- void **HECMW_print_error** (void)
エラーメッセージをログに出力する
- void **HECMW_print_msg** (int loglv, int msgno, const char *fmt,...)
メッセージをログに出力する
- void **HECMW_print_vmsg** (int loglv, int msgno, const char *fmt, va_list ap)
メッセージをログに出力する

説明

HEC-MW ユーティリティ.
HEC-MW で使用されるユーティリティ関数.

日付:

2004年2月16日

作者:

坂根 一彰

マクロ定義

```
#define HECMW_DEBUG(args) ((void)0)
```

```
#define HECMW_assert(cond) ((void)0)
```

```
#define HECMW_check_condition(cond,  
isabort) HECMW_check_condition_( (cond)?1:0, #cond, isabort, __FILE__,  
__LINE__ );
```

関数

```
void HECMW_fprintf (FILE * fp, char * fmt, ...)
```

メッセージを出力する

引数:

fp メッセージ出力先ファイルポインタ

fmt 可変引数フォーマット

... メッセージの可変引数

void HECMW_printerr (char * *fmt*, ...)

メッセージを標準出力に出力する

引数:

fmt 可変引数フォーマット

... メッセージの可変引数

char* HECMW_get_date (void)

日付を取得する

呼び出された時点での日付を取得し、フォーマットして返す。フォーマットされた日付は、静的な領域に格納され、次の呼び出しまで保存される。

戻り値:

日付を示す文字列へのポインタ。日付の取得に失敗すればNULLを返す。

void HECMW_assert_ (int *cond*, char * *cond_str*, char * *file*, int *line*)

アサーションを行う

診断した結果、*cond*が偽の場合は、エラーメッセージを標準出力に出力し、プログラムを異常終了させる。

この関数は、HECMW_assertマクロより呼び出される。

引数:

cond 診断する値
cond_str 判定条件の文字列
file ファイル名
line 行番号

int HECMW_check_condition_ (int *cond*, char * *cond_str*, int *isabort*, char * *file*, int *line*)

診断式をチェックする.

*cond*が真なら0を返す. *cond*が偽の場合は, *isabort*が0なら1を返し, そうでなければ エラーメッセージを標準エラー出力に出力し, プログラムを異常終了させる.

引数:

cond 判断する値
cond_str 判定条件の文字列
isabort 異常終了させるかどうかを示すフラグ
file ファイル名
line 行番号

戻り値:

*cond*が真なら0を返す. *cond*が偽の場合は, *isabort*が偽の場合のみ1を返す.

void HECMW_abort (HECMW_Comm *comm*)

プログラムを異常終了させる

引数:

comm 異常終了の対象となるプロセスを含むMPIコミュニケーター

char* HECMW_toupper (char * s)

小文字から大文字への変換を行う

変換後の文字列を格納する領域は、引数で渡された領域が使用され、結果で上書きされる。

引数:

s 変換対象の文字列へのポインタ

戻り値:

変換後の文字列へのポインタ

char* HECMW_tolower (char * s)

大文字から小文字への変換を行う

変換後の文字列を格納する領域は、引数で渡された領域が使用され、結果で上書きされる。

引数:

s 変換対象の文字列へのポインタ

戻り値:

変換後の文字列へのポインタ

void HECMW_print_error (void)

エラーメッセージをログに出力する

HECMW_get_errmsg()によって取得したエラーメッセージをログに出力する。

void HECMW_print_msg (int *loglv*, int *msgno*, const char * *fmt*, ...)

メッセージをログに出力する

可変引数で与えられた情報より生成したメッセージをログに出力する。 *loglv* と *msgno* は必須であるが、可変引数で与えられるメッセージは詳細メッセージであり、省略可能である。出力されるフォーマットは、 *msgno* に対応するメッセージ（詳細メッセージ） となる。詳細メッセージが省略された場合は、 *msgno* に対応するメッセージ となる。

引数:

loglv 出力するログのレベル

msgno 出力するログのメッセージ番号

fnt 詳細メッセージを示す可変引数リストのフォーマット。省略時は空白文字列とすること。

... 詳細メッセージの可変引数

void HECMW_print_vmsg (int *loglv*, int *msgno*, const char * *fnt*, va_list *ap*)

メッセージをログに出力する

詳細メッセージの引数が異なるだけで、機能はHECMW_print_msg()と同じである。

引数:

loglv 出力するログのレベル

msgno 出力するログのメッセージ番号

fnt 詳細メッセージを示す可変引数リストのフォーマット。省略時は空白文字列とすること。

ap 詳細メッセージの可変引数リスト

9. HEC-MW 要素ライブラリ

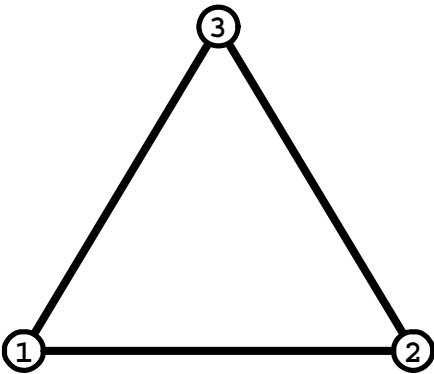
111 ロッド,リンク,トラス要素 (1 次)



112 ロッド,リンク,トラス要素 (2 次)



