PAD Editor α

コマンドリファレンス

この文書の内容は 2019 年 12 月時点の PAD Editor α の情報です。PAD Editor “筑前煮” ではいくつかのコマンドが追加されましたがまだ文書化されていません。

この文書に利用されているフォントは、Rounded M^+ 2c です。フォント Rounded M^+は M^+ OUTLINE FONTS から作られました。Rounded M^+は自家製フォント工房から取得可能です。M^+ OUTLINE FONTS は mplus-fonts.osdn.jp から取得可能です。両フォントは M^+ FONTS LICENSE 下で利用可能です。

目次

1 PAD とは....4
2 PAD Editor について....5
2.1 PAD Editor を利用するには....5
3 コマンド文のフォーマット....6
3.1 コマンド文の文法....6
3.2 コマンド文の要素....6
3.3 変数名....6
4 コマンドの解説....7
4.1 制御コマンド....7
4.1.1 $....7
4.1.2 BREAK (break)....8
4.1.3 CONTINUE (continue)....8
4.2 初期化・代入コマンド....9
4.2.1 LET (let)....9
4.2.2 MOV (move)....9
4.2.3 GLOB (global)....10
4.2.4 SWAP (swap)....10
4.2.5 PI (pi)....10
4.2.6 RAND (random)....11
4.3 加減算・増減分コマンド....12
4.3.1 ADD (add)....12
4.3.2 SUB (subtract)....12
4.3.3 INC (increase)....12
4.3.4 DEC (decrease)....13
4.3.5 NEG (negate)....13
4.4 乗除算・剰余・累乗コマンド....14
4.4.1 MUL (multiply)....14
4.4.2 DIV (divide)....14
4.4.3 MOD (modulo)....14
4.4.4 POW (power)....15
4.4.5 SQRT (square root)....15
4.5 小数処理コマンド....16
4.5.1 INT (integer)....16
4.5.2 FLOOR (floor)....16
4.5.3 CEIL (ceiling)....16
4.5.4 ROUND (round)....17
4.6 ビット演算コマンド....18
4.6.1 AND (and)....18
4.6.2 XOR (exclusive or)....18
4.6.3 OR (or)....18
4.6.4 SHL (shift left)....19
4.6.5 SHR (shift right)....19

4.6.6 NOT (not)....19
4.7 スタック操作コマンド....20
4.7.1 PUSH (push)....20
4.7.2 POP (pop)....20
4.8 配列・ポインタ操作コマンド....21
4.8.1 DIM (dimension)....21
4.8.2 SETA (set array)....21
4.8.3 SETE (set element)....22
4.8.4 GETA (get address)....22
4.8.5 GETP (get pointer)....22
4.8.6 SWAPA (swap for array)....23
4.9 入出力コマンド....24
4.9.1 PRINT (print)....24
4.9.2 PRINTA (print for array)....24
4.9.3 INPUT (input)....25
4.10 比較コマンド....26
4.10.1 NEQ (not equal)....26
4.10.2 EQ (equal)....26
4.10.3 LT (less than)....26
4.10.4 GT (greater than)....26
4.10.5 LTE (less than or equal)....27
4.10.6 GTE (greater than or equal)....27
5 システム変数・システム定数について....28
5.1 システム変数....28
5.1.1 PADAOFFSET....28
5.1.2 STACKOFFSET....28
5.1.3 RETURN....28
5.2 システム定数....28
5.2.1 MEMORY....28
5.2.2 NULL....28
5.2.3 EOF....29
6 参考文献....30

1　PAD とは

構造化チャートとは、高級言語や構造遭プログラミング技術が普及してきた際に目立ったフローチャートの欠点(例えば、if と goto のみによってあらゆる分岐と反復を行う点)を克服しようとしたものです。構造化チャートの例には NS チャートや HCP、PSD などがあります。

ここで扱う **PAD** (Problem Analysis Diagram) とは、㈱日立製作所中央研究所の二村良彦氏らによって 1979 年頃に考案された、構造化チャートの一つです。ワーニエ図の問題点を改善する研究から生まれました。ISO/IEC 8631:1989 (JIS X 0128:1988) の Annex A (informative) に収録されています。PAD はプログラマブル電卓から大型計算機までの多くの機種に対する OS、アプリケーションなど各種のプログラムの開発に使用されてきました。

