

# 「情報関係基礎」プログラミング問題で 省略されている変数初期化部分に関する WaPEN, PyPENの実装

中西渉

watayan@meigaku.ac.jp  
名古屋高等学校

2021-12-05

## 1 はじめに

## 2 DNCL の概要

## 3 関連研究

- DNCL 学習環境
- センター試験等の問題で省略された部分

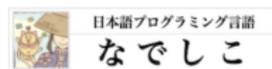
## 4 センター試験等のプログラミング問題における省略部分および WaPEN と PyPEN の追加実装

- PyPEN から WaPEN への移植
- 長さの決まっている配列
- 長さの決まっていない配列
- 2次元配列
- Web ストレージを用いた File I/O (もどき)

## 5 おわりに

# 1. はじめに

## 「なでしこ」がDNCL対応



2021年10月13日

プレスリリース

日本語プログラミング言語なでしこ

なでしこ3 大学入学共通テスト手順記述標準言語

DNCL 対応のお知らせ

日本語プログラミング言語なでしこは、**なでしこ3が大学入学共通テスト手順記述標準言語(DNCL)に対応した**ことについてプレスリリースをいたしました。

DNCLとは大学入学共通テスト(大学入試センター試験)で利用される日本語のプログラミング疑似言語です。

(2021/10/13 プレスリリースより)

# 対応表で書き換えればいいだけ

## DNCL 制御文2

例

もし  $x < 3$  ならば  $x ← x + 1$  を実行する

例

もし  $x < 3$  ならば

|  $x ← x + 1$

を実行し、そうでなければ

|  $x ← x - 1$

を実行する

## なでしこ3 記載例

例

!インデント構文

もし、 $x < 3$ ならば、 $x = x + 1$

→ [プログラム実行](#)

例

!インデント構文

もし、 $x < 3$ ならば、

$x = x + 1$

違えば

$x = x - 1$

→ [プログラム実行](#)

(なでしこ3 DNCL 対応表より)

そのままでは動かない？

→でもセンター試験等の問題自体、そのままでは動かない

∴初期化が省略されている

## 本稿の主旨

- 省略された初期化部分を調査
- そのために WaPEN, PyPEN に施した追加実装

① はじめに

② DNCL の概要

③ 関連研究

- DNCL 学習環境
- センター試験等の問題で省略された部分

④ センター試験等のプログラミング問題における省略部分および WaPEN と PyPEN の追加実装

- PyPEN から WaPEN への移植
- 長さの決まっている配列
- 長さの決まっていない配列
- 2次元配列
- Web ストレージを用いた File I/O (もどき)

⑤ おわりに

## 2. DNCL の概要

### DNCL

- センター試験等「情報関係基礎」プログラミング問題で使用
- 日本語を基本とした擬似言語
- 言語仕様は Web で公開（共通テストに合わせて更新）
  - 変数は宣言不要
  - 配列（添字は 1 から，たまに 0 から）
  - 臨時で関数が用意されることもある

## if 相当

もし～ならば  
| 《処理》  
そうでなければ  
| 《処理》  
を実行する

## while-wend 相当

～の間,  
| 《処理》  
を繰り返す

## do-until 相当

繰り返し,  
| 《処理》  
を, ～になるまで実行する

## for-next 相当

《変数》を《値》から《値》まで《値》ずつ増やしなが  
ら,  
| 《処理》  
を繰り返す

2025年度からの共通テスト

← 試作問題，サンプル問題が公表

「新」DNCL (Python 風?) ... 詳細は不明

### if 相当

もし～ならば：

《処理》

そうでなければ：

《処理》

### while-wend 相当

～の間：

《処理》

### do-until 相当

### for-next 相当

《変数》を《値》から《値》まで《値》ずつ増やしながら：

《処理》

① はじめに

② DNCL の概要

③ 関連研究

- DNCL 学習環境
- センター試験等の問題で省略された部分

④ センター試験等のプログラミング問題における省略部分および WaPEN と PyPEN の追加実装

- PyPEN から WaPEN への移植
- 長さの決まっている配列
- 長さの決まっていない配列
- 2次元配列
- Web ストレージを用いた File I/O (もどき)

⑤ おわりに

## 3.1 DNCL 学習環境

Java アプリケーション

[PEN](#) 大阪学院大学, 大阪市立大学で開発

[PenFlowchart](#) PEN にフローチャートを付加

Java アプリケーションゆえに...

