愛知県立三谷水産高等学校　教諭　山田　学

海洋情報技術〔プログラミング〕　指導書

Basic の基本処理と大学入学用プログラミング言語の対応

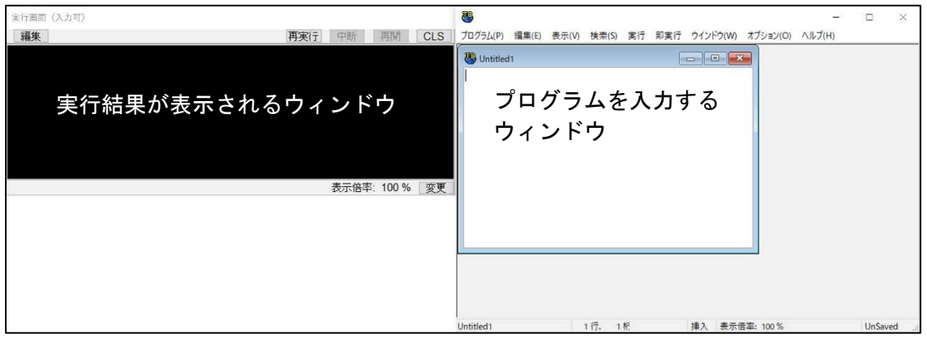
TinyBasic ソフトウェアのダウンロード先

https://tbasic.org/

ファイルをダウンロードし、展開したら、実行ファイル TBasic を生徒の実習用フォルダや USB メモリにコピーして使用する。

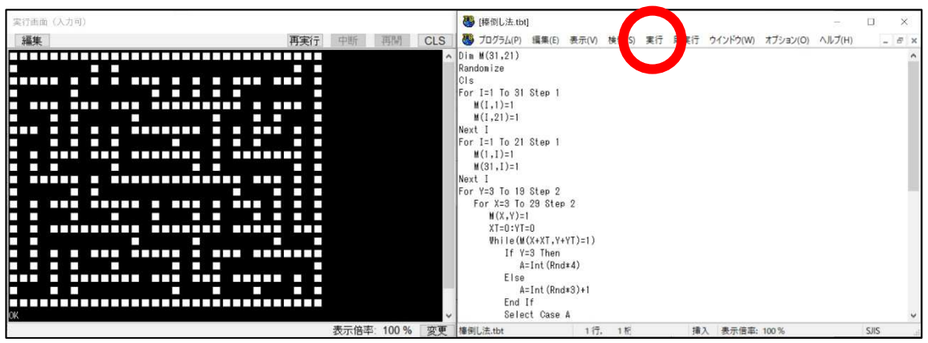
ソフトウェアの起動

TBasic をダブルクリックすると、ソフトウェアが起動する。



プログラムの入力と実行

プログラム入力ウィンドウにプログラムを入力して、
［実行］メニューから［プログラムの実行］を選ぶ、または［Ｆ９］キーを押すことで、
プログラムが実行される。



大学入学用プログラミング言語「DNCL」

高校で学習するプログラミングは、プログラム言語の習得が目的ではなく、プログラム的な思考力やアルゴリズムの理解が目的となります。このため、各学校で用いられるプログラム言語は、Python や、VBA、C 言語、BASIC など様々なものとなります。しかし、特定の言語に特化した出題をすると，他の言語でプログラミングを習った人が不利となるため、大学入試では、共通テスト用プログラム表記である、疑似コードを用います。このプログラム表記は，何か特定のプログラミング言語を習ったことがあれば，容易に習得できる表記法になっているため、どの言語でプログラミングを学んでも十分理解することができます。

出題例　大学入学共通テスト「情報Ｉ」試作問題より

```
(1) kakaku = 46
(2) min_maisu = 100
(3) [サ] を [シ] から 99 まで 1 ずつ増やしながら繰り返す：
(4)     shiharai = kakaku + tsuri
(5)     maisu = [ス] + [セ]
(6)     もし [ソ] < min_maisu ならば：
(7)         [タ] = [ソ]
(8) 表示する(min_maisu)
```

愛知県立三谷水産高等学校　教諭　山田　学

海洋情報技術〔プログラミング〕

①代入と表示

Basic プログラム

```
Kingaku = 1200
Print Kingaku
```

疑似コード

```
Kingaku = 1200
表示する(Kingaku)
```

※プログラムは、変数 Kingaku に 1200 を代入し、表示しています。
Basic プログラムと、疑似コードを見比べてみましょう。

②計算

Basic プログラム

```
A = 10
B = 3
Print ”加算結果”;A + B
```

疑似コード

```
A = 10
B = 3
表示する(”加算結果”,A + B)
```

※変数 A と変数 B を加算した結果を表示しています。
疑似コードの演算子は、四則演算（＋－＊／）　正数の除算（÷）　剰余（%）べき乗（＊＊）となります。

```
A + B は 13
A - B は 7
A * B は 30
A / B は 3.333333
A ÷ B は 3                ※ Basic では、Int(A/B)
A % B は 1                ※ Basic では、A Mod B
A ** B は 1000            ※ Basic では、A^B          となります。
```

③判断

Basic プログラム

```
Tensu = 27
If Tensu < 29 Then
  Print ”赤点”
Else
  Print ”合格”
End if
```

疑似コード

```
Tensu = 27
もし Tensu < 29 ならば：
  表示する(”赤点”)
そうでなければ：
 └ 表示する(”合格”)
```

※Tensu が 29 点未満のとき"赤点"と表示し、それ以外のとき"合格"と表示するプログラムである。
疑似コードでは、End If に該当する部分が　└　でまとめてある点を注意する。

④繰り返し(For)