PAD は構造化プログラムを 2 次元的に展開した図式に描かれます。特に、PAD が標準的に備えている制御構造は PASCAL に基づいて定めてあるので、PAD は PASCAL を 2 時限的に展開したような図式であり、PASCAL Diagram ということもできます。

Figure 1 は、1 から 10 の総和を求めるプログラムをフローチャートと PAD で描いた例です。PAD では繰り返し処理の内容が構造的に描かれるので、フローチャートに比べて全体の流れがわかりやすくなっています。

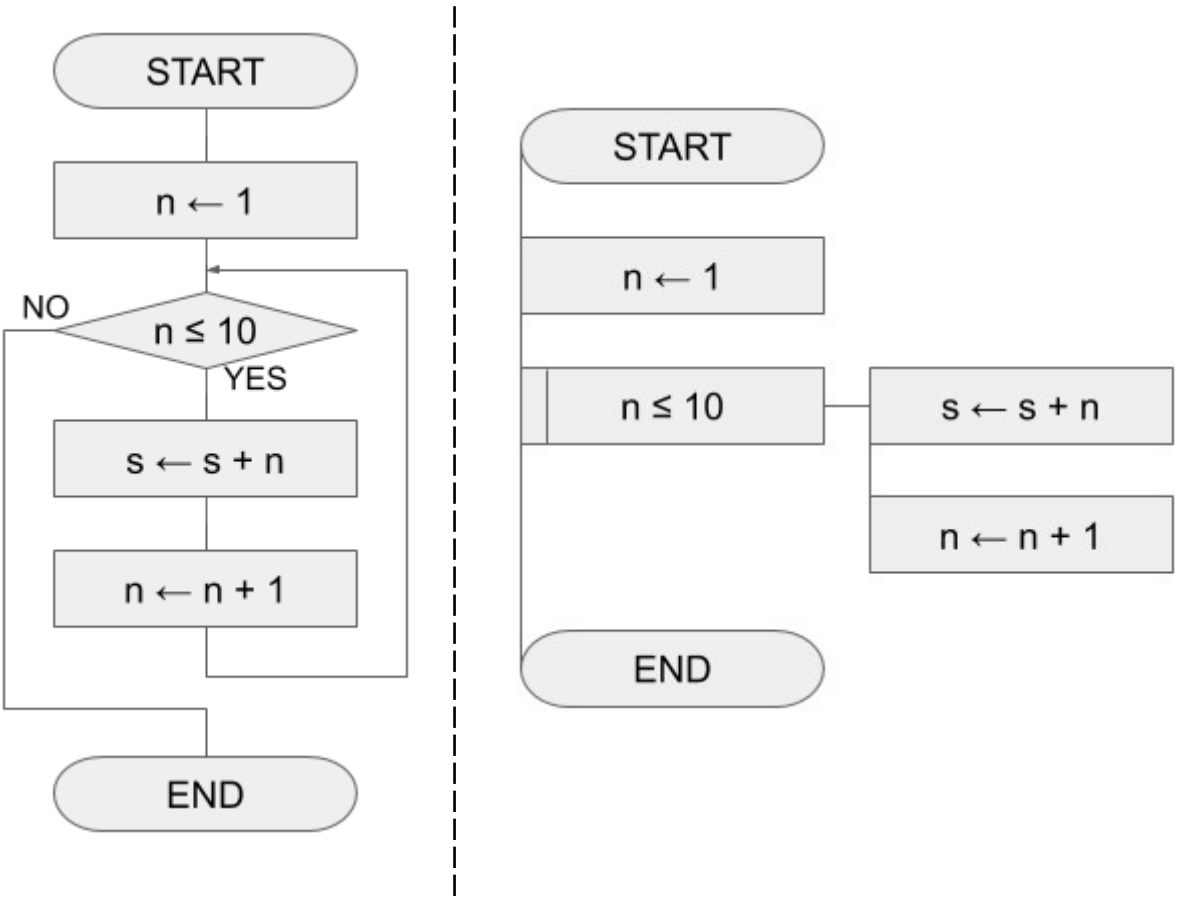

Figure 1: フローチャート (左) と PAD (右)

2　PAD Editor について

PAD Editor は KusaReMKN による PAD 開発・実行環境です。北陸職業能力開発大学校情報技術科 PAD Simulator Project による **PAD Simulator** の UI 改善、機能向上、及びマルチプラットフォーム対応を目標に開発されています。

