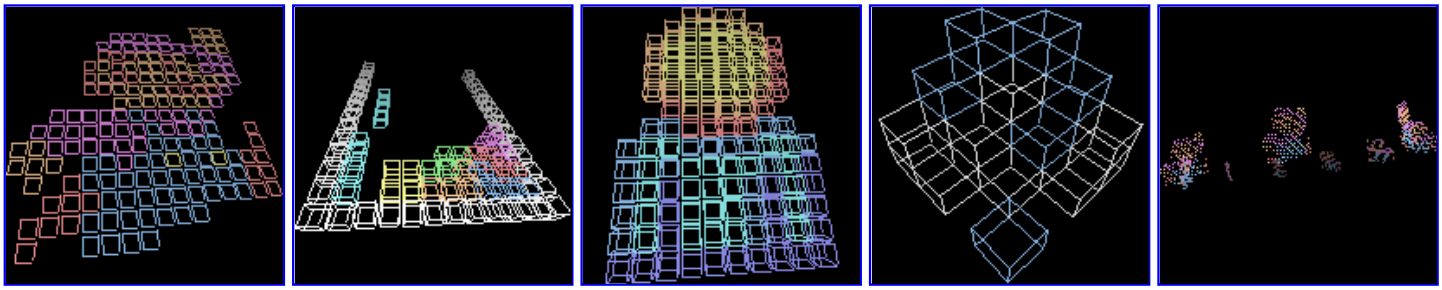
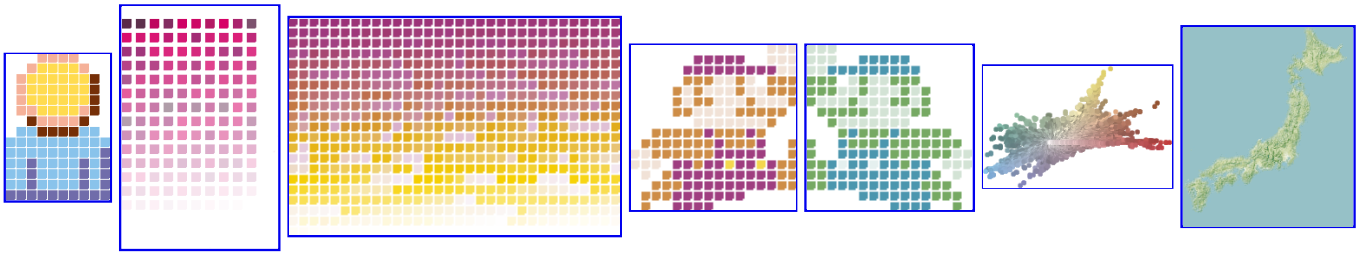
CX's UWSC Diary

「こくばん.in」投稿サンプル（※ 以下の画像は id:cx20 がWindows自動化ツールUWSCを使用して「こくばん.in」に投稿したものです。）

「はてなロクロ」投稿サンプル



「はてなハイク」投稿サンプル



<「超難問クイズ」を力技で解いて…

2010-01-31

UWSC 基礎文法最速マスター

20:49 | tB 6 users

UWSC の文法一覧です。他の言語をある程度知っている人はこれを読めば UWSC の基礎をマスターして UWSC のスクリプトを修正できるようになっています。簡易リファレンスとしても利用できると思いますので、これは足りないと思うものがあれば教えてください。

1. 基礎

変数宣言の強制

ソースコードの先頭に「Option Explicit」を入れるようにしましょう。変数の宣言が強制されコードの品質も上がります。

```
Option Explicit
```

メッセージの表示

```
Print "Hello world."          // Print 文はデバッグ用です。結果はログウィンドウに表示されます
```

```
MsgBox( "Hello world" )        // メッセージボックスにメッセージを表示します
```

コメント

コメントです。

```
// コメント
```

変数の宣言

変数の宣言です。UWSC には変数、配列変数、ハッシュ変数があります。

```
// 変数
Dim num

// 配列変数
Dim students[10]

// ハッシュ変数
HashTbl month_num
```

スクリプトの実行

コマンドラインよりスクリプトを実行するには次のようにします。

```
uwsc //K script.uws
```

出力結果をファイルに書き出すには Option LogPath, Option LogFile を指定します。

```
Option LogPath="C:¥TEMP¥UWSC.log"
Option LogFile=3 // 1:ログ出力しない, 2:日付（時分秒）付けない, 3:日付（時分秒）を付ける
```

デバッガの起動

UWSC Free 版にはデバッガの機能はありません。UWSC Pro（シェアウェア）を検討してみてください。

2. 数値

数値の表現

スカラ変数に数値を代入できます。整数でも小数でも代入できます。

```
Dim num = 1
Dim num = 1.234
Dim num = 100000000
```

四則演算

四則演算です。

```
num = 1 + 1 // 2
num = 1 - 1 // 0
num = 1 * 2 // 2
num = 1 / 2 // 0.5
```

余りと商の求め方です。商を求めるには普通の除算を行った後にInt関数で整数部を取り出します。

```
# 商
num = Int(3/2)

# 余り
num = 3 Mod 2
```

