PAD Editor “筑前煮”

取扱説明書・内部仕様書

この文書は **PAD Editor “筑前煮”**の操作方法、及び動作について説明するものです。この文書の内容は 2019 年 12 月時点の情報です。最新版の文書が公開されている場合はそちらをご利用ください。最新版の文書は `https://kusaremkn.web.fc2.com/manual/` から取得可能です。

ここから下はメモ

文章内の太字はフォントを * bold にしてね。**こんな感じ**。斜体は使えないよ。

とりあえず PAD の内部動作まで OK

1. 特徴
    1. 安全上の注意
    2. 使用上のお願い
    3. 各部の名前
2. 使い方
    1. ●●したいとき
3. 困ったとき
4. 仕様

目次

1 フローチャートと PAD....4
2 PAD Editor について....5
3 PAD Editor の各部の名称....6
3.1 コマンドパレットとコマンドブロックの種類....7
3.2 操作ボタンとそれぞれの動作....8
3.2.1 実行ボタン....8
3.2.2 保存ボタン....8
3.2.3 開くボタン....8
3.2.4 クリアボタン....8
3.2.5 PAD-Text 変換ボタン....8
3.2.6 実行結果クリアボタン....8
3.2.7 ライブラリ取得ボタン....8
3.2.8 Text-PAD 変換ボタン....8
4 コマンドリファレンス....9
4.1 制御コマンド....9
4.1.1 BREAK....9
4.1.2 CONTINUE....9
4.1.3 INT3....9
4.1.4 RETURN....9
4.2 置換コマンド($)....10
4.3 パースコマンド(C:)....11
4.4 レガシコマンド....12
4.4.1 LET....12
4.4.2 MOV....12
4.4.3 GLOB....12
4.4.4 SWAP....12
4.4.5 PI....12
4.4.6 RAND....12
4.4.7 ADD....13
4.4.8 SUB....13
4.4.9 INC....13
4.4.10 DEC....13
4.4.11 NEG....13
4.4.12 MUL....13
4.4.13 DIV....13
4.4.14 MOD....14
4.4.15 POW....14
4.4.16 SQRT....14
4.4.17 INT....14
4.4.18 FLOOR....14
4.4.19 CEIL....14
4.4.20 ROUND....14
4.4.21 AND....14

4.4.22 XOR....15
4.4.23 OR....15
4.4.24 SHL....15
4.4.25 SHR....15
4.4.26 NOT....15
4.4.27 PUSH....15
4.4.28 POP....15
4.4.29 DIM....16
4.4.30 SETA....16
4.4.31 SETE....16
4.4.32 GETA....16
4.4.33 GETP....16
4.4.34 SWAPA....16
4.4.35 PRINT....16
4.4.36 PRINTA....17
4.4.37 INPUT....17
4.4.38 NEQ....17
4.4.39 EQ....17
4.4.40 LT....17
4.4.41 GT....17
4.4.42 LTE....17
4.4.43 GTE....18
5 システム変数・システム定数....19
5.1 システム変数....19
5.1.1 PADAOFFSET....19
5.1.2 STACKOFFSET....19
5.1.3 RETURN....19
5.2 システム定数....19
5.2.1 MEMORY....19
5.2.2 RAND_MAX....19
5.2.3 NULL....19
5.2.4 null....20
5.2.5 undefined....20
5.2.6 EOF....20
6 PAD Editor の動作....21
6.1 関数群....21
6.2 PAD 描画域....23
7 参考文献....23

1　フローチャートと PAD

フローチャートは、プロセスやシステム、アルゴリズムなどの各ステップの操作とその流れを示す図です。Figure 1はその一例です。この例は、1 から 10 までの総和を求めるプロセスが描かれています。プロセスの各段階を箱で表し、その流れを矢印で表します。フローチャートはプロセスやプログラムの設計及び文書化に使われます。フロチャートを作成・利用することによる利点は、①問題解決の方法を視覚的に明確に表せる、②処理手順を追いやすい、手順に問題がある時、それを発見・修正するのが容易になる、③問題解決を複数人数で行うとき、担当箇所の明確化や、説明する際の理解向上に役立つ、などがあります。しかし、コンピュータ科学の観点からは、フローチャートはプログラミングにおいて無条件ジャンプと条件分岐のみ(goto と if のみ)を制御機構として利用している構造となり、スパゲッティプログラムとなってしまうなどの問題があります。

