TLArray パッケージ (v0.3)
簡易マニュアル

ワトソン

2015 年 11 月 25 日

1　概要

1.1　基本的な定義

TLArray パッケージにおける配列とは，複数のトークン列を扱うことのできるデータ構造をさす．各配列は半角の英数字により構成される配列名をもち，また，各配列の保持する個々のデータ（トークン列）を要素と呼ぶ．要素には順序が存在しており，1 つの要素には 1 つのインデックスが対応している．ただし，先頭要素のインデックスを 0 とし，以降の要素はそれぞれ続く自然数と対応する．

ある配列に対して，その保持する要素の数を長さまたはサイズという．長さが 0 の配列を空の配列と呼ぶ．配列が保持するデータに応じた “型” のようなものは存在しないが，一部の命令はすべての要素が整数（展開すると整数を表す文字列になるトークン列またはカウンタレジスタ）の場合にのみ動作する．そのため本資料においては，便宜的にすべての要素が整数であるものを整数配列と呼ぶことにする．

1.2　表記法

紙面で配列を表現する場合は，各要素を順にカンマ区切りで並べ，配列全体を角括弧で囲って表現することとする．以下にいくつか例を示す．

[]（空の配列），[1, 2, 3]，[abc, \foo, def, \bar]

1.3　インデックスの負数表記

既に説明したように，インデックスとは配列のある 1 つの要素に対応する自然数である．各インデックスは必ず 1 つ別の表記をもつ．すなわち一番最後の要素のインデックスを −1 とし，そこから 1 つ前の要素に遡るごとに 1 ずつ減じた値をインデックスの負数表記と呼ぶ．以下に，一例を示しておく．

配列	[	A,	B,	C	]
インデックス		0	1	2	
負数表記		−3	−2	−1	

この負数表記は，インデックスを必要とする任意の場面で使用することが可能である．

1.4　取得と出力

TLArray パッケージにおいて，取得とは指定した制御綴（以下，ターゲットと呼ぶ）に値を格納すること，出力とは DVI や PDF に値を印字することを指す．ここでいう値とは，完全展開された状態のトークン列である．本パッケージが提供する命令の中には，取得する値が整数であることが確実であるものもあるが，そうした場合にもターゲットには「整数を表す文字トークン列」という形でしか値を渡すことは出来ない．

また配列に渡したトークン列は原則として展開されずに配列内に保持されるが，これらのトークン列を取得または出力した場合，原則としてその操作を行った時点で完全に展開されることに留意されたい．

2　配列の宣言

すべての配列は `\NewArray` で宣言してから用いる必要がある．

```
\NewArray[⟨長さ⟩][⟨初期値⟩]{⟨配列名⟩}
```

`\NewArray` は上記のような書式をもつが，このうち ⟨長さ⟩ と ⟨初期値⟩ は省略可能であり，これらを省略した場合 ⟨配列名⟩ に指定した名称をもつ空の配列を生成する．⟨配列名⟩ には基本的な半角英数字のみが使用可能であり，その他はサポート対象外とする（実際には使用できる場合もある）．また，⟨配列名⟩ に既存の配列の名前と同じ文字列を指定することはできない．

一方 ⟨長さ⟩ と ⟨初期値⟩ をいずれも指定した場合は，⟨初期値⟩ を ⟨長さ⟩ 個もつ ⟨配列名⟩ という名の配列が生成される．さらに，⟨長さ⟩ を指定して ⟨初期値⟩ を省略した場合には `\relax` を ⟨長さ⟩ 個もつ ⟨配列名⟩ という名の配列が生成される．ここで ⟨長さ⟩ のみを指定して ⟨初期値⟩ を指定することはできないので注意すること．

なお，以降に登場するすべての命令は，この `\NewArray` で宣言済みの配列に対してのみ正常に動作する．

3　配列の取得・出力

`\LengthofArray` を用いると，指定した配列の長さ（サイズ）を取得または出力することができる．なお，`\LengthofArray` には `\SizeofArray` という別名も存在する．

```
\LengthofArray[⟨ターゲット⟩]{⟨配列名⟩}
\SizeofArray[⟨ターゲット⟩]{⟨配列名⟩}
```

上記の書式において，⟨ターゲット⟩ を指定した場合は取得，省略した場合は出力が行われる（以下同様）．

配列の特定の要素を取得または出力するには `\GetArray` 命令を使用する．

```
\GetArray[⟨ターゲット⟩]{⟨配列名⟩}{⟨インデックス⟩}
```