- JRE 必要
- インストール必要

→ Web アプリケーションが開発される

# どんくり

- C に似た構文の言語との相互変換
- 変数一覧機能
- プロファイル機能

The screenshot shows the DNCL (Donkuri) environment. At the top, there are tabs for '実行' (Execute), '保存' (Save), '読み込み' (Load), and 'サンプル一覧' (Sample List). The main area is divided into two panes. The left pane, titled 'DNCL 英語表示', contains C-like code for a program named 'どんくり'. The code defines an array of names and their corresponding 'tokuten' values, then iterates through them to print each name and its value. The right pane shows the output of the program, which lists the names and their values: あきよ: 81, なつお: 77, はるこ: 73, ふゆき: 68.

```
1 Name[1]="はるこ", Tokuten[1]=73
2 Name[2]="なつお", Tokuten[2]=77
3 Name[3]="あきよ", Tokuten[3]=81
4 Name[4]="ふゆき", Tokuten[4]=68
5
6 j=4-1
7 saigo=10
8 saigo> 1の間
9 iを1からjまで1ずつ増やしながら
10 もし Tokuten[i]< Tokuten[i+1]ならば
11     n = Name[i]
12     Name[i]=Name[i+1]
13     Name[i+1]=n
14     t=Tokuten[i]
15     Tokuten[i]=Tokuten[i+1]
16     Tokuten[i+1]=t
17     saigo=i
18     を実行する
19     を繰り返す
20     j=saigo-1
21     を繰り返す
22
23 iを1から4まで1ずつ増やしながら
24 str=Name[j]','+',Tokuten[j]
25 strを表示する
26 strを表示する
27 を繰り返す
```

あきよ: 81  
なつお: 77  
はるこ: 73  
ふゆき: 68

## wPEN

- 短冊プログラミング
- 作問・解答機能

The screenshot displays the wPEN programming environment. The main window shows a code editor with the following code:

```
1 変数 year, age  
2 year ← input()  
3 age ← year-1964  
4 もし year-1964>11 ならば  
5 「お祝い」 を表示する  
6 を実行する
```

The console window shows the output:

```
2021  
お祝い
```

The settings panel on the right shows the following options:

- 変数
- ←
- ← input()
- を表示する
- を修正なしで表示する
- もし
- もし ~ そうでなければ
- もし ~ そうでなくもし
- ~ の間、繰り返し
- 繰り返し、~ になるまで実行する
- 繰り返しながら繰り返し
- 減らしながら繰り返し

## XTetra

- ビジュアル・タイピング両対応
- プロファイル機能
- JavaScript への変換

標準出力(実行あり) 標準出力(実行なし) 標準入力 標準入力メッセージあり 代入 条件分岐文(IF文) 条件分岐文(IF-else文) 条件分岐文(IF-else-if文) 順次繰返し文(for文) 順次繰返し文(for-in文) 逐次繰返し文(while文) 逐次繰返し文(while文) 逐次繰返し文(repeat-until文) 関数・子関数 関数(戻り値あり) 呼び出し 配列

```

実行 ステップ 停止 [Alt] 実行 JavaScript 変換
1 // 乱数ソート
2 A ← {17, 32, 18, 41, 52, 20, 87, 12, 59, 69}
3 初期値(A) * と A を表示する
4 ソート欄 * と A を表示する
5 ← 標準関数(A)
6 8 番号から昇 * 3 まで 1 ずつ減らしながら、
7   | 番号 1 ~ 1 から N まで 1 ずつ減らしながら、
8   | 必ず A[番号] > A[] ならば、
9   |   | 番号 - 1
10  |   | 数値交換する
11  |   | を繰り返す
12  |   | copy ← A[番号]
13  |   | A[番号] ← A[]
14  |   | A[] ← copy
15  |   | を繰り返す
16 ソート欄 * と A を表示する
17
18