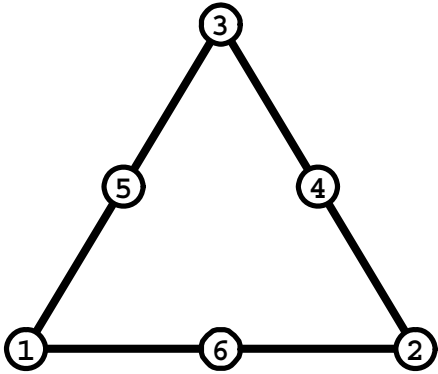
231 三角形要素 (1 次)



Surface ID

ID	connectivity
1	2 - 3
2	3 - 1
3	1 - 2

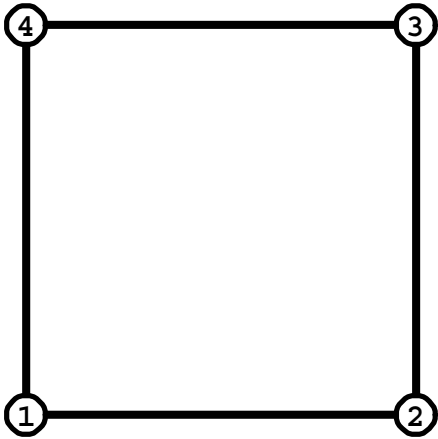
232 三角形要素 (2 次)



Surface ID

ID	connectivity
1	2 - (4) - 3
2	3 - (5) - 1
3	1 - (6) - 2

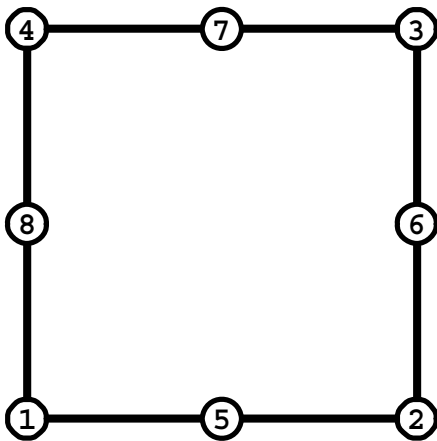
241 四角形要素 (1 次)



Surface ID

ID	connectivity
1	4 - 1
2	2 - 3
3	1 - 2
4	3 - 4

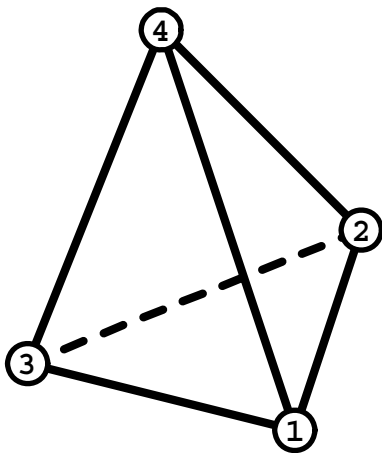
242 四角形要素（2 次）



Surface ID

ID	connectivity
1	4 - (8) - 1
2	2 - (6) - 3
3	1 - (5) - 2
4	3 - (7) - 4

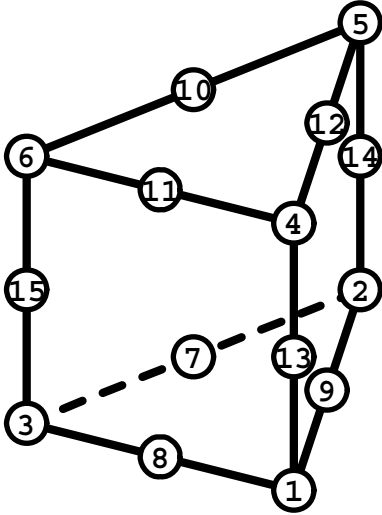
341 四面体要素（1 次）



Surface ID

ID	connectivity
1	2 - 3 - 4
2	1 - 4 - 3
3	1 - 2 - 4
4	1 - 3 - 2

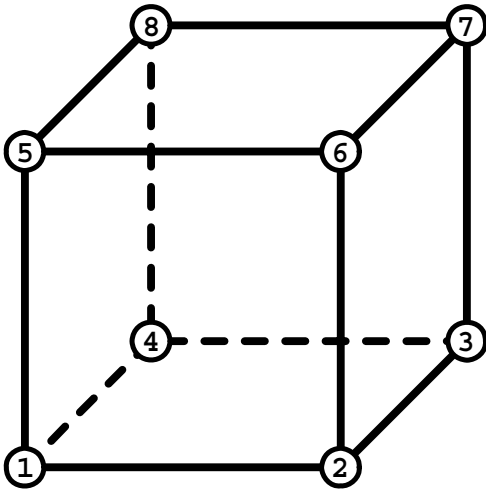
352 三角柱要素（2 次）



Surface ID

ID	connectivity
1	2 - (7) - 3 - (15) - 6 - (10) - 5 - (14)
2	3 - (8) - 1 - (13) - 4 - (11) - 6 - (15)
3	1 - (9) - 2 - (14) - 5 - (12) - 4 - (13)
4	3 - (7) - 2 - (9) - 1 - (8)
5	4 - (12) - 5 - (10) - 6 - (11)