構造化チャートは、高級言語や構造化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これらの欠点を克服しようとしたものです。段階的詳細化などの問題解決の手法を反映するような工夫がなされました。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。

PAD (Problem Analysis Diagram: 問題分析図)は、(株)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二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 を二次元的に展開したような図式であり、PASCAL Diagram とも言えます。Figure 2は Figure 1と同様の処理を PAD で描いたものです。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ります。

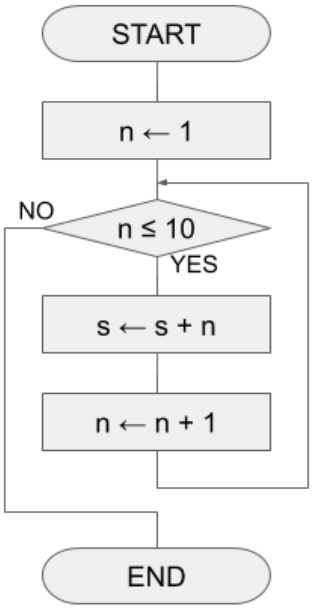


Figure 1: フローチャートの例

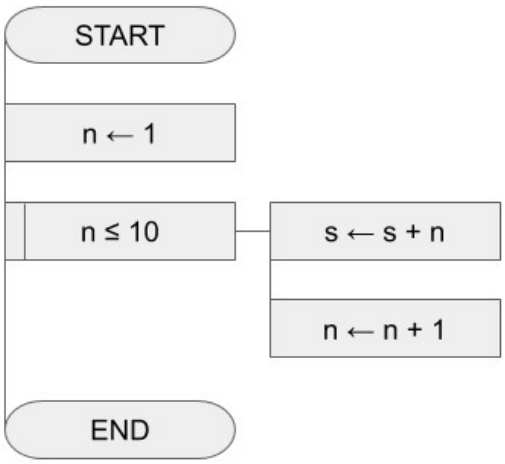


Figure 2: PAD の例

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** のユーザインタフェース改善、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、低機能であり、潜在的なバグが多く含まれます。また、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。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。安定版の最新バージョンは `www.kusaremkn.tk/mknpad/`から利用可能です。安定版ではライブラリの機能が利用できません。

動作確認済みの推奨ブラウザは 2020/01/05 時点で以下の通りです(より推奨する順に):

- Google Chrome, Chromium
- Mozilla Firefox
- Microsoft Edge (Chromium)
- Opera
- mac OS で動作する Safari

上記以外のブラウザでは動作を保証しません。Internet Explorer や Microsoft Edge (EdgeHTML)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　PAD Editor の各部の名称

