1.3.5 UCD ファイル・バイナリ・フォーマット

UCD バイナリ・フォーマットは２種類のファイルで構成されます。

- コントロールファイル
  このファイルには、各ステップのデータファイル名を記述します。また、データの記述方式（繰り返しのタイプ）も指定します。このファイルは１つの非構造型データにつき１つ用意する必要があり、すべてアスキー形式で記述します。ファイル読み込みダイアログからは、このファイルを指定することにより、データ全体を読み込むことができます。

- データファイル
  １ステップ当たりのデータ本体をバイナリ形式で記述します。このファイルは、１つの非構造型データにつき、そのステップ数だけ存在します。

コントロールファイル

このファイルは、バイナリ・フォーマットの全体の情報をアスキー形式で記述するものです。拡張子は .inp にして下さい。

フォーマットは以下の通りです。

```
#                    ----------(1)
#
＜サイクル・タイプ＞           ----------(2)
＜各ステップ毎のファイルパス＞    --------(3)
...
```

(1)#

で始まる行はコメント行として無視します。ただし、(2) の部分以降に入れることはできません。

(2) ＜サイクル・タイプ＞

繰り返しのタイプ。**data, geom, data_geom** から指定します。
３つのタイプの違いについては、「1.3.4 UCD ファイル・アスキー・フォーマット」－ 1-20 ページの説明をご参照ください。
１時刻（時系列ではない）のデータの場合、data_geom を指定して下さい。

(2) ＜各ステップ毎のファイルパス＞

１ステップのファイル名、２ステップのファイル名 ... と時系列データの数分だけ、ファイル名を定義します。
絶対パス、相対パスのいずれの指定も可能です。

以下に３ステップのデータを持つコントロールファイルの例を示します。

```
# UCD Binary format sample
#
data
c:/AVS/sample/ucdbin01.dat
c:/AVS/sample/ucdbin02.dat
c:/AVS/sample/ucdbin03.dat
```

データファイル

このファイルは、１ステップ分の非構造格子型データを記述するファイルで、バイナリフォーマットで記述します。
ファイルの拡張子は任意です。

C 言語、ならびに Fortran(unformatted) で作成されたデータをサポートしています。

データファイルの構造を以下に示します。大きく６つに分類できます。

- ◆ ファイル情報
- ◆ ステップ情報
- ◆ 節点情報　(形状データ)
- ◆ 要素情報　(形状データ)
- ◆ 節点データ
- ◆ 要素データ（

ファイル情報とステップ情報は必ず必要です。
形状データと設定／要素データは、コントロールファイルに記述されたサイクルタイプによって、１回だけ出力、もしくは、全ステップに対して出力といった構成が決まります。

◆ファイル情報

以下の２つの情報です。

レコード名	データ型	バイト数	繰り返し
キーワード	char[7]	7	1
バージョン	float[1]	4	1

キーワードは、以下のいずれかの文字列となります。

- 'AVS UCD'
  32bit 版用に作成したデータの場合には、この文字列、AVS（半角空白）UCD の７文字を指定します。
- 'AVSUC64'
  64bit 版用に作成した大規模データの場合には、こちらを指定して下さい。
  格子数など 32bit を超える値のデータを扱うことができます。
  以下、各フォーマット内の各変数の int と long の扱いの違いについてもご注意下さい。

バージョンは 1.0 に固定です。float で 1.0 を出力します。

注意： *以降の説明も含め、表中の繰り返しはそのレコードが繰り返される回数を示します。繰り返しのない場合、1と示します。*

注意： *FORTRANの場合には、以降の説明も含め、1レコード分の出力はWrite文1回の記述で行う必要があります。*

◆ステップ情報

以下の２つの情報です。

レコード名	データ型	バイト数	繰り返し
タイトル	char[70]	70	1
ステップ番号	int[1]	4	1
ステップ時刻	float[1]	4	1

タイトルには 70 文字以内の文字列を指定できます。この情報は画面表示のみに利用されるので記述内容は任意です（70 文字未満の場合は、残りのバイトをスペース、NULL 文字等で埋めてください)。

ステップ番号は、必ず１から始まる番号で出力してください。
アスキーのコントロールファイルに５つのファイルがある場合、１～５の値をそれぞれのバイナリファイルに出力してください。