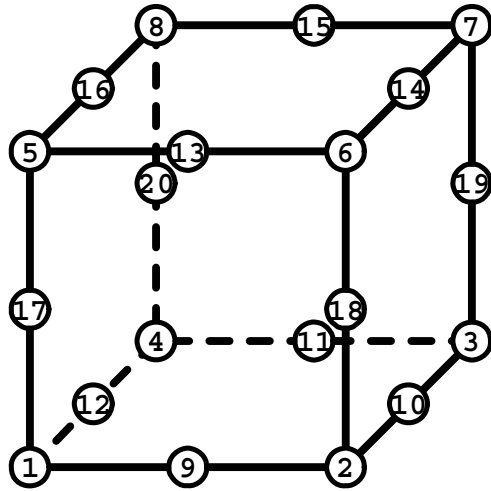
361 六面体要素（1 次）



Surface ID

ID	connectivity
1	4 - 1 - 5 - 8
2	2 - 3 - 7 - 6
3	1 - 2 - 6 - 5
4	3 - 4 - 8 - 7
5	4 - 3 - 2 - 1
6	5 - 6 - 7 - 8

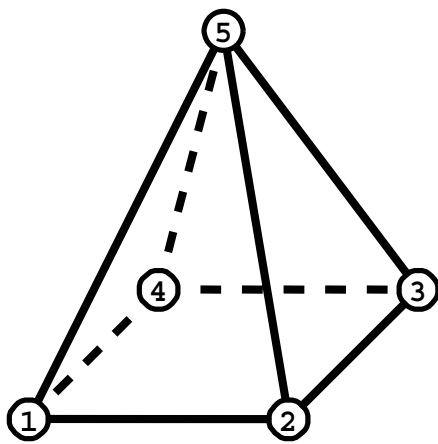
362 六面体要素 (2 次)



Surface ID

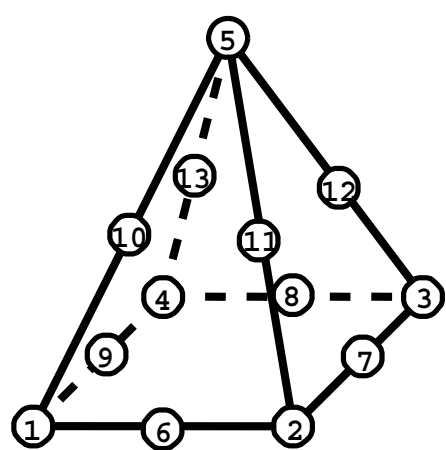
ID	connectivity
1	4 - (12) - 1 - (17) - 5 - (16) - 8 - (20)
2	2 - (10) - 3 - (19) - 7 - (14) - 6 - (18)
3	1 - (9) - 2 - (18) - 6 - (13) - 5 - (17)
4	3 - (11) - 4 - (20) - 8 - (15) - 7 - (19)
5	4 - (11) - 3 - (10) - 2 - (9) - 1 - (12)
6	5 - (13) - 6 - (14) - 7 - (15) - 8 - (16)

371 四角錘要素 (1 次)

Surface ID

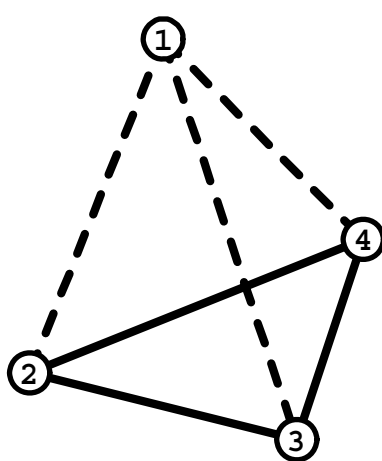
ID	connectivity
1	4 - 1 - 5
2	2 - 3 - 5
3	1 - 2 - 5
4	3 - 4 - 5
5	4 - 3 - 2 - 1

372 四角錐要素（2 次）



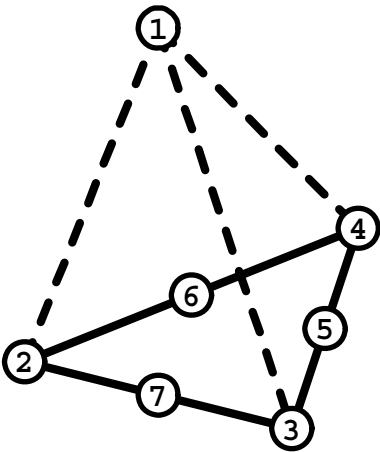
Surface ID	
ID	connectivity
1	4 - (9) - 1 - (10) - 5 - (13)
2	2 - (7) - 3 - (12) - 5 - (11)
3	1 - (9) - 2 - (14) - 5 - (12)
4	3 - (8) - 4 - (13) - 5 - (12)
5	4 - (8) - 3 - (7) - 2 - (6) - 1 - (9)

431 マスタースレーブ要素（三角形断面，1 次）



Surface ID	
ID	connectivity
1	2 - 4 - 3

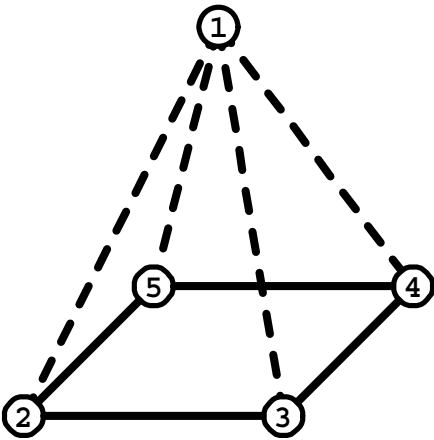
432 マスタースレーブ要素（三角形断面，2 次）



Surface ID

ID	connectivity
1	2 - (6) - 4 - (5) - 3 - (7)