PAD Editor の画面の一例を Figure 3 に示します。各要素は図に振られた①–⑥の部分に大分されます。

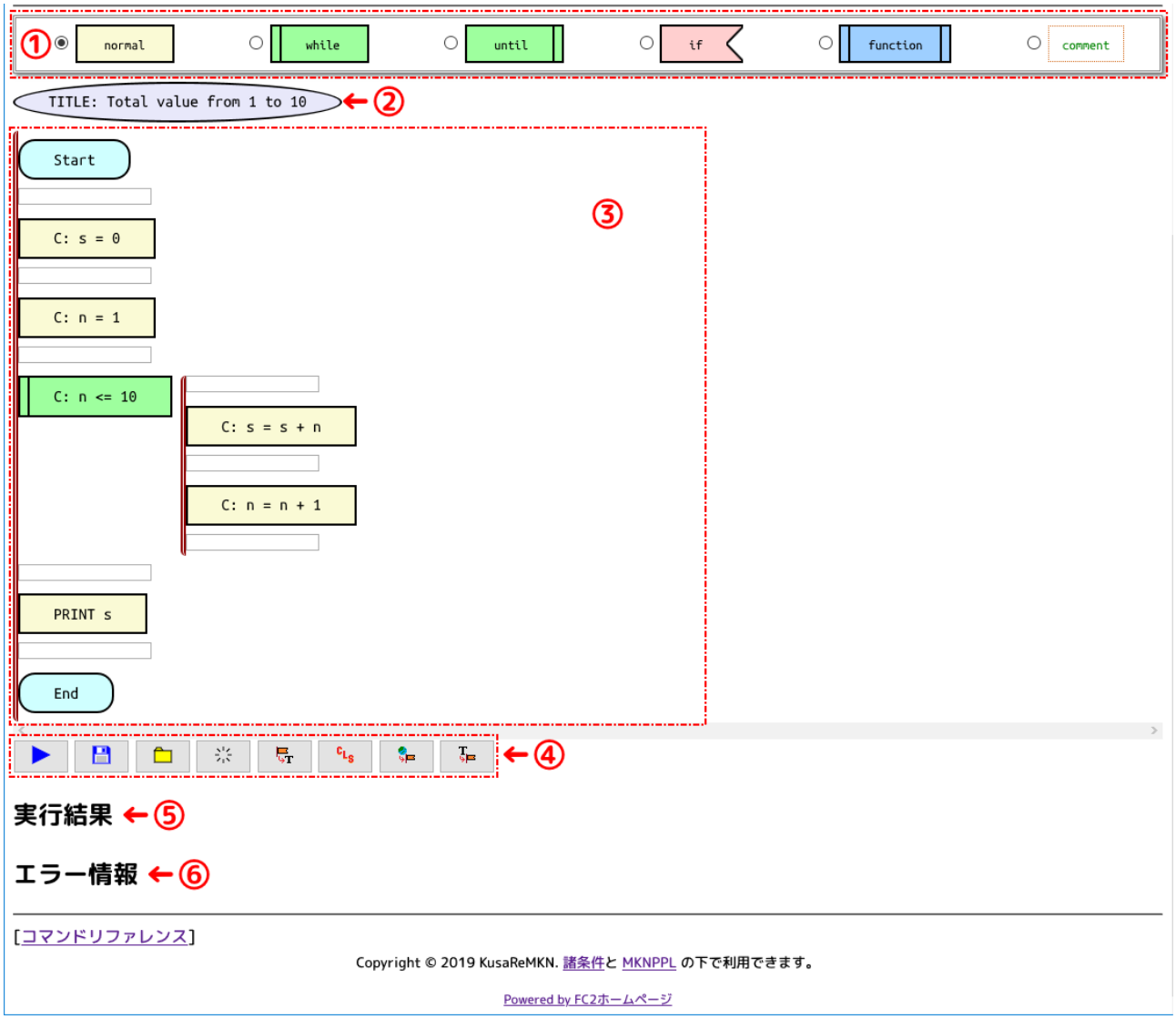


Figure 3: PAD Editor “筑前煮” の画面の一例

① ブロックパレット

配置する PAD のブロックを指定します。ラジオスイッチの選択状態はユーザが最後に選択したブロックを示しています。

② プログラムタイトル

プログラムのタイトルが表示されます。ダブルクリックで編集できます。

③ PAD 描画領域

PAD を記述する領域です。それぞれのブロックはダブルクリックで編集できます。

④ 操作ボタン

プログラムの実行、ファイルへの保存などができます。

⑤ 実行結果出力領域

プログラムの出力はここに表示されます。

⑥ エラー情報表示領域

プログラムのエラー情報はここに表示されます。

3.1 コマンドパレットとコマンドブロックの種類

ここではブロックの種類とその働きについて説明します。

normal ブロックは処理を表します。フローチャートの単純な四角形と同様です。

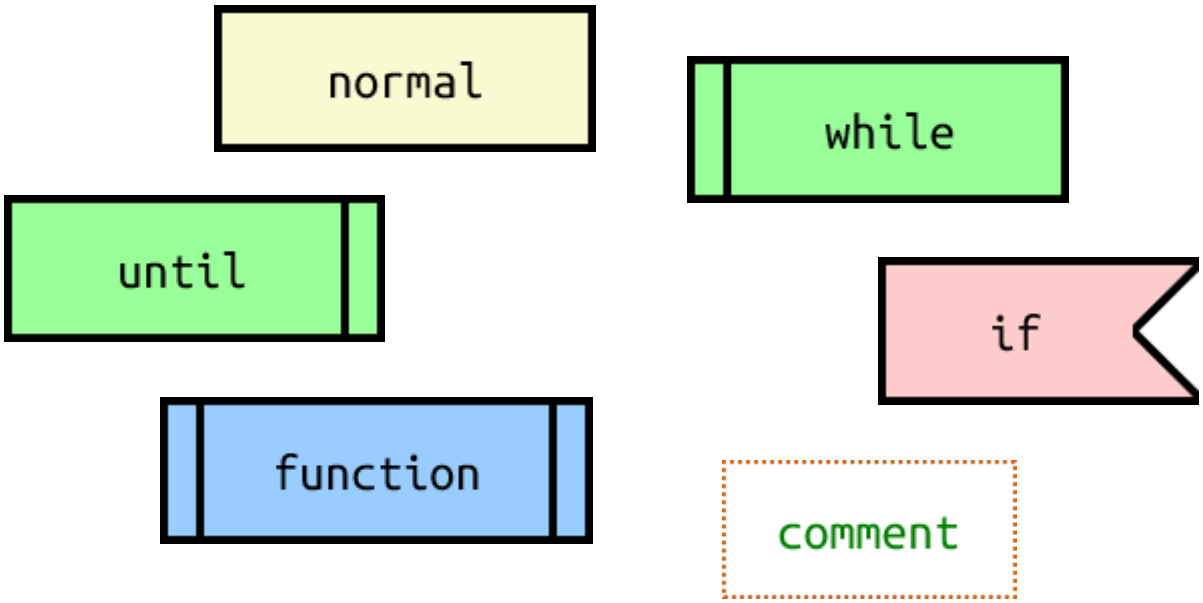
while ブロックは前判定繰り返し処理の条件式を表します。C 言語の while 文に相当します。PAD Editor において、このブロックに入るコマンドは比較コマンド系です。このブロックを配置すると自動的に繰り返し処理を記述するネストが追加されます。

until ブロックは後判定繰り返し処理の条件式を表します。C 言語の do-while 文に相当します。PAD Editor において、このブロックに入るコマンドは比較コマンド系です。このブロックを配置すると自動的に繰り返し処理を記述するネストが追加されます。

if ブロックは条件分岐の条件式を表します。C 言語の if-else 文に相当します。PAD Editor において、このブロックに入るコマンドは比較コマンド系です。このブロックを配置すると自動的に分岐処理を記述するネストが追加されます。