```

ビューワー

標準  
グローバル変数

ステップ	グローバル変数
0	10
1	10
2	10
3	10
4	10
5	10
6	10
7	10
8	10
9	10
10	10
11	10
12	10
13	10
14	10
15	10
16	10
17	10
18	87

ローカル変数  
実行回数

A	10
N	10
I	3
NO	3
J	10
copy	87

コンソール 実行停止

ソート前 {17, 32, 18, 41, 52, 20, 87, 12, 59, 69}

ソート後 {12, 17, 18, 20, 32, 41, 52, 59, 69, 87}

# WaPEN

- コード ↔ フローチャート
- サーバ不要（ローカルで実行可能）

The screenshot displays the WaPEN web application interface, which is used for converting code to flowcharts and vice versa. The interface is divided into several sections:

- Navigation and Settings:** Includes buttons for '新規' (New), '実行' (Execute), 'ステップ実行' (Step Execute), 'リセット' (Reset), '実行履歴' (Execution History), 'フローチャート' (Flowchart), 'コード<->フローチャート' (Code <-> Flowchart), and 'URL生成' (URL Generation). There are also 'Load', 'Save', and 'ファイル名:' (Filename) options.
- Code Editor:** A text area containing the following code:
 

```

1 a=1
2 bを1から100まで1ずつ増やしながら、
3 | a←a*b
4 | bと「!」とaを表示する
5 | を繰り返す
      
```
- Output:** A text area showing the execution results:
 

```

3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
11!=39916800
12!=479001600
13!=6227808000
14!=-87178291200
15!=-1307674368000
16!=-20922789888000
17!=-3556874208960000
18!=-64023757057280000
実行時エラーです
3行目:整数で表される範囲
を越えました
      
```
- Flowchart:** A flowchart diagram on the right side of the interface. It starts with 'はじめ' (Start), followed by 'a←1', then a loop 'b←1から100まで1ずつ増やす' (Increase b from 1 to 100 by 1). This is followed by 'a←a\*b', a decision diamond 'bと「!」とa', and finally 'おわり' (End).
- File Management:** Buttons for 'アップロード' (Upload), 'ダウンロード' (Download), '削除' (Delete), and '全削除' (Delete All) are located at the top right.
- Input/Output Controls:** A row of buttons for '入力 (整数)' (Input Integer), '入力 (実数)' (Input Real), '入力 (文字列)' (Input String), '入力 (数値)' (Input Numeric), '出力' (Output), '出力 (改行なし)' (Output No Line Break), and '代入' (Assignment). Below these are buttons for 'もし' (If), 'もしも〜そうでなければ' (If not), and a range selection button '〜の間, 繰り返す' (Between, Repeat).
- Navigation and Help:** A row of buttons labeled 'サンプル1' through 'サンプル8' (Sample 1 to 8) and a 'マニュアル' (Manual) link.

## 新 DNCL に近づけた版も...

- どんくり → 新 DNCL 対応版
- wPEN → 新 DNCL 対応版
- XTetra → つちのこ
- WaPEN → PyPEN

新規 実行 ステップ実行 リセット 実行結果

フローチャート  コード→フローチャート  コード→Python  URL生成

Load Save ファイル名: \_\_\_\_\_

問題選択  視点

+ 0 -

```

1 a=1
2 bを1から100まで1ずつ増やしながら：
3 ... a=a*b
4 ... 表示する(bと"1"とa)
5
6

```

```

21=2
31=6
41=24
51=120
61=720
71=5040
81=40320
91=362880
101=3628800
111=39916800
121=479001600
131=6277920000
141=87178291200
151=1307674368000
161=2092789888000
171=355687428096000
181=6402373785728000
実行時エラーです
3行目: 整数で表される範囲を超えました