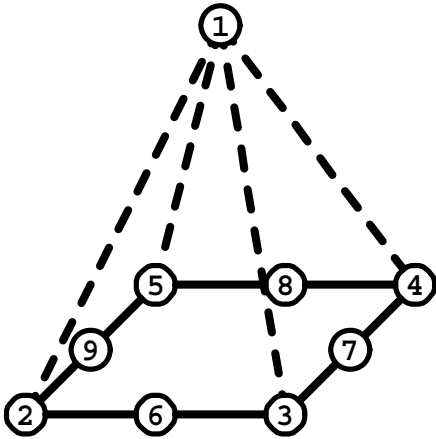
441 マスタースレーブ要素（四角形断面，1 次）



Surface ID

ID	connectivity
1	5 - 4 - 3 - 2

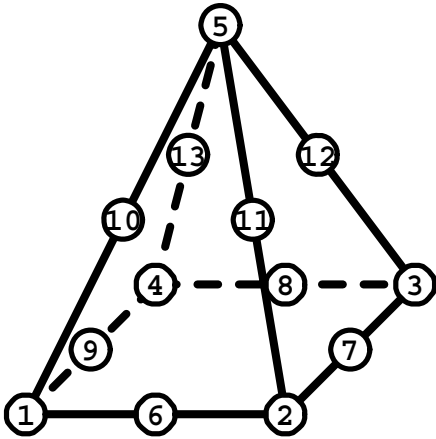
442 マスタースレーブ要素（四角形断面，２次）



Surface ID

ID	connectivity
1	5 - (8) - 4 - (7) - 3 - (6) - 2 - (9)

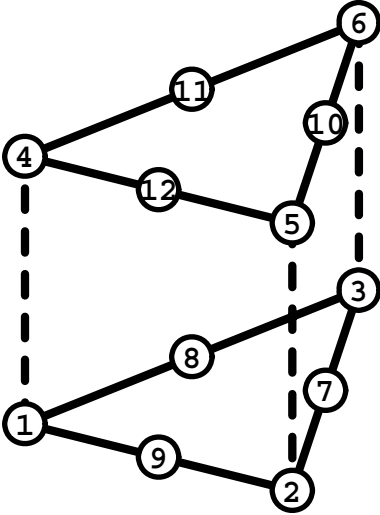
531 インタフェース要素（三角形断面，１次）



Surface ID

ID	connectivity
1	4 - (9) - 1 - (10) - 5 - (13)
2	2 - (7) - 3 - (12) - 5 - (11)
3	1 - (6) - 2 - (11) - 5 - (10)
4	3 - (8) - 4 - (13) - 5 - (12)
5	4 - (8) - 3 - (7) - 2 - (6) - 1 - (9)

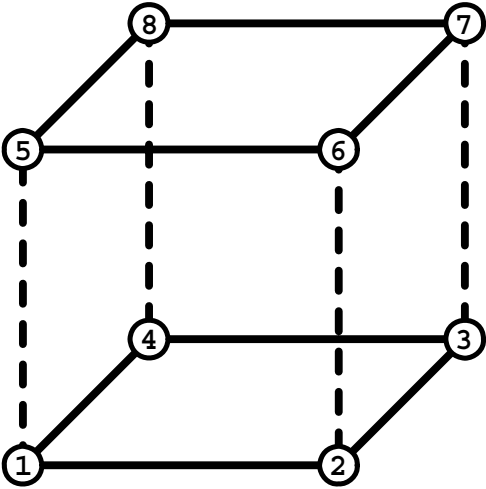
532 インタフェース要素（三角形断面，2 次）



Surface ID

ID	connectivity
1	3 - (7) - 2 - (9) - 1 - (8)
2	4 - (12) - 5 - (10) - 6 - (11)

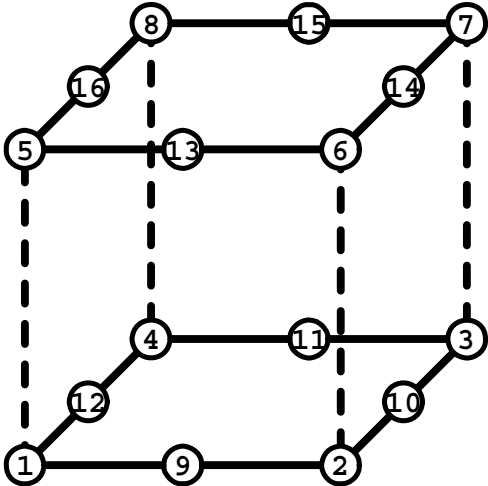
541 インタフェース要素（四角形断面，1 次）



Surface ID

ID	connectivity
1	4 - 3 - 2 - 1
2	5 - 6 - 7 - 8

542 インタフェース要素（四角形断面，2 次）



Surface ID

ID	connectivity
1	4 - (11) - 3 - (10) - 2 - (9) - 1 - (12)
2	5 - (13) - 6 - (14) - 7 - (15) - 8 - (16)

