

プログラム表記の例示

1 変数

通常の変数の例：`kosu`, `kingaku_kei`
(変数名は英字で始まる英数字と『`_`』の並び)
配列変数の例：`Tokuten[3]`, `Data[2,4]` (配列名は先頭文字が大文字)
※特に説明がない場合、配列の要素を指定する添字は `0` から始まる

2 文字列

文字列はダブルクォーテーション (『`"`』) で囲む
`moji = "I'll be back."`
`message = "祇園精舎の" + "鐘の声"` ※`+`で連結できる

3 代入文

`kosu = 3` , `kingaku = 300` ※複数文を1行で表記できる
`kingaku_goukei = kingaku * kosu`
`namae = "Komaba"`
`Data = [10,20,30,40,50,60]`
`Tokuten` のすべての値を `0` にする
`nyuryoku = 【外部からの入力】`

4 算術演算

加減乗除の四則演算は、『`+`』、『`-`』、『`*`』、『`/`』で表す
整数の除算では、商（整数）を『`÷`』で、余りを『`%`』で表す
べき乗は『`**`』で表す

5 比較演算

『`==`』(等しい), 『`!=`』(等しくない), 『`>`』, 『`<`』, 『`>=`』, 『`<=`』

6 論理演算

『`and`』(論理積), 『`or`』(論理和), 『`not`』(否定)

7 関数

値を返す関数例：`kazu = 要素数(Data)`
`saikoro = 整数(乱数()) * 6) + 1`
値を返さない関数例：表示する(`Data`)
表示する(`Kamoku[i]`, "の得点は", `Tensu[i]`, "です")
※「表示する」関数はカンマ区切りで文字列や数値を連結できる
※「表示する」関数以外は基本的に問題中に説明あり

8 制御文（条件分岐）

もし <code>x < 3</code> ならば:	もし <code>x >= 3</code> ならば:
<code>x = x + 1</code>	<code>x = x - 1</code>
L <code>y = y + 1</code>	そうでなくもし <code>x < 0</code> ならば:
	<code>x = x * 2</code>
もし <code>x == 3</code> ならば:	そうでなければ:
<code>x = x - 1</code>	L <code>y = y * 2</code>
そうでなければ:	
L <code>y = y * 2</code>	※ と L で制御範囲を表し, L は制御文の終わりを示す

9 制御文（繰返し）

`x` を `0` から `9` まで `1` ずつ増やしながら繰り返す:

```
L goukei = goukei + Data[x]
```

※「減らしながら」もある

`n < 10` の間繰り返す:

```
| goukei = goukei + n
```

```
L n = n + 1
```

※ | と L で制御範囲を表し, L は制御文の終わりを示す

10 コメント

`atai = 乱数()` #`0` 以上 `1` 未満のランダムな小数を `atai` に代入する
※1行内において#以降の記述は処理の対象とならない

参考

「表示する」関数内での改行は、出力結果には反映されない。

プログラム

```
表示する("工芸品", kougeihin, "...",  
        "部員", tantou, ":",  
        Akibi[tantou], "日目～",  
        Akibi[tantou] + サ, "日目")
```

出力結果

工芸品 1 … 部員 1 : 1 日目～ 4 日目
(以下略)