```

入力 (整数) 入力 (実数) 入力 (文字列) 入力 (真偽) 出力 改行無出力

代入 += -= \*= /% //% %& | % ^

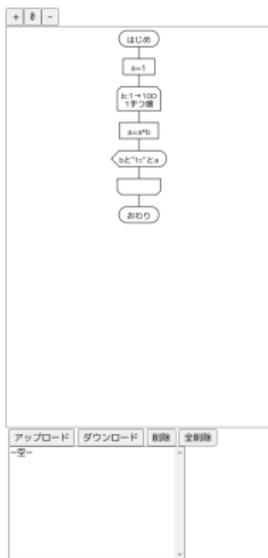
もし もし〜そうでなければ

〜の間 進みながら 進らしながら 配列の要素について 繰り返しを抜ける

配列に追加 配列に連結 戻数 手続き 値を返す 手続きを抜ける

サンプル1 サンプル2 サンプル3 サンプル4 サンプル5 サンプル6 サンプル7 サンプル8

[マニュアル](#)



## 3.2 センター試験等の問題で省略された部分

配列の初期化部分は問題に掲載されないことが多い（表1参照）

「どんくり」は2008～2018年度問題をサンプルに採用

実行	保存	読み込み	サンプル一覧
DNCL	英語表示		センター試験過去問題
1	Namae[1] ← 「はるこ」		2018年度 第3問 問2
2	Namae[2] ← 「なつお」		2018年度 第3問 問3
3	Namae[3] ← 「あきよ」		2017年度 第3問 問2
4	Namae[4] ← 「ふゆき」		2017年度 第3問 問3
5			2017年度 第3問 問2
6	j ← 4-1		2016年度 第3問 問2
7	saigo ← 10		2016年度 第3問 問3
8	saigo > 1の間、		2016年度 第3問 問2
9	iを1からjまで1ずつ増やして		2016年度 第3問 問3
10	もしTokuten[i] < Namae[i]		2015年度 第3問 問2
11	なら Tokuten[i] ← Namae[i]		2015年度 第3問 問3
12	Namae[i] ← Tokuten[i]		2014年度 第3問 問2
13	Namae[i+1] ← Tokuten[i]		2014年度 第3問 問3
14	t ← Tokuten[i]		2014年度 第3問 問2
15	Tokuten[i] ← Tokuten[i+1]		2014年度 第3問 問3
16	Tokuten[i+1] ← t		2014年度 第3問 問2
17	saigo ← i		2014年度 第3問 問3
18	を繰り返す		2013年度 第3問 問2
19	を繰り返す		2013年度 第3問 問3
20	j ← saigo-1		2012年度 第3問 問2
21	を繰り返す		2012年度 第3問 問3
22			2012年度 第3問 問2
23			2012年度 第3問 問3
24	iを1から4まで1ずつ増やして		2011年度 第3問 問1
25	str ← Namae[i]+Tokuten[i]		2011年度 第3問 問2
26	strを表示する		2011年度 第3問 問3
27	を繰り返す		2010年度 第3問 問2
			2010年度 第3問 問3

## ① はじめに

## ② DNCL の概要

## ③ 関連研究

- DNCL 学習環境
- センター試験等の問題で省略された部分

## ④ センター試験等のプログラミング問題における省略部分および WaPEN と PyPEN の追加実装

- PyPEN から WaPEN への移植
- 長さの決まっている配列
- 長さの決まっていない配列
- 2次元配列
- Web ストレージを用いた File I/O (もどき)