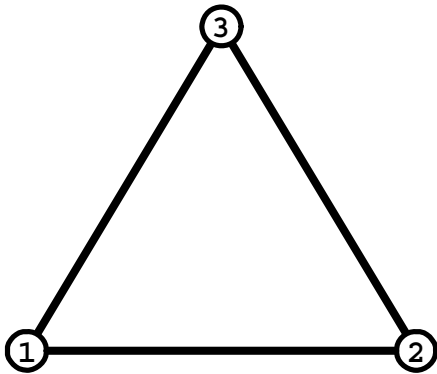
611 梁要素（1 次）



612 梁要素（2 次）



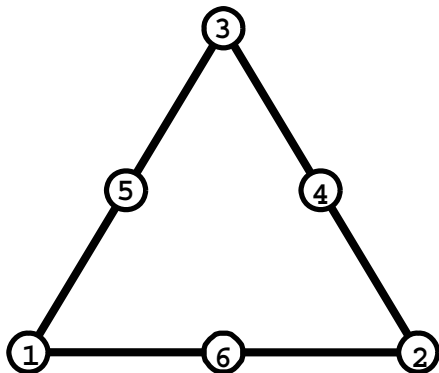
731 三角形シェル要素（1次）



Surface ID

ID	connectivity
1	1 - 2 - 3 [表]
2	3 - 2 - 1 [裏]

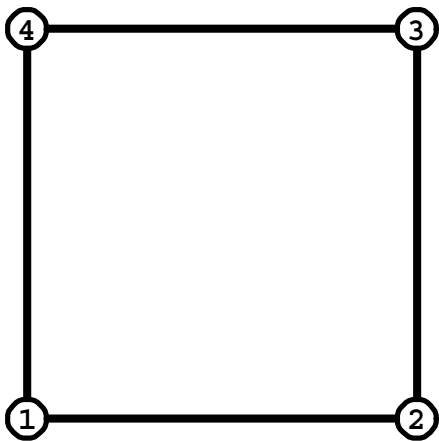
732 三角形シェル要素（2次）



Surface ID

ID	connectivity
1	1 - (6) - 2 - (4) - 3 - (5) [表]
2	3 - (4) - 2 - (6) - 1 - (5) [裏]

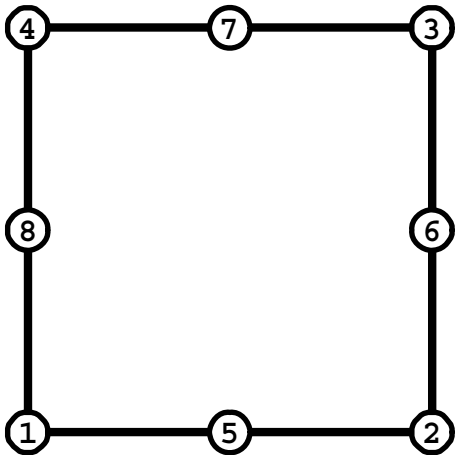
741 四角形シェル要素（1次）



Surface ID

ID	connectivity
1	1 - 2 - 3 - 4 [表]
2	4 - 3 - 2 - 1 [裏]

742 四角形シェル要素（2次）



Surface ID

ID	connectivity
1	1 - (5) - 2 - (6) - 3 - (7) - 4 - (8) [表]
2	4 - (7) - 3 - (6) - 2 - (5) - 1 - (8) [裏]

10. HEC-MW を用いたプログラム作成方法

10.1. はじめに

HEC-MW のほとんどの機能は API として提供されている。ユーザは、必要な機能が HEC-MW で提供されていれば、それをサブルーチンコールとして利用することができる。

以下では、HEC-MW を利用したプログラムの作成方法を紹介する。

10.2. コンパイル・リンク方法

HEC-MW ライブラリを使用したプログラムのコンパイル方法を示す。ただし、HEC-MW のインストールは既に完了していることを前提としている。インストール方法の詳細については、「HEC-MW インストールマニュアル」を参照すること。

インストールされた HEC-MW の各ディレクトリが以下のようになっているとする。

HEC-MW インクルードディレクトリ	/usr/local/hecmw/include
HEC-MW ライブラリディレクトリ	/usr/local/hecmw/lib

また、HEC-MW をコンパイルしたコンパイラは、以下のものであるものとする。

Fortran コンパイラ	mpif90
---------------	--------

この場合、HEC-MW を利用したプログラム "test.f90" は、

```
mpif90 test.f90 -I/usr/local/hecmw/include -L/usr/local/hecmw/lib -lfhecmw -lhecmw
```

でコンパイル・リンクできる。

MPI を用いた並列環境下においては、Fortran コンパイラが mpif90 でない場合、MPI のインクルードディレクトリとライブラリディレクトリ、ライブラリを明示する必要がある。MPI を使用する場合、開発環境における MPI の利用方法を確認されたい。

10.3. 最小の HEC-MW 利用プログラム

HEC-MW を利用するためには以下のことをプログラム内で行わなければならない。

1. hecmw モジュールを use する
2. プログラムの最初に hecmw_init をコールする
3. プログラムの最後で hecmw_finalize をコールする

HEC-MW を用いた最小のプログラムを以下に示す。

test.f90

```
program test
  use hecmw
  call hecmw_init
  call hecmw_finalize
end program test
```

プログラムの冒頭では、モジュール `hecmw` を `use` する必要がある。これにより HEC-MW の全サブルーチンがアクセス可能となる。

プログラム開始直後に、サブルーチン `hecmw_init` をコールする必要がある。この中では HEC-MW API 群に必要な初期化処理と共に、並列環境で実行する場合には MPI の初期化なども行われる。

