

データ型の強制

強制とは、あるタイプのデータをプロシージャに渡し、別のタイプを受け取ることです。渡されたデータはバックグラウンドで新しいタイプに変換されます。プロシージャのユーザーは、よく似た多くのデータタイプを使用できる利点があります。プロシージャの作者は、タスクに合った最適なデータ型のみを扱うことに集中できるという利点があります。

$$p([[1,2], [3,4]]) \rightarrow p\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$$

Maple パッケージコマンドの使用: LinearAlgebra

パラメータに対して行列を要求する多くの [LinearAlgebra](#) ルーチンがあります。データタイプの強制の紹介の前に、これらのルーチンは [Arrays](#) または [lists](#) の場合はエラーを発生します。データタイプの強制では、これらのルーチンはより多くの入力を受け取ることができます。

```
> LinearAlgebra:-Rank( Array( [ [ 1, 2 ], [3, 4] ] ) );  
2 (1.1)
```

```
> LinearAlgebra:-Determinant( [ [ 6, 7 ], [8, 9] ] );  
-2 (1.2)
```

Maple のライブラリのその他の多くのルーチンは、強制の使用のために更新されています。新機能の利用のためにプログラミングの知識は必要ありません。同じコマンドを使用するだけでより多くのデータタイプで作業することができます。

強制には、整数から浮動小数、名前から文字列、行列から名前の変換が含まれます。ターゲットプロシージャの著者はコアデータ構造を扱い、パラメータ処理を使用して他のタイプをサポートします。データタイプの強制を使用すると、コードの作成がより簡単になり、使用するタイプのみ
の宣言ですむため Maple は強制を管理します。

強制を使用するパラメータ宣言

```
> p := proc( m::Matrix )  
    whattype(m);  
end;  
p( Array( [ [ 1, 2 ], [3, 4] ] ) );  
Matrix (2.1)
```

```
p( Vector[row]( [ 5, 6, 7 ] ) );  
Matrix (2.2)
```

```
p( [ [ 1, 2, 3 ], [ 4, 5, 6 ] ] );  
Matrix (2.3)
```

上記の例で チルダ (~) を `M::Matrix` パラメータに追加すると、Maple は Matrix と同じもの

を受け取ります。ベクトル形式で渡すことができます。その裏側では n -要素の列ベクトルは $n \times 1$ 行列に変換されます。この場合、実行中の変換やデータのコピーはなく、エイリアスまたは同一データへの異なるビューが適用されます。「強制」はデータ構造が適切な形式に変換されることです。

`rtable` から `rtable` への強制で重要な点は、配列の下限オプションをコントロールすることです。Maple は配列が宣言されたときに下限を選択できますが、インデックスが 1 以外から始まる配列の処理をコードが行わない場合があります。強制を使用すると、1 ベースの配列を指定することができます。

▼ 配列の範囲の統一

この例では、渡される 2×2 配列の下限が 2 および 6 です。プロシージャ 'p' の呼び出し中にエイリアスが作成され、Bには新規のデータビューがあります。プロシージャの内部で B は通常の 1 ベースの配列として使用されます。

```
[ > p := proc( B::~Array(1..,1..) )
      ArrayDims(B);
    end:
  A := Array(2..3, 6..7) :
  ArrayDims(A);
                                     2..3, 6..7                                (3.1)
```

```
  p(A);
                                     1..2, 1..2                                (3.2)
```

1 ベースでない配列を使用する場合も考えられます。たとえば、C のような別の言語で作成されたコードを変換する場合は、0 ベースの配列を使用するとより簡単です。つぎの例では、M [0,0] はプロシージャ内部の最初の配列で、プロシージャの外では呼び出し側が Maple の配列および行列のデフォルトを使用しています。強制はデータをコピーせずに最も効率的な方法で行われます。

```
[ > p := proc( M::~Array(0..,0..) )
      M[0,0];
    end:
  p( <1, 2; 3, 4 > );
                                     1                                    (3.3)
```

上記の例の強制では、タイプ T についてコマンド `~T` が存在するときに適用されます。現在、`~Array`、`~Matrix` および `~Vector` は Maple の最上位の組込みプロシージャです。コマンドを最上位レベルのコマンドまたはモジュールのエクスポートとして追加できます。

▼ ユーザ定義強制の追加

以下の例では、モジュール内でのプライベート強制ルーチンの宣言方法を説明します。
 "ModuleApply" を使用すると、モジュールはプロシージャと同様に動作しますが、モジュールの場合は他のメソッドとデータが関連しています。赤色で強調表示されたメソッドは、浮動小数を有理数に変換する方法を宣言しています。青色で強調表示されたメソッドはパラメータ r を使用し、強制タイプ ~rational として宣言しています、このプロシージャが浮動小数引数で呼び出されると、プロシージャのボディは有理数に強制された部分のみを表示します。

```

> Frac := module()
  local ModuleApply, ~rational;
  ~rational := proc( a::float ) convert(a,rational); end;
  ModuleApply := proc( r::~-rational ) frac(r); end;
end module:
Frac(1.5);

```

$$\frac{1}{2} \quad (4.1)$$

```

Frac( $\frac{3}{2}$ );

```

$$\frac{1}{2} \quad (4.2)$$

長い形式 `coerce()` のパラメータ宣言では、強制候補のタイプ、および変換方法について正確にコントロールすることができます。 `coerce()` 宣言の内部は、タイプおよび/またはプロシージャのシーケンスです。プロシージャは "arrow" プロシージャとインラインで宣言され、通常各プロシージャの明示的なタイプを含んでいます。

▼ `coerce()` を使用したフレキシブルかつ拡張された構文

例:

名前または文字列をプロシージャの最初の引数として使用するための簡単な方法ですが、文字列形式を扱う必要があります。

```

> p := proc( s::coerce( string, (s::name)->convert(s,string) ) )
  s;
end:

```

このプロシージャは単一引数を受け取り変更せずに返します。このプロシージャが s を `string` として呼び出される場合は、最初の強制 (緑色) と一致します。これは文字列と一致するタイプです。リターン値は文字列です。引数は変更されずに渡されます。

```

p("a string");

```

$$\text{"a string"} \quad (5.1)$$

同じプロシージャで s が `name` である場合は、2 番目の強制 `s->convert(s,string)` が適用され

ます。これはインラインコードとして実行され、渡された引数を文字列に変換します。リターン値は文字列です。パラメータはプロシージャのボディのすべてのコードの実行前に変更されています。

```
p(`a name`);
```

"a name"

(5.2)

渡される引数の種類 (整数など) によっては、強制のプロシージャが一致しないことがあります。's' は予想されるパラメータなので、予想したタイプと一致せずエラーが発生します。

```
p(expect + an + Error);
```

```
Error, invalid input: p expects its 1st argument, s, to be of type  
string or coercible via proc (s::name) options operator, arrow:  
convert(s, string) end proc, but received expect+an+Error
```

詳細は、以下のヘルプトピックを参照してください。

変換

- [プロシージャのパラメータ修飾子](#)
- [プロシージャ](#)
- [~ 配列、~ 行列、~ ベクトル](#)