## ⑤ おわりに

## 4.1 PyPEN から WaPEN への移植

最近 PyPEN にかかりきり

→ WaPEN の開発が停滞

PyPEN を WaPEN に移植（構文解析などを除く）

→ その結果

- 両方での新機能実装が容易に
- 互換性の放棄
  - 変数宣言の廃止 → 本来の DNCL には近づいた

## 4.2 長さの決まっている配列

PEN...配列への代入が1つずつ

PenFlowchart...まとめたの代入をサポート

### PEN

```
te[0] ← 「グー」  
te[1] ← 「チョキ」  
te[2] ← 「パー」
```

### PenFlowchart

```
te ← { 「グー」 , 「チョキ」 , 「パー」 }
```

## WaPEN での書き方 1

```
te ← [「グー」, 「チョキ」, 「パー」]
```

## WaPEN での書き方 2

```
te ← []
```

```
te に「グー」を追加する
```

```
te に「チョキ」を追加する
```

```
te に「パー」を追加する
```

## さらに...

```
a ← [0,0,0,0,0,0,0,0,0,0]
```

```
b ← 10 個の 0
```

```
c ← [0]*10
```

## 細かい違い :

```
1 a ← 10個のrandom(5)
2 b ← [random(5)]*10
3 変数を確認する
```

```
*** 変数確認 ***
```

```
a:[5,1,5,1,3,5,3,4,2,0]
```

```
b:[5,5,5,5,5,5,5,5,5,5]
```

```
---
```

## 4.3 長さの決まっていない配列

### 未実装

PyPEN...Python のコードに「直訳」したいので実装しない  
WaPEN...実装したい

「どんくり」が実装の参考になりそう

The screenshot shows a programming environment interface. At the top, there are buttons for '実行' (Execute), '保存' (Save), and '読み込み' (Load), along with a dropdown menu set to '2011年度 第3回 問3'. Below this, there are tabs for 'DNCL' and '英語表示'. A list of instructions is displayed:

- 1 aのすべての値を3にする
- 2 a[1]を表示する
- 3 a[2] ← 4
- 4 aを表示する

To the right of the instructions is a code editor window containing the following code:

```
3
{ , 4 }
```

## 4.4 2次元配列

### 2018年度問題の初期化部分

#### Tetra 版

複数行での初期化が可能

```

/*2018年 情報関係基礎 第3問 問2*/
tate←9,yoko←11
Masu←{{1,1,1,1,1,1,1,1,1},
{1,0,0,0,1,0,0,0,1},
{1,1,1,0,0,0,1,0,1},
{9,0,0,0,1,0,1,0,1},
{1,1,1,0,1,0,1,0,1},
{1,0,1,1,1,0,1,1,1},
{1,0,0,0,1,0,1,0,1},
{1,1,1,0,0,0,1,0,9},
{1,0,0,0,1,1,1,0,1},
{1,0,1,0,0,0,0,0,1},
{1,1,1,1,1,1,1,1,1}}

```

#### (当時の) WaPEN 版

複数行での初期化が**不可能**

1	整数 tate,yoko,Masu[11,9],x,y,nutta,s
2	tate ← 9
3	yoko ← 11
4	Masu[1] ← [0,1,1,1,1,1,1,1,1]
5	Masu[2] ← [0,1,0,0,0,1,0,0,0]
6	Masu[3] ← [0,1,1,1,0,0,0,1,0]
7	Masu[4] ← [0,9,0,0,0,1,0,1,0]
8	Masu[5] ← [0,1,1,1,0,1,0,1,0]
9	Masu[6] ← [0,1,0,1,1,1,0,1,1]
10	Masu[7] ← [0,1,0,0,0,1,0,1,0]
11	Masu[8] ← [0,1,1,1,0,0,0,1,0]
12	Masu[9] ← [0,1,0,0,0,1,1,1,0]
13	Masu[10] ← [0,1,0,1,0,0,0,0,1]
14	Masu[11] ← [0,1,1,1,1,1,1,1,1]