プログラムの終了直前では、サブルーチン `hecmw_finalize` をコール必要がある。これにより、並列処理時は正常に MPI を終了することが出来る。なお、この呼び出し以降何らかの処理を行うことは推奨されない。

この例のプログラムは以下のようなコマンドで、コンパイル・リンクすることが出来る。

```
mpif90 test.f90 -I/usr/local/hecmw/include -L/usr/local/hecmw/lib -lhecmw -lhecmw
```

ただし、生成された実行モジュールのみでは、正常な行えず、エラーを返すことになる。実行には、実行時のカレントディレクトリ内に、全体制御ファイル `"hecmw_ctrl.dat"` が存在する必要がある。

本例では、全体制御ファイル中の情報を参照していないので、空のファイルでも正常実行が可能である。

10.4. メッシュデータ入力方法

ここでは、前節の最小の HEC-MW 利用プログラムに修正し、メッシュデータの入力を可能とした例を示す。

以下に修正を加えたプログラムを示す。

```
program test
  use hecmw
  implicit none
  character(len=HECMW_NAME_LEN) :: name_ID
  type(hecmwST_local_mesh) :: mesh
  call hecmw_init,
    name_ID = 'test',
    call hecmw_get_mesh(name_ID, mesh)
  call hecmw_dist_free(mesh)
  call hecmw_finalize
end program test
```

このプログラム実行時に必要となる全体制御ファイルの内容を示す。

```
!MESH, NAME=test, TYPE=HECMW-ENTIRE
mesh.dat
```

修正プログラムには、サブルーチン `hecmw_get_mesh` が追加されている。このサブルーチンにより、メッシュデータをファイルから読み込まれ、構造体変数 `mesh` に格納される。

`hecmw_get_mesh` の引数 `name_ID` は、ファイル名を示しているのではなく、全体制御ファイルで定義されている `!MESH` の `NAME` 識別子を表している。この例では、実際に読み込まれるファイルは `"mesh.dat"` となる。

Note:

`!MESH` で指定するファイル名は、単一領域メッシュデータについては全て実際のファイル名である、HEC-MW 分散メッシュデータに関しては、ファイル名から末尾の「.<rank>」を除いたものを指定する。よって、分散メッシュファイルのファイル名は、「全体制御ファイルから得られたファイル名.<rank>」でなければならない。

この機能により、プログラムが並列に動作している場合は、各プロセスが入力するファイルを区別することができる。

本例の全体制御ファイルでは、`TYPE=HECMW-ENTIRE` と記述されている。これは、

"mesh.dat" が HEC-MW 単一領域メッシュデータであることを示している.

TYPE には以下を指定することが可能である.

HECMW-DIST	HEC-MW 分散メッシュデータ
HECMW-ENTIRE	HEC-MW 単一領域メッシュデータ
GEOFEM	GeoFEM データ

これらの指定は全て全体制御ファイルで行うため、メッシュデータ入力のためのサブルーチンをタイプごとに記述する必要なく、全て、`hecmw_get_mesh` で読み込むことができる.

読み込まれたメッシュデータは、`hecmw_get_mesh` の第 2 引数 `mesh` で示される変数に格納される. データ格納のために必要なメモリ領域は、`hecmw_get_mesh` 内で自動的に確保される.

分散メッシュデータ構造体 `hecmwST_local_mesh` は、HEC-MW が内部的に使用するデータ構造である. メッシュデータからの情報は全てこの構造体に格納され、その後の処理に利用される. プログラム開発者はその情報を利用することができる.

以後、`hecmw_get_mesh` によって取得した分散メッシュデータ構造体をもとに、様々な処理を行うことになる.

また、サブルーチン `hecmw_dist_free` がコールされているが、これは分散メッシュデータ構造体に確保されているメモリ領域を開放するためのである.

10.5. メッシュデータ出力方法

この章では、メッシュデータ入力プログラムに修正を加え、メッシュデータの出力を可能にする方法について説明する.

修正を加えたプログラムを以下に示す.

```
program test
  use hecmw
  implicit none
  character(len=HECMW_NAME_LEN) :: name_ID
  type(hecmwST_local_mesh) :: mesh
  call hecmw_init
  name_ID = 'test'
  call hecmw_get_mesh(name_ID, mesh)
  name_ID = 'test-out'
  call hecmw_put_mesh(name_ID, mesh)
  call hecmw_dist_free(mesh)
  call hecmw_finalize
end program test
```

新たに追加されたサブルーチン `hecmw_put_mesh` により分散メッシュデータ構造体に格

納されているデータがファイルに出力される.

`hecmw_put_mesh` では, 出力ファイル名は全体制御ファイルで定義しておく必要がある.
全体制御ファイルの内容を以下に示す.

```
!MESH, NAME=test, TYPE=HECMW-ENTIRE
mesh.dat

!MESH, NAME=test-out, TYPE=HECMW-DIST
mesh.out
```

入力ファイル指定のための!`MESH`(`NAME` は"`test`")に加えて, 出力ファイル指定のための!`MESH` (`NAME` は"`test-out`")が定義されている.

出力ファイルは, 必ず分散メッシュデータとして出力される. 従って, 全体制御ファイルの `TYPE` 指定にも `HECMW-DIST` を指定しなければならない.

また, 出力されるファイル名は, 全体制御ファイルから取得したファイル名に「.`<rank>`」を付加したものとなる. これにより, プログラムが並列に動作している場合は, 複数の分散ファイルが出力されることとなる.