ステップ時刻は、浮動少数点で与える任意の値です（バージョン 7.0 では利用されていません)。

◆節点情報

節点数と座標情報です。

レコード名	データ型	バイト数	繰り返し
節点数	int/long[1]	4/8	1
座標値記述タイプ	int	4	1
XYZ 座標情報	-	-	-

節点数は32bit版と64bit版の違いによって、4byteか8byteの違いがあります。64bit版では大規模データの扱いができるように、long型でその節点数を出力してください。

XYZ 座標情報には、２種類のタイプがあります。節点数の次のレコードに出力されている座標値記述タイプの値、１か２によって、その出力方法が異なります。

●座標値記述タイプ＝１の場合：

レコード名	データ型	バイト数	繰り返し
節点番号、X 座標値、Y 座標値、Z 座標値	int/long[1], float[1], float[1], float[1]	4/8, 4, 4, 4	節点数

座標値記述タイプが１の場合、１つの節点の出力を１レコードとして扱います。

注意： *節点数と同様、64bit 版では、節点番号は long 型で出力します。*

●座標値記述タイプ＝２の場合：

レコード名	データ型	バイト数	繰り返し
節点番号	int/long[節点数]	4/8* 節点数	1
X 座標値	float[節点数]	4* 節点数	1
Y 座標値	float[節点数]	4* 節点数	1
Z 座標値	float[節点数]	4* 節点数	1

座標値記述タイプが２の場合、節点数分だけの情報をひとかたまりとして、一度に出力します。

注意： *節点数と同様、64bit 版では、節点番号は long 型で出力します。*

◆要素情報

要素タイプや要素の構成情報です。

レコード名	データ型	バイト数	繰り返し
要素数	int/long[1]	4/8	1
全要素の要素番号配列	int/long [要素数]	4/8* 要素数	1
全要素のマテリアル番号配列	int[要素数]	4* 要素数	1
全要素の要素タイプ配列	char[要素数]	1* 要素数	1
全要素の要素の構成配列	int/long [構成数*要素数]	4/8* 構成数 * 要素数	1

要素数は三角形や四角形、三角柱など、UCD データを構成する要素の数を指定します。

注意： *節点数と同様、64bit 版では、要素数は long 型で出力します。*
以降、同様に int/long の記載については、32bit 版、64bit 版における出力の違いを示しています。

要素番号は、要素を識別するための一意の番号です。通常、１番から順に連番で与えます（連番でなくても結構です）。異なる要素で同じ番号を使用しないように注意して下さい。

マテリアル番号は、要素をグループ分けするのに用いる整数値です。

要素タイプは、三角形や四角形など要素の種類を ID で示したものです。要素と ID の関係は、先に述べた「 要素タイプと結線リスト」 － 1-24 ページにある () 内の数値 0 ～ 14 をご参照ください。
図に記された ID（整数値）を 1byte で出力します。

要素の構成配列は、「 要素タイプと結線リスト」 － 1-24 ページの図を参考に要素を構成する結線リスト情報を出力します。
例えば三角形要素であれば、３つの節点番号を結ぶ配列となります。その要素の構成を要素数分だけ列挙します。

◆節点データ

節点上に定義されたデータ値の情報です。

レコード名	データ型	バイト数	繰り返し
節点データ成分数	int[1]	4	1
節点データ記述タイプ	int[1]	4	1
節点データ情報	-	-	-

節点データ成分数は、その節点上のデータが何個あるかを示します。以降で説明するベクトル長データをあわせてご参照ください。
例えば、ある節点に U,V,W の３ベクトルデータがある場合、データ成分数＝１、ベクトル長＝３として表現できます。また異なるケースでは、節点上に温度、圧力の２つのデータがある場合、データ成分数＝２、ベクトル長をそれぞれ１として扱うことができます。

> 注意： *節点データが存在しない場合（要素データのみが存在するような場合）は節点データ成分数＝０として、記述タイプから後はスキップしてください。*

節点データ情報には、４種類のタイプがあります。節点データ記述タイプの値、１か２か３か４によって、その出力方法が異なります。

●節点データ記述タイプ＝１の場合：

レコード名	データ型	バイト数	繰り返し
成分名、単位、ベクトル長、NULL データ設定フラグ (0/1)、NULL データ値	char[16], char[16], int[1], int[1], float[1]	16,16,4,4,4	成分数
節点毎の全成分データ値	float[成分数]	4* 成分数	節点数