# コマ後などの改行を許すよう拡張

## 当時の WaPEN 版

```

1  整数  tate,yoko,Masu[11,9],x,y,nutta,s
2  tate ← 9
3  yoko ← 11
4  Masu[1] ← [0,1,1,1,1,1,1,1,1,1]
5  Masu[2] ← [0,1,0,0,0,1,0,0,0,1]
6  Masu[3] ← [0,1,1,1,0,0,0,1,0,1]
7  Masu[4] ← [0,9,0,0,0,1,0,1,0,1]
8  Masu[5] ← [0,1,1,1,0,1,0,1,0,1]
9  Masu[6] ← [0,1,0,1,1,1,0,1,1,1]
10 Masu[7] ← [0,1,0,0,0,1,0,1,0,1]
11 Masu[8] ← [0,1,1,1,0,0,0,1,0,9]
12 Masu[9] ← [0,1,0,0,0,1,1,1,0,1]
13 Masu[10] ← [0,1,0,1,0,0,0,0,0,1]
14 Masu[11] ← [0,1,1,1,1,1,1,1,1,1]

```

## 今の WaPEN 版

```

1  tate←9
2  yoko←11
3  Masu←[
4  [0,0,0,0,0,0,0,0,0,0],
5  [0,1,1,1,1,1,1,1,1,1],
6  [0,1,0,0,0,1,0,0,0,1],
7  [0,1,1,1,0,0,0,1,0,1],
8  [0,9,0,0,0,1,0,1,0,1],
9  [0,1,1,1,0,1,0,1,0,1],
10 [0,1,0,1,1,1,0,1,1,1],
11 [0,1,0,0,0,1,0,1,0,1],
12 [0,1,1,1,0,0,0,1,0,9],
13 [0,1,0,0,0,1,1,1,0,1],
14 [0,1,0,1,0,0,0,0,0,1],
15 [0,1,1,1,1,1,1,1,1,1]]

```

## 4.5 Web ストレージを用いた File I/O もどき

PEN, PenFlowchart...File I/O アリ

→ Web アプリケーションへの移植は？

- ローカルファイルは扱えない
- WaPEN や PyPEN はサーバを絡ませたくない

## 試作問題第 5 問

- (01) Angoubun = ["p", "y", "e", "b", ..., (省略) ..., "k", "b", "d", "r", "."]
- (02) 配列 Hindo のすべての要素に 0 を代入する
- (03) i を 0 から 要素数 (Angoubun) - 1 まで 1 ずつ増やしながらか:
- (04) |    bangou = 差分(  )           ケ ①Angoubun[i]
- (05) |    もし bangou != -1 ならば:
- (06) |    |    =  + 1       コ ④Hindo[bangou]
- (07) 表示する (Hindo)

## (省略) は千数百文字

```

pyeb cmybo knx cofox iokbc kqy yeb pkdrobc lbyeqrđ pybđr yx đrsc myxđsxoxđ, k
xog xkđsyx, myxmosfon sx vslobđi, knx nonsmkđon dy dro zbyzycsđsyx đrkd kvv
wox kbo mbokđon oaekv.
xyg go kbo oxqkqon sx k qbokđ msfsv gkb, docđsxq grodrob đrkd xkđsyx, yb kxi
xkđsyx cy myxmosfon knx cy nonsmkđon, mxv vyxq oxnebo. go kbo wod yx k qbokđ
lkđđvo-psovn yp đrkd gkb. go rkfo mywo dy nonsmkđo k zybđsyx yp đrkd psovň, k
c k psxkv bođđsxq zvkmo pyb đryco gry robo qkfo đrosb vsfoc đrkd đrkd xkđsyx
wsqrđ vsfo. đđ sc kvdyqodrob psđđsxq knx zbyzob đrkd go cryevň ny đrsc.
led, sx k vkbqob coxco, go mxv xyđ nonsmkđo -- go mxv xyđ myxcombkđo -- go mk
x xyđ rkvvvg -- đrsc qbyexň. dro lbkfo wox, vsfsxq knx nokň, gry cđbeqqvň ro
bo, rkfo myxcombkđon đđ, pkb klyfo yeb zyyb zyqob dy knň yb nodbkmd. dro gybv
ň gsvv vsđđvo xyđo, xyb vyxq bowowlob grkd go cki robo, led đđ mxv xofob pybq
od grkd dro nsn robo. đđ sc pyb ec dro vsfsxq, bkđrob, dy lo nonsmkđon robo
dy dro expsxscron gybu grsmr droi gry pyeqrđ robo rkfo đrec pkb cy xylvi knfk
xmon. đđ sc bkđrob pyb ec dy lo robo nonsmkđon dy dro qbokđ đkuo bowksxđsyx lo
pybo ec -- đrkd pbyw đroco ryxybon nokň go dkuo sxmbokcon nofyđsyx dy đrkd mk
eco pyb grsmr droi qkfo dro vkcd pevň wokcebo yp nofyđsyx -- đrkd go robo rsq
rvi bocyvfo đrkd đroco nokň crkvv xyđ rkfo nson sx fksx -- đrkd đrsc xkđsyx,
exnob qyn, crkvv rkfo k xog lsbđr yp pboonyw -- knx đrkd qyfobxwođ yp dro zo
vzvo. li dro zovzvo. nvb dro zovzvo. crkvv xyđ zobscr pbyw dro okbđr.