function ブロックは定期済み処理を表します。このブロックを配置すると、指定した定義済み処理が存在しない場合は処理を定義するネストが PAD の最後尾に追加されます。

comment ブロックはコメントを表します。このブロックの内容は処理に影響しません。



3.2 操作ボタンとそれぞれの動作

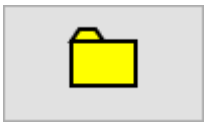
3.2.1 実行ボタン

PAD のプログラムを実行します。

3.2.2 保存ボタン

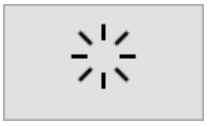
PAD をファイルとして保存します。blob に対応している必要があります。

3.2.3 開くボタン



ファイルとして保存された PAD のデータをロードします。

3.2.4 クリアボタン



表示されている PAD をクリアします。実はページをリロードします。

3.2.5 PAD-Text 変換ボタン

表示されている PAD をテキスト形式に変換します。

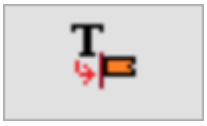
3.2.6 実行結果クリアボタン

表示されている実行結果をクリアします。

3.2.7 ライブラリ取得ボタン

サーバ上のライブラリから PAD のデータを取得して展開します。

3.2.8 Text-PAD 変換ボタン



テキスト形式に変換されたデータから PAD を復元します。

4　コマンドリファレンス

PAD Editor “筑前煮”の各ブロックに記述するコマンドについて説明します。

4.1 制御コマンド

制御コマンドはプログラムの動作を制御するコマンドです。BREAK、CONTINUE、INT3、及びRETURNの４つのコマンドが含まれます。

4.1.1　BREAK

BREAKコマンドはC言語の予約語breakと同様の機能を提供します。ループ内で利用してループを脱出します。ループ外で利用すると関数を脱出しますが、この動作は副作用的であるためこの動作を利用しないでください。関数を脱出するにはRETURNコマンドを利用します。