10.6. 結果データ出力方法

解析処理を行った後に, 結果データを `POST` 処理用に残す場合がある. `HEC-MW` では, 結果データをファイルに出力する方法を提供している.

結果データ出力のサンプルプログラムの一部を以下に示す.

```
program test
  use hecmw
  implicit none
  character(len=HECMW_NAME_LEN) :: name_ID
  type(hecmwST_local_mesh) :: mesh
  integer(kind=kint) :: tstep, n_dof
  character(len=HECMW_HEADER_LEN) :: header
  character(len=HECMW_NAME_LEN) :: label
  real(kind=kreal), dimension(:), pointer :: displacement
  real(kind=kreal), dimension(:), pointer :: strain
  real(kind=kreal), dimension(:), pointer :: stress

  ...

  header = 'result sample'
  call hecmw_result_init(mesh, tstep, header)
  label = 'DISPLACEMENT'
  n_dof = 3
  call hecmw_result_add(1, n_dof, label, displacement)
  label = 'STRAIN'
  n_dof = 6
  call hecmw_result_add(1, n_dof, label, strain)
  label = 'STRESS'
  n_dof = 7
  call hecmw_result_add(1, n_dof, label, stress)
```

```
call hecmw_result_write()
call hecmw_result_finalize()

...

end program test
```

全体制御ファイルの内容のうち、関係する部分を以下に示す.

```
!RESULT, NAME=result-out, IO=OUT
result.out
```

結果データの出力を容易にするために、HEC-MW では以下のような結果データ出力用サブルーチンが用意されている.

- | | |
|--------------------------|-----------|
| 1. hecmw_result_init | 結果データ出力準備 |
| 2. hecmw_result_add | 結果データ指定 |
| 3. hecmw_result_write | 結果データ |
| 4. hecmw_result_finalize | 結果データ出力終了 |

まず、hecmw_result_init によって出力ルーチン内部の初期化を行なう. これは、結果データの出力に必要な情報を事前に取得する役目がある.

取得するデータは以下のとおり.

メッシュデータ
タイムステップ
結果ファイルヘッダ

結果ファイルヘッダは、結果ファイルの識別しやすくするために指定する任意の文字列であり、結果ファイルの先頭行に出力される.

初期化が終了した後は、hecmw_result_add によって、出力したい結果データの情報を指定する. 指定する内容は以下のとおり.

指定する結果データは節点値なのか、要素値なのか (節点 : 1, 要素 : 2)
自由度数
結果データにつけるラベル
結果データそのもの (浮動小数点型一次元配列)

この指定は複数回行うことができる.

また、指定した時点では、実際にはファイルへの出力は行われず、HEC-MW 内部に情報が蓄積される。従って、ここで指定したデータの内容は、実際にファイルへの出力が終了するまで変更してはならない。

出力したい結果データの指定を全て終了したら、`hocmw_result_write` を実行し実際にファイルに出力する。

出力するファイル名は全体制御ファイルから取得する。取得するファイル名は、`!RESULT` で `IO` パラメータが `OUT` のもののうち、最初に定義されているものとなる。もし、出力ファイルを明示的に指定したいのなら、`hecmw_result_write_by_name` を使用して、引数に `name_ID` を指定すると、定義されている `!RESULT` から `NAME` が該当するものを取得することができる。この場合、`IO` パラメータは無視される。

実際に出力するファイルの名前は、取得したファイル名に「.<rank>.<tstep>」を付加したものとなる。

最後に、終了処理として `hecmw_result_finalize` をコールすること。これによって、`hecmw_result_add` された情報がクリアされる。

10.7. 結果データ入力方法

結果データをファイルから入力するのは非常に容易である。

サンプルプログラムの一部を以下に示す。

```
...
integer(kind=kint) :: tstep
type(hecmwST_result_data) :: result
...
call hecmw_result_read(tstep, result)
...
```

全体制御データの内容のうち、関係するものを以下に示す。

```
!RESULT, NAME=result-in, IO=IN
result.in
```

結果データをファイルから入力するためには、`hpmcw_result_read` をコールするのみでよい。結果データは結果データ構造体に格納され、返される。

結果データ構造体は以下のようになっている。

```

type hecmwST_result_data
  integer(kind=kint) :: nn_component
  integer(kind=kint) :: ne_component
  integer(kind=kint), pointer :: nn_dof(:)
  integer(kind=kint), pointer :: ne_dof(:)
  character(len=HECMW_NAME_LEN), pointer :: node_label(:)
  character(len=HECMW_NAME_LEN), pointer :: elem_label(:)
  real(kind=kreal), pointer :: node_val_item(:)
  real(kind=kreal), pointer :: elem_val_item(:)
end type hecmwST_result_data