節点データ記述タイプが１の場合、まず、全成分の成分名やベクトル長などのデータ情報を記述します。その後、各節点毎にデータ値を列挙します。

以下に、アスキーデータを使ってデータの並びを表現した例を示します。

```
2
1
Velocity     m/s       3       1       -999
Temperature  degree    1       0       0
U1           V1        W1      T1
U2           V2        W2      T2
U3           V3        W3      T3
```

まず、データ成分数 = 2 を出力します。次に記述タイプ = 1 を示しています。２つのデータ成分のうち、１つ目の速度成分（Velocity）はベクトル長 = 3 で構成され、２つ目の温度（Temperature）はベクトル長 = 1 のデータです。
NULL フラグはオンに設定され -999 の値は NULL 値として扱われます。
実際には U1, V1, W1, ... にデータ値が出力されますが、データの並びは、この記述タイプでは、節点後に出力されます。

●節点データ記述タイプ＝２の場合：

レコード名	データ型	バイト数	繰り返し
成分名、単位、ベクトル長、NULL データ設定フラグ (0/1)、NULL データ値	char[16], char[16], int[1], int[1], float[1]	16,16,4,4,4	成分数
成分毎の全節点データ値	float[節点数]	4* 節点数	成分数

節点データ記述タイプが２は、先の記述タイプが１の場合とほとんど同じです。データ値の出力において、成分毎に出力されます。

先の例と同様、アスキーデータを使ってデータの並びを表現した例を示します。

```
2
2
Velocity     m/s       3       1       -999
Temperature  degree    1       0       0
U1           U2        U3      ...
V1           V2        V3      ...
W1           W2        W3      ...
T1           T2        T3      ...
```

U1, U2, ... の実際のデータ値の並びを比較してみてください。
この記述タイプでは成分毎に出力されます。

●節点データ記述タイプ＝３の場合：

レコード名	データ型	バイト数	繰り返し	
成分名、単位、ベクトル長	char[16], char[16], int[1]	16,16,4	成分数	
対象成分の有効節点数	int/long[1]	4/8	1	成分数
対象成分の有効節点の節点番号、節点データ値	int/long[1], float[ベクトル長]	4/8,4* ベクトル長	有効節点数	

節点データ記述タイプが３のケースでは、NULL 値を指定する代わりに、必要なデータ値だけを出力します。
成分名等の情報出力の後、まず、１成分目の有効な節点数を出力します。
次に、その有効節点の番号とデータ値をベクトル長の数だけ列挙し、成分数の分だけ繰り返します。

以下のアスキーデータで表現した例をあわせてご参照ください。

```
2
3
Velocity     m/s        3
Temperature  degree     1
3
1            U1         V1         W1
3            U3         V3         W3
4            U4         V4         W4
2
1            T1
2            T2
```

速度成分の有効な節点は 1,3,4 で、節点番号 =2 には NULL 値が設定されます。温度については、1,2 番のみが有効で、3,4 番の節点の温度は NULL が設定されます。

●節点データ記述タイプ＝４の場合：

レコード名	データ型	バイト数	繰り返し	
成分名、単位、ベクトル長	char[16], char[16], int[1]	16,16,4	成分数	
対象成分の有効節点数	int/long[1]	4/8	1	成分数
対象成分の有効節点の節点番号	int/long[有効節点数]	4/8* 有効節点数	1	
対象成分の有効節点の節点データ値	float[有効節点数]	4* 有効節点数	ベクトル長	

このタイプは３のタイプと同様、NULL 値の定義を有効データのみ出力することで指定します。

タイプ３の例とあわせて、以下のアスキーデータで表現した例をご参照ください。

```
2
4
Velocity     m/s        3
Temperature  degree     1
3
1            3          5
U1           U3         U5
V1           V3         V5
W1           W3         W5
2
1            2
T1           T2
```

速度成分の有効な節点は 1,3,4 で、節点番号 =2 には NULL 値が設定されます。温度については、1,2 番のみが有効で、3,4 番の節点の温度は NULL が設定されます。

◆要素データ

要素データの記述は、節点データ記述と同じです。節点を要素と適宜読み替えて参照ください。例えば、節点番号は要素番号に読み替えます。

また、先に述べたように、節点データを持たず要素データのみを持つデータでは、節点データ数を０として出力し、続けて、要素データ数を出力、いずれかの記述タイプでデータを定義します。