書式: BREAK

4.1.2　CONTINUE

CONTINUEコマンドはC言語の予約語continueと同様の機能を提供します。ループ内で利用して以降のコマンドを実行せずにループの終端に移動します。ループ外で利用すると関数を脱出しますがこの動作は副作用的であるためこの動作を利用しないでください。関数を脱出するにはRETURNコマンドを利用します。

書式: CONTINUE

4.1.3　INT3

INT3コマンドはブレイクポイントコマンドです。このコマンドを実行しようとすると、BreakPointエラーを伴ってプログラムの動作を停止します。同時に各変数の値が表示されるので、デバッグ時のブレイクポイントとして利用可能です。このコマンドの動作は今後変更される可能性があります。

書式: INT3

4.1.4　RETURN

RETURNコマンドはC言語の予約語returnと同様の機能を提供します。関数内で利用して以降のコマンドを実行せずに関数を脱出します。このコマンドは0–1個の引数を取ります。コマンドに引数が指定された場合、システム変数RETURNにその値が代入されます。

書式: RETURN［(変数または定数)］

4.2 置換コマンド($)

置換コマンド $ は、後続する演算子を用いた一部の演算式をレガシコマンドに翻訳しようと試みます。これはちょうど C 言語におけるプリプロセッサのような動作です。翻訳に成功した場合、翻訳されたコマンド文がレガシコマンドパーサにより実行されます。翻訳に失敗した場合、置換コマンド文そのものがレガシコマンドパーサによって実行され、UnknownCommand エラーで停止します。演算式の翻訳時にオペランドのチェックは行われません。翻訳先のコマンド文で引数エラーなどが起きた場合は、レガシコマンドパーサによってエラー処理されます。

書式: $（翻訳される式)

対応している演算子と置換先のコマンドの対応を表 1 に示します。

表 1: 置換コマンド対応表

翻訳元	翻訳先
Foo = Bar	LET Foo Bar
Foo <- Bar	MOV Foo Bar
Foo := Bar	GLOB Foo Bar
Foo ++	INC Foo
Foo –	DEC Foo
Foo != Bar	NEQ Foo Bar
Foo == Bar	EQ Foo Bar
Foo < Bar	LT Foo Bar
Foo > Bar	GT Foo Bar
Foo <= Bar	LTE Foo Bar
Foo >= Bar	GTE Foo Bar
Foo <=> Bar	SWAP Foo Bar
Foo += Bar	ADD Foo Bar
Foo -= Bar	SUB Foo Bar
Foo *= Bar	MUL Foo Bar
Foo /= Bar	DIV Foo Bar
Foo %= Bar	MOD Foo Bar
Foo &= Bar	AND Foo Bar
Foo ^= Bar	XOR Foo Bar
Foo |= Bar	OR Foo Bar
Foo <<= Bar	SHL Foo Bar
Foo >>= Bar	SHR Foo Bar

4.3 パースコマンド(C:)

パースコマンド C: は、C 言語の二項演算子を利用した複合式を解釈しようと試みます。置換コマンドとの違いは、式をレガシコマンドに変換するのではなく軽量パーサによって式の値を確定的に計算する点です。そのため、オペランドエラーが発生した場合はパースコマンドによるエラー(C:から始まるエラーメッセージ)として報告されます。このコマンドの式の値はシステム変数 RETURN に格納されます。このコマンドは比較コマンドとしても利用可能です。

書式: `C:` (パースされる式)

対応している演算子とその優先順位を表 2 に示します。

表 2: パースコマンド対応の演算子、優先順位、及び結合規則

優先順位	演算子	結合規則
1 (より高い)	括弧 ( )	左から右
2	(なし)	右から左
3	乗除算、剰余 * / %	左から右
4	加減算 + -	〃
5	シフト演算 << >>	〃
6	比較演算 < <= > >=	〃
7	比較演算 == !=	〃
8	ビット積 &	〃
9	ビット差 ^	〃
10	ビット和 |	〃
11	論理積 &&	〃
12	論理和 ||	〃
13	(なし)	右から左
14	単純代入 =	〃
14	複合代入	〃
14	+= -=	〃
14	*= /= %=	〃
14	<<= >>=	〃
14	&= ^= \=	〃
15 (より低い)	(なし)	左から右

4.4 レガシコマンド

レガシコマンドは PAD Editor (仮) で実装された簡易実行機能のためのコマンドです。PAD Editor の各ブロックと組み合わせて利用することであらゆる計算をすることが可能です。