```

## 省略しないで書くと...

```

+ 0 -
1 Angobun=["p","y","e","b","c","m","y","b","o",
2 "k","x","n","c","o","f","o","x","i","o",
3 "k","b","c","k","q","y","y","e","b","p",
4 "k","d","r","o","b","c","l","b","y","e","q","r",
5 "d","p","y","b","d","r","y","x","d","r",
6 "s","c","n","y","x","d","s","x","o","x","d",
7 "k","x","o","g","x","k","d","s","y","x",
8 "m","y","x","m","o","s","f","o","n","s","x",
9 "v","s","l","o","b","d","i","k","x","n","n",
10 "o","n","s","m","k","d","o","n","d","y","d",
11 "r","o","z","b","y","z","y","c","s","d","s","y",
12 "x","d","r","k","d","k","v","v","w","o",
13 "x","k","b","o","m","b","o","k","d","o","n",
14 "o","a","e","k","v","x","y","g","g","o",
15 "k","b","o","o","x","q","k","q","o","n",
16 "s","x","k","q","b","o","k","d","m","s",
17 "f","s","v","g","k","b","d","o","c","d","s",

```

~ (略) ~

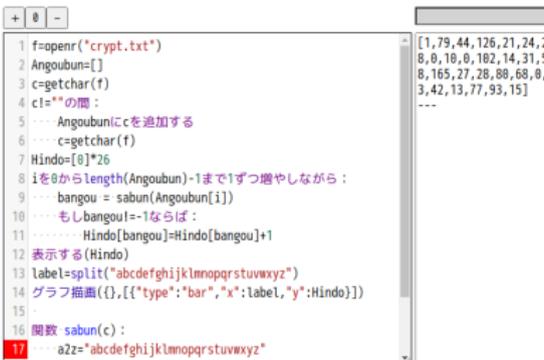
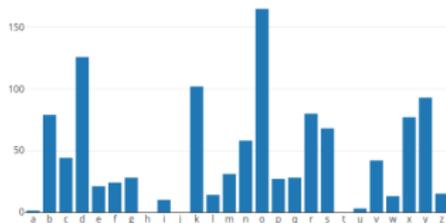
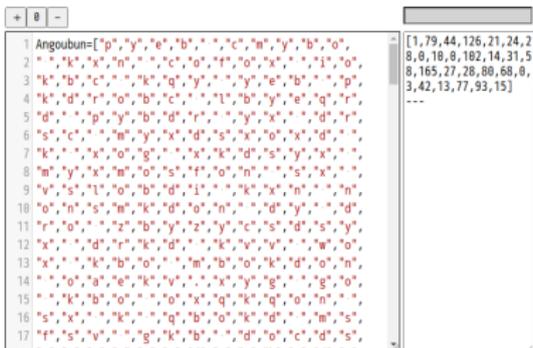
```

+ 0 -
109 "p","y","b","d","r","o","z","o","y","z",
110 "v","o","c","r","k","v","y","x","y","d",
111 "z","o","b","s","c","r","p","b","y","w","d",
112 "r","o","o","k","b","d","r","."]
113 Hindo = 26個の0
114 iを0からlength(Angobun)-1まで1ずつ増やしながら：
115 ---- bangou = sabun(Angobun[i])
116 ---- もしbangou != -1ならば：
117 ---- ---- Hindo[bangou] = Hindo[bangou] + 1
118 表示する(Hindo)
119 関数 sabun(c)：
120 ---- aZz="abcdefghijklmnopqrstuvwxyZ"
121 ---- iを0から25まで1ずつ増やしながら：
122 ---- ---- もしc==aZz[i]ならば：
123 ---- ---- ---- iを返す
124 ---- ---- -1を返す
125