現時点で PAD Editor は α テスト版であり、低機能で潜在的なバグが多く含まれます。また、動作を記述する言語が独自言語であり、動作の最小単位が低レベルであるため利用におけるハードルが高くなっています。これらは、以降のアップデートで改善・改修される予定です。

PAD Editor はテューリング完全です。これは、1 次元セルラオートマトンである Rule110 を再現することで確認されました。Rule110 は Matthew Cook 氏によってテューリング完全性が証明されています。

2.1　PAD Editor を利用するには

PAD Editor はブラウザ上で動作する WEB アプリケーションとして提供されます。ダウンロードやインストール作業は一切必要ありません。開発段階の最新バージョンは kusaremkn.web.fc2.com から利用可能です。

推奨ブラウザは、より推奨する順に、Google Chrome, Firefox です。それ以外のブラウザでは動作を保証しません。IE はもってのほかです。滅びて下さい。

既知のバグに、Firefox 上で動作中に無限ループに陥るとに大量のメモリ領域を占有するというものがあります。Chrome では “エラーが発生しました” となり、動作が停止します。

PAD Editor を利用して生じた損失、損害の如何なるものにおいても製作者はその責任を負い兼ねます。

3　コマンド文のフォーマット

3.1 コマンド文の文法

PAD Editor のコマンド文はポーランド記法(前置記法; 演算子が式の先頭に配置される記法)で表記されます。しかし、$コマンドはこの限りではありません。

コマンドの要素はそれぞれスペースで区切られる必要があります。

1 つのコマンドブロックには 1 つのコマンド文を記述することができます。

3.2 コマンド文の要素

コマンド文はコマンドとオペランドに大分できます。コマンド文に含まれるアルファベットの大文字小文字は区別されます。

コマンドとはコマンド文の最初の語を指し、コマンド文のオペランドの扱いや処理を示します。これは常に大文字です。

オペランドとはコマンドに続く引数を指します。オペランドにはいくつかの種類があります。

変数: 記号 var で表されます。変数とは値を入れる入れ物のことです。

定数: 記号 num で表されます。これは 123, -1e3, .25 などが含まれます。

配列: 記号 arr で表されます。配列は変数に含まれます。

ポインタ: 記号 ptr で表されます。ポインタは配列に含まれ、変数に含まれます。

変数または定数: 記号 v/n で表されます。

配列またはポインタ: 記号 a/p で表されます。

3.3 変数名

変数名は、数として解釈可能な文字以外で始まるスペースを含まない文字列が利用可能です。しかし、今後のアップデートによって仕様が変更される恐れがあるため、アルファベットもしくはアンダースコアから始まる、アルファベット、数字、もしくはアンダースコアを含む文字列の利用を推奨します。

利用不可な変数名: 123　+1a　.012hoge　1e3　etc...

利用可能な変数名: a　hoge　_abc123　$var　j[n+2*p]　*p　(123)　etc...

推奨される変数名: a　hoge　_abc123　etc...

4　コマンドの解説

この章では、PAD Editor αに組み込まれているコマンドについて解説します。

4.1 制御コマンド

4.1.1　　$

書式

$ *statement*

説明

$コマンドは、後続する中置記法などの文を解釈し、翻訳しようと試みます。すべての式の要素はスペースで区切られれている必要があります。

$コマンドは現時点で以下の文を解釈できます(括弧内は翻訳先のコマンド)。

- 単純代入
  - a = b (LET a b)
  - a <- b (MOV a b)
  - a := b (GLOB a b)
- 交換演算
  - a <=> b (SWAP a b)
- 条件演算 (条件ブロックのみで有効)
  - a != b (NEQ a b)
  - a == b (EQ a b)
  - a < b (LT a b)
  - a > b (GT a b)
  - a <= b (LTE a b)
  - a >= b (GTE a b)
- 複合代入
  - a += b (ADD a b)
  - a -= b (SUB a b)
  - a *= b (MUL a b)
  - a /= b (DIV a b)
  - a %= b (MOD a b)
  - a &= b (AND a b)
  - a ^= b (XOR a b)
  - a |= b (OR a b)
  - a <<= b (SHL a b)
  - a >>= b (SHR a b)
- 単項演算
  - a ++ (INC a)
  - a -- (DEC a)