4.4.1 LET

変数に値を代入します。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。MOV コマンドと全く同じ動作をします。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。

書式: `LET %Var %VarNum`

4.4.2 MOV

変数に値を代入します。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。LET コマンドと全く同じ動作をします。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。

書式: `MOV %Var %VarNum`

4.4.3 GLOB

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。

書式: `GLOB %GVar %VarNum`

4.4.4 SWAP

変数値を交換します。引数はすべて変数である必要があります。

書式: `SWAP %Var %Var`

4.4.5 PI

変数に円周率 π を代入します。

書式: `PI %Var`

4.4.6 RAND

変数に 0–RAND_MAX の整数乱数を代入します。

書式: `RAND %Var`

4.4.7 ADD

第 1 引数に第 1 引数と第 2 引数の和を代入します。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。

書式: `ADD %Var %VarNum`

4.4.8 SUB

第 1 引数に第 1 引数と第 2 引数の差を代入します。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。

書式: `ADD %Var %VarNum`

4.4.9 INC

変数値を 1 だけインクレメントします。

書式: `INC %Var`

4.4.10 DEC

変数値を 1 だけデクレメントします。

書式: `DEC %Var`

4.4.11 NEG

変数値をその 2 の補数で置き換えます。変数値の 0 からの減算と同等です。

書式: `NEG %Var`

4.4.12 MUL

第 1 引数に第 1 引数と第 2 引数の積を代入します。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。

書式: `MUL %Var %VarNum`

4.4.13 DIV

第 1 引数に第 1 引数と第 2 引数の商を代入します。演算の結果は浮動小数点小数になります。第 1 引数は変数である必要があります。第 2 引数は定数、変数のいずれも指定可能です。

書式: `DIV %Var %VarNum`

4.4.14 MOD

第１引数に、第１引数を第２引数で割った余りを代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `MOD %Var %VarNum`

4.4.15 POW

第１引数に第１引数を第２引数で累乗したものを代入します。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `POW %Var %VarNum`

4.4.16 SQRT

変数値をその平方根で置き換えます。

書式: `SQRT %Var`

4.4.17 INT

変数値をその値以下の最大の整数値で置き換えます。FLOOR コマンドと全く同じ動作をします。

書式: `INT %Var`

4.4.18 FLOOR

変数値をその値以下の最大の整数値で置き換えます。INT コマンドと全く同じ動作をします。

書式: `FLOOR %Var`

4.4.19 CEIL

変数値をその値以上の最小の整数値で置き換えます。

書式: `CEIL %Var`

4.4.20 ROUND

変数値をその値を四捨五入した整数値で置き換えます。

書式: `ROUND %Var`

4.4.21 AND

第１引数に第１引数と第２引数のビットごとの積を代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `AND %Var %VarNum`

4.4.22 XOR

第１引数に第１引数と第２引数のビットごとの排他的論理和を代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `XOR %Var %VarNum`

4.4.23 OR

第１引数に第１引数と第２引数のビットごとの和を代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `OR %Var %VarNum`

4.4.24 SHL

第１引数に第１引数に第２引数回２を掛けた値を代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `SHL %Var %VarNum`

4.4.25 SHR

第１引数に第１引数を第２引数回２で割った値を代入します。両引数値は整数である必要があります。第１引数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `SHR %Var %VarNum`

4.4.26 NOT

変数値をその１の補数で置き換えます。変数値のビット単位の否定演算です。

書式: `NOT %Var`

4.4.27 PUSH

引数値をスタックに積み上げます。

書式: `PUSH %Any`

4.4.28 POP

変数にスタックから積み下ろした値を代入します。

書式: `POP %Var`

4.4.29 DIM

第２引数で指定されたサイズの領域を確保し、第１引数にその領域を指す(長さの情報を含んだ)ポインタを代入します。第１変数は変数である必要があります。第２引数は定数、引数のいずれも指定可能です。