さらに `\ShowArray` 命令を用いれば，すべての要素をカンマ区切りで出力することも可能である．

```
\ShowArray{⟨配列名⟩}
```

4　配列の末尾・先頭に対する操作

4.1　Push と Pop

\PushArray 命令は，指定した配列の末尾に要素を追加する．

\PushArray{⟨配列名⟩}{⟨値⟩}

ここで \PushArray 命令は ⟨値⟩ を展開せずに配列に追加するが，後ろに * をつけて \PushArray* とすると ⟨値⟩ を完全展開したものを配列に追加することができる．

一方，\PopArray 命令は末尾の要素を取得または出力し，その要素を配列から除去する．

\PopArray[⟨ターゲット⟩]{⟨配列名⟩}

4.2　Shift と Unshift

\ShiftArray 命令は先頭の要素を取得または出力し，その要素を配列から除去する．その後，配列に要素が残存していた場合にはすべての要素のインデックスが 1 つ前にずらされる．

\ShiftArray[⟨ターゲット⟩]{⟨配列名⟩}

\UnshiftArray は指定した配列の先頭に要素を追加する（* を付けると完全展開してから追加）．このとき元々存在していた要素のインデックスは 1 つずつ後ろにずらされる．

\UnshiftArray{⟨配列名⟩}{⟨値⟩}

※これらの命令は内部的に重い処理を行うため（特に大きな配列に対しては）あまり乱用しない方がよい．

5　配列に対する操作

5.1　要素の挿入

\InsertArray を用いると，任意の配列の任意のインデックスに要素を追加することができる（* を付けると完全展開してから追加される）．

\InsertArray{⟨配列名⟩}{⟨インデックス⟩}{⟨値⟩}

上記の書式で指定したインデックスが当該配列の長さ未満であった場合，挿入位置以降に元々存在した要素のインデックスは 1 つずつ後ろにずらされる．すなわち，[1, 2, 3] という形の配列 foo に対して \InsertArray{foo}{1}{x} を行うと [1, x, 2, 3] となる．

また，インデックスに当該配列の長さ以上の値を指定した場合，元々の配列の末尾と挿入位置の間の要素には \relax が代入された状態となり，配列の長さは指定インデックスから 1 を減じた値に変更される．

※長さが L の配列 A に対する操作で，⟨インデックス⟩ に指定する整数 i は，ほとんどの場合 $-L \leq i < L$ を満たす必要があるが，\InsertArray だけが例外的に $-L \leq i$ を満たす整数であればよい．

※ ⟨インデックス⟩ に -1 を指定した場合，要素挿入前の配列の最後方要素の位置に要素が挿入される．

5.2 要素の削除

\DeleteArray は任意の配列の任意の要素を削除する．このとき，削除した要素よりも後ろに存在していた要素のインデックスはすべて 1 つずつ前にずらされる．

\DeleteArray{⟨配列名⟩}{⟨インデックス⟩}

5.3 要素の更新

\SpliceArray は任意の配列の任意の要素を指定した値に更新する（* を付けると完全展開後に更新する）．

\SpliceArray{⟨配列名⟩}{⟨インデックス⟩}{⟨値⟩}

なお，この命令は指定箇所に破壊的な代入を行うためのものなので，使用時には十分注意されたい．
※ TLArray パッケージでは \relax を一般的なプログラミング言語における nil, null に相当するものと捉えている．したがって \SpliceArray 命令の第三引数である ⟨値⟩ に \relax を指定することで，任意の要素の “削除” に近いことが可能であり，不都合がない場合はこちらの方法を用いた方が \DeleteArray による削除より高速に動作する．

5.4 要素の置換

\ReplaceArray は指定した値と一致する要素を，すべて指定の値に置換する（* を付けると完全展開後に置換する）．

\ReplaceArray{⟨配列名⟩}{⟨検索値⟩}{⟨置換値⟩}

これにより，対象の配列で ⟨検索値⟩ と一致する値をもつ要素は，すべて ⟨置換値⟩ に置換される．
※ ⟨検索値⟩ と一致する要素の判定は，それぞれの値（非展開）を代入した制御綴をプリミティブ \ifx で比較することにより行われる．