操作

解釈され、翻訳されたコマンドによります。

4.1.2 BREAK (break)

書式

BREAK

説明

BREAK コマンドは、C 言語の予約語である break に相当します。

ループ内で利用する場合は、そのループを脱出します。

ループ外(main の直下など)で利用する場合はプログラムを脱出しますが、この動作は副作用的であるので利用を推奨しません。この動作は今後のアップデートで削除されるかもしれません。

操作

break;

4.1.3 CONTINUE (continue)

書式

CONTINUE

説明

CONTINUE コマンドは、C 言語の予約語である continue に相当します。

ループ内で利用する場合は、この文以降を実行せずにループの終端に移動します。

ループ外(main の直下など)で利用する場合はプログラムを脱出しますが、この動作は副作用的であるので利用を推奨しません。この動作は今後のアップデートで削除されるかもしれません。

操作

continue;

4.2 初期化・代入コマンド

4.2.1 LET (let)

書式

LET *var v/n*

説明

LET コマンドは、変数に値を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。動作は MOV コマンドと全く同じです。

LET コマンドは、主に定数を変数に代入(定義)するときに利用されます。

動作

var ← v/n;

4.2.2 MOV (move)

書式

MOV *var v/n*

説明

MOV コマンドは、変数に値を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。動作は LET コマンドと全く同じです。

MOV コマンドは、主に変数をコピーするときに利用されます。

動作

var ← v/n;

4.2.3 GLOB (global)

書式

GLOB *glob-var v/n*

説明

GLOB コマンドは、グローバル変数に値を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。グローバル変数に値を書き込める命令は GLOB だけであることに注意してください。

動作

glob-var ← v/n;

4.2.4 SWAP (swap)

書式

SWAP *var1 var2*

説明

SWAP コマンドは、変数値を交換します。左辺値、右辺値ともに変数である必要があります。

動作

temp ← var1;

var1 ← var2;

var2 ← temp;

4.2.5 PI (pi)

書式

PI *var*

説明

PI コマンドは、変数に円周率 π を代入します。

動作

var ← π;

4.2.6 RAND (random)

書式

RAND var

説明

RAND コマンドは、変数に乱数値を代入します。乱数値は 0–RAND_MAX の整数値です。

動作

var ← rand();

4.3 加減算・増減分コマンド

4.3.1 ADD (add)

書式

ADD *var v/n*

説明

ADD コマンドは、左辺に左辺値と右辺値の和を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。

動作

var ← var + v/n;

4.3.2 SUB (subtract)

書式

SUB *var v/n*

説明

SUB コマンドは、左辺に左辺値と右辺値の差を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。

動作

var ← var - v/n;

4.3.3 INC (increase)

書式

INC *var*

説明

INC コマンドは、変数値を 1 増加します。

動作

var ← var + 1;

4.3.4 DEC (decrease)

書式

DEC *var*

説明

DEC コマンドは、変数値を 1 減少します。

動作

var ← var - 1;

4.3.5 NEG (negate)

書式

NEG *var*

説明

NEG コマンドは、変数値を正負反転します。

動作

var ← -var;

4.4 乗除算・剰余・累乗コマンド

4.4.1 MUL (multiply)

書式

MUL *var v/n*

説明

MUL コマンドは、左辺に左辺値と右辺値の積を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。

動作

var ← var * v/n;

4.4.2 DIV (divide)

書式

DIV *var v/n*

説明

DIV コマンドは、左辺に左辺値と右辺値の商を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。

動作

var ← var / v/n;

4.4.3 MOD (modulo)

書式

MOD *var v/n*

説明

MOD コマンドは、左辺に左辺値を右辺値で割った余りを代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var % v/n;

4.4.4 POW (power)

書式

POW *var v/n*

説明

POW コマンドは、左辺に左辺値を右辺値で累乗した値を代入します。左辺値は変数である必要があります。右辺値は定数、変数のいずれも指定可能です。

動作

var ← var ** v/n;

4.4.5 SQRT (square root)

書式

SQRT *var*

説明

SQRT コマンドは、指定された変数値の平方根を変数に代入します。

動作

var ← var ** 0.5;

4.5 小数処理コマンド

4.5.1 INT (integer)

書式

INT *var*

説明

INT コマンドは、指定された変数値以下の最大の整数値を代入します。動作は FLOOR コマンドと全く同じです。

動作

var ← floor(var);