書式: `DIM %Var %VarNum`

4.4.30 SETA

第１引数の指す配列にそれ以降の引数値を配列の要素としてそれぞれストアします。配列はすべての要素をストアされるのに十分な領域を持っている必要があります。

書式: `SETA %Ptr %VarNum [%VarNum […]]`

4.4.31 SETE

第１引数の指す配列の第２引数番目に第３引数値をストアします。

書式: `SETE %Ptr %VarNum %VarNum`

4.4.32 GETA

第１引数に第２引数の指す配列の第３引数番目のメモリ番地を示す整数値を代入します。

書式: `GETA %Var %Ptr %VarNum`

4.4.33 GETP

第１引数に第２引数の指す配列の第３引数番目を指すポインタを代入します。

書式: `GETP %Var %Ptr %VarNum`

4.4.34 SWAPA

第１引数の配列の第２引数番目と第３引数番目の値を交換します。

書式: `SWAPA %Ptr %VarNum %VarNum`

4.4.35 PRINT

与えられた引数の値を表示して改行します。引数には定数、変数のいずれも指定可能です。複数の値を指定した場合はスペースで区切られます。

書式: `PRINT %Any [%Any […]]`

4.4.36 PRINTA

第 1 引数の指す配列の要素の値を表示して改行します。第 2 引数には表示する要素の数を指定できます。第 1 引数がポインタである場合は第 2 引数は必須です。

書式: `PRINTA %Ptr [%VarNum]`

4.4.37 INPUT

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はシステム変数 RETURN にシステム定数 EOF が代入されます。この場合、変数の値は操作されません。

書式: `INPUT %Var [%Var […]]`

4.4.38 NEQ

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値と第 2 引数の値が等しくないときに真になります。

書式: `NEQ %Any %Any`

4.4.39 EQ

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値と第 2 引数の値が等しいときに真になります。

書式: `EQ %Any %Any`

4.4.40 LT

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値が第 2 引数の値より小さいときに真になります。

書式: `LT %Any %Any`

4.4.41 GT

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値が第 2 引数の値より大きいときに真になります。

書式: `GT %Any %Any`

4.4.42 LTE

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値が第 2 引数の値以下のときに真になります。

書式: `LTE %Any %Any`

4.4.43 GTE

このコマンドは条件式で利用する比較コマンドです。第 1 引数の値が第 2 引数の値以上のときに真になります。

書式: `GTE %Any %Any`

5　システム変数・システム定数

PAD Editor には、その内部状態を保持するための**システム変数**や内部状態を定義したり便宜上確保されうる**システム定数**があります。これらの値はグローバル変数や局所変数として確保されています。これらの値を変更すると内部状態が不安定になる恐れがあります。

5.1 システム変数

5.1.1　PADAOFFSET

この値は局所変数として確保されています。この値は配列として利用されていない最初の領域のオフセットを示す整数値です。DIM コマンドによって利用されます。初期値は 1 です。

5.1.2　STACKOFFSET

この値はグローバル変数として確保されています。この値はスタックの現在の深さを示す整数値です。PUSH コマンド、POP コマンドによって利用されます。初期値は 1 です。

5.1.3　RETURN

この値はグローバル変数として確保されています。この値は実行結果を表す値です。RETURN コマンドによって利用されるほか、INPUT コマンドやパースコマンドでも利用されます。初期値は undefined です。

5.2 システム定数

5.2.1　MEMORY

この値は局所変数として確保されています。この値は配列として利用される領域の最初の領域を指すポインタです。GETA コマンドで得られるオフセットはこのポインタからのオフセットです。

5.2.2　RAND_MAX

この値は局所変数として確保されています。この値は RAND コマンドで得られる乱数の最大値を定義しています。この値を変更することで乱数の最大値を変更できますが、この動作は推奨されません値は 32767 です。

5.2.3　NULL

この値はグローバル変数として確保されています。この値は配列として利用されない領域を指すポインタです。

5.2.4 null

この値はグローバル変数として確保されています。JavaScript の値 `null` を格納しています。

5.2.5 undefined

この値はグローバル変数として確保されています。この値は一度も利用されていない変数の初期値と同じ値です。JavaScript の値 `undefined` を格納しています。