```

変数名	内容
nn_component	節点コンポーネント数
ne_component	要素コンポーネント数
nn_dof	節点自由度数
ne_dof	要素自由度数
node_label	節点ラベル
elem_label	要素ラベル
node_val_item	節点値
elem_val_item	要素値

node_val_item の並びは、節点ごとに全コンポーネントが順に格納されている。

elem_val_item の並びは、節点ごとに全コンポーネントが順に格納されている。

結果ファイルのデータは全てこの構造体に格納されるので、これを参照して処理を行うことができる。

10.8. リスタートデータ出力方法

HEC-MW では、リスタートデータを出力する方法を提供している。

リスタートファイルに出力可能なデータは特に決まっておらず、ユーザが自由に選択することができる。

リスタートデータ出力の手順を以下に示す。

1. ファイルに出力したいデータを指定する（複数回可能）
2. ファイルに出力する

以下にサンプルプログラムの一部を示す。

```

...

integer(kind=kint),pointer,dimension(:) :: int_data
real(kind=kreal),pointer,dimension(:) :: real_data
real(kind=kreal),pointer,dimension(:) :: real_data2

...

allocate(int_data(100))
allocate(real_data(200))
allocate(real_data2(300))

...

call hecmw_restart_add_int(int_data, size(int_data))
call hecmw_restart_add_real(real_data, size(real_data))
call hecmw_restart_add_real(real_data2, size(real_data2))
call hecmw_restart_write()

...

```

全体制御ファイルを以下に示す.

```

!RESTART,NAME=restart-out,IO=OUT
restart.out

```

サブルーチン `hecmw_restart_add_int`, `hecmw_restart_add_real` は, それぞれ第1引数に整数型, 浮動小数点型の一次元配列をリスタートデータとして指定することができる. これらで指定された情報は, **HEC-MW** 内部に蓄えられ, ファイルに書き出されるまで保持される. 従って, 実際にファイルに出力されるまで指定したデータを書き換えてはならない.

また, 第2引数には, 指定した配列の要素数を与える.

`hecmw_restart_add_int`, `hecmw_restart_add_real` で指定されたデータは, `hecmw_restart_write` がコールされた時点でファイルに出力される.

このとき出力されるファイルの名前は, 全体制御ファイルから取得される. 実際に出力されるファイルの名前は, 「全体制御ファイルから取得したファイル名.<rank>」となる. `hecmw_rsetart_write` で取得されるファイル名は, `!RESTART` で, `IO` パラメータが `OUT` または `INOUT` のもののうち, 最初に定義されたものである. `hecmw_restart_write_by_name` を使用すると, `NAME` が指定できる. この場合, `IO` パラメータは無視される.

10.9. リスタートデータ入力方法

リスタートファイルからリスタートデータを入力するには, 以下のようにする.

1. リスタートファイルをオープンする

2. リスタートファイルから入力（出力した時と同じ回数）
3. リスタートファイルをクローズする

以下にサンプルプログラムの一部を示す.

```
...  
  
integer(kind=kint),pointer,dimension(:) :: int_data  
real(kind=kreal),pointer,dimension(:) :: real_data  
real(kind=kreal),pointer,dimension(:) :: real_data2  
  
allocate(int_data(100))  
allocate(real_data(200))  
allocate(real_data2(300))  
call hecmw_restart_open()  
call hecmw_restart_read_int(int_data)  
call hecmw_restart_read_real(real_data)  
call hecmw_restart_read_real(real_data2)  
call hecmw_restart_close()  
  
...
```

以下に全体制御ファイルを示す.

```
!RESTART,NAME=restart-in,I0=IN  
restart.in
```

リスタートファイルのオープンは, `hecmw_restart_open` で行う.

このときオープンされるファイルは, `!RESTART` で I/O パラメータが `IN` または `INOUT` のもののうち, 最初に定義されているものとなる. `hecmw_restart_open_by_name` を使用すれば, `NAME` 指定ができる. この場合, I/O パラメータは無視される.

オープンした後は, `hecmw_restart_read_int`, `hecmw_restart_read_real` によって, 実際にファイルからデータを入力し, 変数に格納する. このとき注意すべきことは, リスタートの入力と出力は整合性を取らなければならないということである. つまり, 出力と入力の順番, 型, 配列要素数は全て一致していなければならない. また, 格納先の領域は事前にユーザが確保しておく必要がある.

全てのリスタートデータを入力し終わったら, `hecmw_restart_close` によってファイルを閉じる.

10.10. 可視化方法（メモリ渡し）

可視化には, ファイル渡しとメモリ渡しの二通りの方法がある.

ファイル渡しとは, 解析の結果データを予め結果ファイルとしてファイルに出力しておき,

それを別プロセスとして可視化機能を実行し可視化を行う方法である。

メモリ渡しとは、解析終了後、メモリ上に存在している結果データを使用して同一プロセスで可視化まで行う方法である。

可視化の手順は以下のとおりである。

1. 可視化データ構造体の作成
2. 可視化初期化
3. 可視化
4. 可視化終了処理

ここでは可視化の詳細には触れませんが、以上の手順を踏めばよいことになる。

サンプルプログラムの一部を以下に示す。

```
...  
integer(kind=kint) :: step, max_step, is_force  
type(hecmwST_result_data) :: result  
...  
call hecmw_visualize_init  
call hecmw_visualize(mesh, result, step, max_step, is_force)  
call hecmw_visualize_finalize  
...
```

まず、可視化を行うために結果データが必要である。この結果データは、ユーザが結果データ構造体に格納しておく必要がある。

`hecmw_visualize_init` によって可視化の初期化を行う。

次に、`hecmw_visualize` をコールすると、実際に可視化が行われる。

可視化実行時に与える情報は以下のとおりである。

メッシュデータ

結果データ

タイムステップ

最大タイムステップ

最後のステップで強制的に描くかどうかのフラグ

最後に可視化の終了処理として、`hecmw_visualize_finalize` をコールすること。

以上で、可視化が終了する。