4.5.2 FLOOR (floor)

書式

FLOOR *var*

説明

FLOOR コマンドは、指定された変数値以下の最大の整数値を代入します。動作は INT コマンドと全く同じです。

動作

var ← floor(var);

4.5.3 CEIL (ceiling)

書式

CEIL *var*

説明

CEIL コマンドは、指定された変数値以上の最小の整数値を代入します。

動作

var ← ceil(var);

4.5.4 ROUND (round)

書式

ROUND *var*

説明

ROUND コマンドは、指定された変数値を四捨五入した値を代入します。

動作

var ← round(var);

4.6 ビット演算コマンド

4.6.1 AND (and)

書式

AND *var v/n*

説明

AND コマンドは、左辺に左辺値と右辺値のビット毎の積を代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var AND v/n;

4.6.2 XOR (exclusive or)

書式

XOR *var v/n*

説明

XOR コマンドは、左辺に左辺値と右辺値のビット毎の排他的論理和を代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var XOR v/n;

4.6.3 OR (or)

書式

OR *var v/n*

説明

OR コマンドは、左辺に左辺値と右辺値のビット毎の和を代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var OR v/n;

4.6.4 SHL (shift left)

書式

SHL *var v/n*

説明

SHL コマンドは、左辺に左辺値に右辺値回 2 を掛けた値を代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var << v/n;

4.6.5 SHR (shift right)

書式

SHR *var v/n*

説明

SHR コマンドは、左辺に左辺値を右辺値回 2 をで割った値を代入します。左辺値は変数であり、整数が格納されている必要があります。右辺値は定数、変数が指定可能で整数である必要があります。

動作

var ← var >> v/n;

4.6.6 NOT (not)

書式

NOT *var*

説明

NOT コマンドは、変数に変数値をビットごとに反転した値を代入します。変数には整数が格納されている必要があります。

動作

var ← NOT var;

4.7 スタック操作コマンド

4.7.1 PUSH (push)

書式

PUSH *v/n*

説明

PUSH コマンドは、変数、または定数の値をスタックに積みます。

動作

stack[stack_offset] ← v/n;

stack_offset ← stack_offset + 1;

4.7.2 POP (pop)

書式

POP *var*

説明

POP コマンドは、スタックの最上位から値を変数に取り出します。

動作

stack_offset ← stack_offset - 1;

var ← stack[stack_offset];

4.8 配列・ポインタ操作コマンド

4.8.1 DIM (dimension)

書式

DIM *a/p v/n*

説明

DIM コマンドは、右辺値で指定される分の領域を確保し、左辺値にその領域を指す(長さの情報を含んだ)ポインタを代入します。左辺値は変数である必要があります。右辺値は定数、変数が指定可能です。

動作

a/p.offset ← memory_offset;

a/p.length ← v/n;

memory_offset ← memory_offset + v/n;

4.8.2 SETA (set array)

書式

SETA *a/p v/n1* [*v/n2* [*...*]]

説明

SETA コマンドは、初項の指す配列に後続の値を配列の要素としてそれぞれ代入します。配列はすべての項を代入するのに十分な領域を持っている必要があります。

動作

For (n ← 1 to argc – 1)

    a/p[n] ← argv[n+1];

Next

4.8.3 SETE (set element)

書式

SETE *a/p v/n1 v/n2*

説明

SETE コマンドは、第 1 項の配列の第 2 項番目に第 3 項の値を代入します。

動作

a/p[v/n1] ← v/n2;

4.8.4 GETA (get address)

書式

GETA *var a/p v/n*

説明

GETA コマンドは、第 1 項の変数に第 2 項の指す配列の第 3 項番目のメモリ番地を代入します。

動作

var ← a/p.offset + v/n;

4.8.5 GETP (get pointer)

書式

GETP *a/p1 a/p2 v/n*

説明

GETP コマンドは、第 1 項の変数に第 2 項の指す配列の第 3 項番目を指すポインタを代入します。

動作

v/a1 → a/p2[v/n];

4.8.6 SWAPA (swap for array)

書式

SWAPA *a/p v/n1 v/n2*

説明

SWAPA コマンドは、第 1 項の配列のオフセット第 2 項番目と第 3 項番目の値を交換します。

動作

tmp ← a/p[v/n1];

a/p[v/n1] ← a/p[v/n2];

a/p[v/n2] ← tmp;