5.2.6 EOF

この値はグローバル変数として確保されています。値は−1 です。

6　PAD Editor の動作

この章では、PAD Editor の内部動作を説明します。

この章では、HTML のタグ、クラス、及び ID 名を以下のように省略表記します。

TagName.ClassName#IDName → <TagName class="ClassName" id="IDName">

6.1 関数群

それぞれのファイルに含まれるプログラム要素の一覧です。

- /js/createblock.js
  - `CreateBlock(ClickedEmptyBlock): boolean`　コマンドブロックの挿入処理
- /js/editblock.js
  - `EditBlock(ClickedCommandBlock): boolean`　コマンドブロックの編集
- /js/padio.js
  - `PadOutput: HTMLElement`　実行結果の表示先(`#PadOutput`)
  - `ErrorOutput: HTMLElement`　エラー情報の表示先(`#cmdlin`)
  - `IOInit(): void`　出力要素の初期化
  - `PrintOut(str): void`　実行結果の表示
  - `PrintErr(str): void`　エラー情報の表示
- /js/runpad.js
  - `PADStart(): void`　プログラム実行前ルーティン
  - `PADGLOBVARS: {}`　グローバル変数記憶域
  - `PADVARS: {}`　局所変数記憶域
  - `FullCommand: string`　コマンド全文のバッファ
  - `RunPAD(PAD, n?): any`　プログラム実行関数
    - `CalcCCode(Command): void`　軽量パーサ
    - `IsVar(ValueStr): boolean`　変数判定関数
    - `GetValue(ValueStr): any`　値取得関数
    - `GetVarName(str): string`　内部変数名取得関数
    - `Natural2PAD(Command): void`　置換コマンドパーサ
    - `RUN(Command): "return"|"break"|"continue"|"next"`　コマンド実行関数

- /js/saveload.js
  - `Text2Pad(str): boolean` テキスト形式から PAD への変換
  - `Pad2Text(): string` PAD からテキスト形式への変換
  - `SavePAD2File(): boolean` PAD をファイルに保存する処理
  - `GetLibrary(name?): boolean` サーバから PAD を取得する
  - `ErrorFileCount: number` エラーファイルを読み込んだ回数
  - `LoadPADfromFile(): void` ファイルから PAD を読み込み
- /js/updateblock.js
  - `SelectedBlock: any` 選択されたブロックの HTML 要素
  - `BlockSelected: boolean` ブロックが選択されているか否か
  - `(): void` 虚無をクリックしたときのハンドラ
  - `Select(e): void` クリックされたときのハンドラ
  - `ClickRight(e): boolean` 右クリックされたときのハンドラ
  - `UpdateBlocks(t?): void` ブロックにリスナを付加

6.2 PAD 描画域

PAD の描画域は、`div.PAD#PAD` です。これは同時にプログラムメモリとして利用されています。

7 MKNPAD ファイル

保存ボタンを利用して出力されるファイルである MKNPAD ファイルについて説明します。

7.1 ファイルの形式

ファイルの形式は表 3 ように定義されています。

表 3: PAD Editor で利用するファイル

拡張子	MIME	ファイルの説明
`.Chikuzen-ni.mknpad`	`application/x-mknpad`	PAD Editor “筑前煮”で出力されるデータ
`.mknpad`	`application/x-mknpad`	PAD Editor で出力されるデータ
`.mknpad.data`	`application/x-mknpad-data`	`mknpad` ファイルのデータ部のみを含むもの

7.2 ファイルの実体

7.2.1 mknpad ファイル

`mknpad` ファイルの実体は JSON (`application/json`)です。最小の構成で、表 4 のキーとそれに対応するデータを含んでいる必要があります:

表 4: mknpad ファイルの最低要件

キー	データ型	データ内容
`Length`	`Number`	`Data` の文字列長
`Data`	`String`	`#PAD` の `innerHTML` を `escape` し `base64` エンコードしたもの

7.2.2 .mknpad.data ファイル

`mknpad.data` ファイルは `mknpad` ファイルのキー `Data` に対応するデータのみを記述したファイルです。サーバに配置され、ライブラリ処理に利用されます。

8 開発履歴

PAD Editor “筑前煮”に至るまでの詳細は https://kusaremkn.web.fc2.com/diary.html を参照してください。

- 2019/11/15
  - PadSim として製作開始。
  - normal ブロックの配置、編集、削除ができるようになった。

- 2019/11/24
    - PAD Editor (仮)に改称。
    - 実行機能(レガシコマンド)の実装。
- 2019/12/06
    - PAD Editor “筑前煮”に改称。
    - 翻訳コマンドの完成。
    - レガシコマンドによるチューリング完全性の確認。

9 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