```

外部ファイルから読み込ませたい

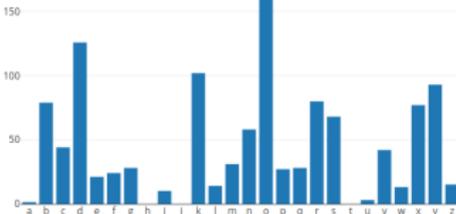
→ Web ストレージを用いた File I/O (もどき) を実装



アップロード   ダウンロード   削除   全削除

gettysburg.txt

crypt.txt



## ① はじめに

## ② DNCL の概要

## ③ 関連研究

- DNCL 学習環境
- センター試験等の問題で省略された部分

## ④ センター試験等のプログラミング問題における省略部分および WaPEN と PyPEN の追加実装

- PyPEN から WaPEN への移植
- 長さの決まっている配列
- 長さの決まっていない配列
- 2次元配列
- Web ストレージを用いた File I/O (もどき)

## ⑤ おわりに

## 5. おわりに

WaPEN, PyPEN に追加実装

→省略されてきた初期化部分への対応

「どんくり」や Tetra に追いつこうとしただけではあるが

## 高野ら [23]

情報系の標準的な常識からすると、わざわざ曖昧さを残しておくのは不自然である。ここからは著者らの推測の範囲に過ぎないが、ひとつ考えられる理由として、DNCLの設計者は、DNCLを独立したプログラミング言語として学習することを嫌がっていると考えられる。つまり、プログラミング教育は、世の中に実在するプログラミング言語で実習することを望み、(略)

## 水野 [24]

少し懸念していることは、DNCLだけを使ってプログラミングの授業が行われることがないように、ということです。やはり授業では、教科書で使われている実用的なプログラム言語を使っていたきたい。

## DNCL だからダメ？

ダメなのは DNCL にとどまること

- 導入の一方法として
- わからなくなったときに振り返る先として

ということなら意味はあるのではないか

## Python で書き方に迷ったときのアンチョコになればいい

```

+ 0 -
1 Angoubun=[]
2 f=openr('crypt.txt')
3 c=getchar(f)
4 c!=""の間：
5 ... Angoubunにcを追加する
6 ... c=getchar(f)
7 close(f)
8 Hirabun=""
9 hukugousuu=26-10
10 iを0からlength(Angoubun)-1まで1ずつ増やしながら：
11 ... bangou=sabun(Angoubun[i])
12 ... もしbangou!=-1ならば：
13 ... Hirabun+=moji((bangou+hukugousuu)%26)
14 ... そうでなければ：
15 ... Hirabun+=Angoubun[i]
16 表示する(Hirabun)
17 .

```

```

def sabun(c):
    a2z = 'abcdefghijklmnopqrstuvwxyz'
    for i in range(0,25+1,1):
        if c==a2z[i]:
            return i
    return -1
def moji(n):
    a2z = 'abcdefghijklmnopqrstuvwxyz'
    if 0<=n<=25:
        return a2z[n]
    else:
        return 'アルファベットでない'
Angoubun = []
f = open('crypt.txt','r')
c = f.read(1)
while c!="":
    Angoubun.append(c)
    c = f.read(1)
f.close()
Hirabun = ''
hukugousuu = 26 - 10

```