4.9 入出力コマンド

4.9.1 PRINT (print)

書式

PRINT *v/n* [*v/n* [⋯]]

説明

PRINT コマンドは、与えられた引数の値を表示して改行します。引数には変数、定数が指定可能です。複数の値を指定した場合は、スペースで区切られます。

動作

```
For (n ← 1 to argc)
    print argv[n];
    print ' ';
Next
print '\n';
```

4.9.2 PRINTA (print for array)

書式

PRINTA *a/p* [*v/n*]

説明

PRINTA コマンドは、左辺で指定される配列の要素を表示します。右辺が指定されていて配列の長さより小さい値の場合は、右辺値個数分の要素のみを表示します。左辺にポインタを指定した場合は右辺は必須項目になります。

動作

```
For (n ← 1 to a/p.length or v/n)
    print a/p[n];
    print ' ';
Next
print '\n';
```

4.9.3 INPUT (input)

書式

INPUT *v/n* [*v/n* [⋯]]

説明

INPUT コマンドは、ユーザに入力を促し変数に値を代入します。ユーザからの入力が不十分であったり無効な値が含まれる場合は、ユーザに再度入力を促します。ユーザが入力をキャンセルした場合システム変数 RETURN にシステム定数 EOF が代入されます。

動作

```
Try
    For (n ← 1 to argc)
        argv[n] ← input();
    Next
Catch (e)
    If e = InputError
        Retry;
    Elif e = InputCanceled
        return EOF;
    Endif
```

4.10 比較コマンド

比較コマンドは、条件式でのみ利用可能です。

4.10.1 NEQ (not equal)

書式

NEQ *v/n1 v/n2*

説明

v/n1 と v/n2 が等しくない場合、真になります。

4.10.2 EQ (equal)

書式

EQ *v/n1 v/n2*

説明

v/n1 と v/n2 が等しい場合、真になります。

4.10.3 LT (less than)

書式

LT *v/n1 v/n2*

説明

v/n1 が v/n2 より小さい場合、真になります。

4.10.4 GT (greater than)

書式

GT *v/n1 v/n2*

説明

v/n1 が v/n2 より大きい場合、真になります。

4.10.5 LTE (less than or equal)

書式

LTE *v/n1 v/n2*

説明

v/n1 が v/n2 以下の場合、真になります。

4.10.6 GTE (greater than or equal)

書式

GTE *v/n1 v/n2*

説明

v/n1 が v/n2 以上の場合、真になります。

5　システム変数・システム定数について

PAD Editor には、システム変数やシステム定数という特別な変数があります。これらはシステムの情報を保持したり、便宜上確保されるものがあります。これらの値を変更するとシステムが不安定になる恐れがあるので変更しないようにしてください。

この章ではシステム変数とシステム定数の内容と用途などを説明します。

5.1 システム変数

5.1.1　PADAOFFSET

この値は数値です。配列領域として次に割り当てがされる領域のオフセット値が記憶されています。

DIM コマンドで配列を確保する場合にこの値が利用されます。

5.1.2　STACKOFFSET

この値は数値です。グローバル領域に確保されます。次にスタックが積み込まれるスタックのオフセット値が記憶されています。

PUSH コマンドと POP コマンドでスタックを操作する際にこの値は利用されます。

5.1.3　RETURN

この値は不定値です。直前の動作の返値が記憶されています。

現時点では、この値を操作するコマンドは INPUT コマンドのみです。将来的には関数の戻り値や様々なコマンドの返値を得るために使われるかもしれません。

5.2 システム定数

5.2.1　MEMORY

この値はポインタ値です。配列領域として利用される領域の先頭を指しています。

GETA コマンドで取得されるオフセット値は、このポインタからのオフセットであるともいえます。異なる配列間の値の受け渡し(SWAPA コマンドを利用)などの場合に利用できるかもしれません。

5.2.2　NULL

この値はポインタ値です。グローバル領域に確保されます。配列領域として利用されることのない無効な領域を指しています。

現時点ではこの値の利用可能性はありません。

5.2.3 EOF

この値は数値です。グローバル領域に確保されます。−1 を記憶しています。

この値は C 言語の stdio.h 内で定義される定数 EOF との互換のために用意されました。入出力系の動作でエラー値として利用されます。

6　参考文献

- フローチャート - Wikipedia
- PAD（Problem Analysis Diagram）によるプログラムの設計および作成