インクリメントとデクリメント

インクリメントとデクリメントです。

```
// インクリメント
i = i + 1

// デクリメント
i = i - 1
```

3. 文字列

文字列の表現

文字列は「"」ダブルクォーテーションで囲みます。変数にタブや改行コードをセットするには「<#TAB>」や「<#CR>」を用います。

```
str1 = "abc"
str2 = "a<#TAB>bc<#CR>"
```

文字列操作

各種文字列操作です。

```
// 結合
join = "aaa" + "bbb"

// 分割
// 分割操作の構文はありません

// 長さ
len = Length("abcdef")            // 6
len = Length("あいうえお")        // 5（文字数を数えるには Length 関数を使用します）
len = LengthB("あいうえお")       // 10（文字のバイト数を数えるには LengthB 関数を使用します）

// 切り出し
substr = Copy("abcd", 1, 2)       // ab

// 検索
pos = Pos("cd", "abcd")           // 見つかった場合はその位置、見つからなった場合は 0 が返ります
pos = Pos("うえ", "あいうえお")   // 3（文字数で扱う場合は Pos 関数を使用します）
pos = PosB("うえ", "あいうえお")  // 5（バイト数で扱う場合は PosB 関数を使用します）
```

4. 配列

配列変数の宣言と代入

配列です。

```
// 配列の宣言
Dim array[2]

// 配列への代入
array[0] = 1
array[1] = 2
array[2] = 3
```

配列の要素の参照と代入

配列の要素を参照と代入です。

```
// 要素の参照
Print array[0]
Print array[1]

// 要素の代入
array[0] = 1;
array[1] = 2;
```

配列の個数

```
num = Length(array)
```

配列の操作

配列を操作する関数です。

```
Dim array[] = 1, 2, 3

// 先頭を取り出す
first = array[0]              // first は 1

// 末尾を取り出す
last = array[Length(array)-1] // last は 3

// 末尾に追加
ReSize( array, Length(array) )
array[Length(array)-1] = 9    // array は 1,2,3,9
```

5. ハッシュ

ハッシュ変数の宣言と代入

```
HashTbl hash
hash["a"] = 1
hash["b"] = 2
```

ハッシュの要素の参照と代入

ハッシュの要素の参照と代入です。

```
// 要素の参照
Print hash["a"]
Print hash["b"]

// 要素の代入
hash["a"] = 5
hash["b"] = 7
```

ハッシュに関する操作

```
// キーの存在確認
hash["a", HASH_EXISTS]  // キーが存在すれば True を返す

// ハッシュのキーの削除
hash["a", HASH_REMOVE]  // 削除できれば True を返す
```

6. 制御文

If文

　１行に書くIf文です。ブロック形式の If 文は Ifb です。

```
If 条件 Then 式 [Else 式]
```

Ifb ～ Else 文

　ブロック形式の Ifb ～ Else 文です。VB 系と微妙に構文が違う(If→Ifb, End If→EndIf)ので注意してください。

```
Ifb 条件 Then
    式
EndIf
```

Ifb ～ ElseIf 文

　ブロック形式の Ifb ～ ElseIf 文です。

```
Ifb 条件 Then
    式
ElseIf 条件 Then
    式
EndIf
```

While 文

　While 文

```
i = 0
While i < 5

    // 処理

    i = i + 1
Wend
```

For 文

For文です。

```
For i = 0 To 4

Next
```

7. サブルーチン

Procedure 関数

戻り値を返さない処理は Procedure 関数で定義します。

```
Procedure show_sum( num1, num2 )
    Dim total
    total = num1 + num2
    Print total
Fend
```

Function 関数

戻り値を返す処理は Function 関数で定義します。

```
Function sum( num1, num2 )
    Dim total
    total = num1 + num2
    Result = total // 戻り値を指定
Fend
```

8. ファイル入出力

ファイルのオープン、クローズは、FOpen / FClose 関数を、ファイルの読み込み、書き込みは FGet / FPut 関数を使用します。

```
fid = FOpen( "C:¥temp¥hoge.txt", F_READ )
For i = 1 To FGet( fid, F_LINECOUNT )
    strLine = FGet( fid, i )
    Print strLine
Next
FClose( fid )
```

9. 知っておいたほうがよい文法

コマンドライン引数

```
For i = 0 To Length(PARAM_STR)
    Print PARAM_STR[i]
Next
```

Select ～ Case 文

Select ～ Case 文です。複数の条件がある場合に使用します。

```
Select 式
```

```
Case 式
    処理
[Case  式]
    処理
[Default]
    処理
SelEnd
```

例外処理

Try ～ Finally 文は、Try ブロックを中断した場合も、Finally ブロックを実行します。

```
Try
    処理
Finally
    処理
EndTry
```

Try ～ Except 文は、Try ブロックでエラーがあった場合に、Except ブロックを実行します。

```
Try
    処理
Except
    処理
EndTry
```

その他の基礎文法マスター

この記事は他の基本文法マスターに便乗して書いた物です。誤り等ございましたらコメント等して頂けると助かります。

- Perl基礎文法最速マスター - Perl入門～サンプルコードによるPerl入門～
- Route 477 - Ruby基礎文法最速マスター
- PHP基礎文法最速マスター | Shin x blog
- Python基礎文法最速マスター - D++のはまり日誌
- VBA基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- VBScript 基礎文法最速マスター - CX's VBScript Diary - VBScript グループ

コメントを書く

<「超難問クイズ」を力技で解いて…