5.5 要素の除去

\RemoveArray は指定した値と一致する要素を，すべて削除する（この削除は \DeleteArray と同じ）．

\RemoveArray{⟨配列名⟩}{⟨検索値⟩}

5.6 要素の交換

\SwapArray を用いると，特定の配列の 2 つの要素を入れ替えることができる．

\SwapArray{⟨配列名⟩}{⟨インデックス 1⟩}{⟨インデックス 2⟩}

さらに \XSwapArray を用いると，異なる配列間で要素を入れ替えることも可能である．

\XSwapArray{⟨配列名 1⟩}{⟨インデックス 1⟩}{⟨配列名 2⟩}{⟨インデックス 2⟩}

5.7　配列の反転

\ReverseArray は，指定した配列の要素の順序を逆転させる．

\ReverseArray{⟨配列名⟩}

5.8　要素の展開

\ExpandArray は指定した要素を，その時点で完全展開する．

\ExpandArray{⟨配列名⟩}{⟨インデックス⟩}

6　整数配列に対する操作

この節で説明する命令は整数配列に対してのみ有効である．

6.1　最大値と最小値

\MaxofArray は指定した整数配列に含まれる最大の数を取得または出力する．

\MaxofArray[⟨ターゲット⟩]{⟨配列名⟩}

逆に，\MinofArray は指定した整数配列に含まれる最小の数を取得または出力する．

\MinofArray[⟨ターゲット⟩]{⟨配列名⟩}

6.2　総和

\SumArray は指定した整数配列の全要素の総和を取得または出力する．

\SumArray[⟨ターゲット⟩]{⟨配列名⟩}

7　配列と文字列の変換

\StringtoArray で与えた文字列を 1 文字ずつに分解して，指定した配列の末尾に付加させることが可能である．

\StringtoArray{⟨配列名⟩}{⟨文字列⟩}

※ ⟨文字列⟩ に制御綴が含まれる場合，現在の実装では 1 つの制御綴は 1 つの要素として非展開のまま配列に挿入されるが，そのような挙動は「仕様」ではないので注意されたい．

逆に，\toStringArray を用いると，指定した配列のもつ要素をすべてつなげた文字列を出力することができる．ただし，要素に制御綴が含まれる場合には，その展開および実行が行われる．

\toStringArray{⟨配列名⟩}

8　繰り返し処理

\EachArray を用いると，指定した配列のすべての要素に対して特定の操作を施すことができる．

\EachArray[⟨範囲⟩]{⟨配列名⟩}{⟨操作⟩}

ここで ⟨操作⟩ には \foo{⟨配列名⟩}{⟨インデックス⟩} の形の書式をもつ TLArray パッケージの制御綴を指定することができる．また ⟨範囲⟩ は

⟨開始インデックス⟩..⟨終了インデックス⟩

という形で指定する．⟨開始インデックス⟩ が ⟨終了インデックス⟩ よりも前方の要素に対応する場合にはまず ⟨開始インデックス⟩ に対応する要素に対して処理を行い，以降 1 つずつ後方に向かって ⟨終了インデックス⟩ に対応する要素まで処理が行われる．逆に ⟨開始インデックス⟩ が ⟨終了インデックス⟩ より後方の要素に対応する場合には処理は配列の前方に向かって実行される．

なお，⟨範囲⟩ を省略した場合，\EachArray は ⟨操作⟩ に \DeleteArray を指定した際などになるべく高速な処理を行うことができるよう配列の最後方から最初の要素まで順に処理を施すようになっている．つまり，以下の 2 つはまったく同じ挙動を示す．

```
\EachArray[-1..0]{bar}{\ExpandArray}
\EachArray{bar}{\ExpandArray}
```

※\EachArray{bar}{\GetArray} とすると配列の内容を逆順に（区切りのない「文字列」として）出力することができる．これは \ReverseArray してから \toStringArray するよりも高速に動作する．

9　配列の破棄

\DestructArray を用いると，任意の配列を破棄するこができる．

\DestructArray{⟨配列名⟩}

配列を破棄すると，当該の配列にはいかなる操作もできなくなる．また，一度破棄した配列と同名の配列を新たに宣言した場合，その配列はまったく新しい配列となり，破棄前の状態は一切引き継がない．
※ TEX プログラムの中で「宣言した配列は必ず破棄しなければならない」ということはまったくない．配列の破棄は，主として一時的に利用するために宣言した配列に対して使用することを想定している．