Basic プログラム

```
For I=1 To 10 Step 1
  Print I
Next
```

疑似コード

```
Iを1から10まで1ずつ増やしながら繰り返す：
 └ 表示する( I )
```

※１～10 までの数値を表示するプログラムである。

問題１

問題１　次の Basic プログラムを疑似コードで書き直しなさい。

```
Teihen = 20
Takasa = 10
print Teihen * Takasa / 2
```

問題２　次の Basic プログラムを疑似コードで書き直しなさい。

```
A = 47
B = 12
Print "余りは";A%B;"です。"
```

問題３　次の Basic プログラムを疑似コードで書き直しなさい。

```
Kin = 8500
If Kin <=10000 Then
  Print "お金が足りません。"
End If
```

問題４　次の Basic プログラムを疑似コードで書き直しなさい。

```
For A=1 To 100 Step 2
 Print A
Next
```

問題５　次の Basic プログラムを疑似コードで書き直しなさい。

```
ST = 20
ED = 35
Gokei = 0
For I=ST To ED Step 1
  Gokei = Gokei + I
Next
Print ST;"～";ED;"までの合計は";Gokei
```

問題２

問題１　次の疑似コードを Basic プログラムに書き直して実行させなさい。
　　※２つの数を入れ替える。

```
Kazu1 = 50
Kazu2 = 30
A = Kazu1
Kazu1 = Kazu2
Kazu2 = A
表示する（"Kazu1 = ",Kazu1）
表示する（"Kazu2 = ",Kazu2）
```

問題２　次の疑似コードを Basic プログラムに書き直して実行させなさい。
　　※大きい数を表示する。

```
Kazu1 = 63
Kazu2 = 71
もし Kazu1>=Kazu2 ならば：
  表示する（"大きい数は",Kazu1,"です。"）
そうでなければ：
└ 表示する（"大きい数は",Kazu2,"です。"）
```

問題３　次の疑似コードを Basic プログラムに書き直して実行させなさい。
　　※１～Kazu までの合計を求める。

```
Kazu = 30
Gokei = 0
I を１から Kazu まで１ずつ増やしながら繰り返す：
└  Gokei = Gokei + I
表示する（"合計：",Gokei）
```


⑥配列（一次元）

Basic プログラム

```
Dim Kazu(10)
For I=0 To 9 Step 1
  Read Kazu(I)
Next
Data 5,7,2,4,9,6,1,0,3,8
```

疑似コード

```
Kazu = [5,7,2,4,9,6,1,0,3,8]
```

※Kazu[0] には 5 が格納されている。
※Kazu[9] には 8 が格納されている。

※Basic 言語では、配列を使用するときは、最初に Dim で宣言しなければいけません。
　また、配列に値を格納する際には、繰り返し命令等を使用して、順番に格納する必要があります。
※疑似コードでは、あらかじめ配列に値を格納しておく際、右のように記述します。
　配列の添字は０から始まるため、Basic プログラムもそれに合わせています。

例）配列に格納されている値の中から、最大値を表示する。

Basic プログラム

```
Dim Kazu(10)
For I=0 To 9 Step 1
  Read Kazu(I)
Next
Data 5,7,2,4,9,6,1,0,3,8

Max = 0
For I=0 To 9 Step 1
  If Max<=Kazu(I) Then
    Max = Kazu(I)
  End If
Next
Print ”最大値：”;Max
```

疑似コード

```
Kazu = [5,7,2,4,9,6,1,0,3,8]

I を 0 から 9 まで 1 ずつ増やしながら繰り返す：
｜もし Max<=Kazu[I] ならば：
｜｜ Max = Kazu[I]
表示する（”最大値：”,Max）
```

問題

問題１　配列中の、最小値を表示しなさい。
　　Basic プログラムで作成し、疑似コードも記述しなさい。

問題２　配列中の値の、合計を表示しなさい。
　　Basic プログラムで作成し、疑似コードも記述しなさい。

問題３　配列中の値の、平均値を表示しなさい。
　　Basic プログラムで作成し、疑似コードも記述しなさい。

⑦乱数

Basic プログラム

```
Randomize
A = Int(Rnd*6)+1
Print A
```

疑似コード

```
A = 乱数(1,6)
表示する (A)
```

※Basic 言語で乱数を使用するときは、Rnd を使用します。Randomize で乱数を初期化します。
　Rnd は０以上１未満の乱数が発生するため、
　６を掛けて少数以下を切り捨て、最後に＋１することでサイコロを作ることができます。
※疑似コードでは、乱数(m,n)と記述します。ｍ～ｎまでの乱数を発生させます。
※乱数はシミュレーションを行うときなどの値としてよく用いられます。

例）コイン投げをシミュレーションします。
　　コインの表を１、コインの裏を２と決めます。
　　１～２の乱数を１００回発生させ、表と裏の回数を数えます。

Basic プログラム

```
Randomize
Su1 = 0
Su2 = 0
For I=1 To 100 Step 1
  R = Int(Rnd*2)+1
  If R=1 Then
    Su1 = Su1 + 1
  Else
    Su2 = Su2 + 1
  End If
Next
Print ”表：”;Su1
Print ”裏：”;Su2
```

疑似コード

```
Su1 = 0
Su2 = 0
Iを1から100まで1ずつ増やしながら繰り返す：
 | R = 乱数(1,2)
 | もし R=1 ならば：
 |    Su1 = Su1 + 1
 | そうでなければ：
 |  Su2 = Su2 + 1
表示する(”表：”,Su1)
表示する(”裏：”,Su2)
```

問題

問題１　コインを３００回投げた時の、表と裏の回数と割合を表示しなさい。
　　Basic プログラムで作成し、疑似コードも記述しなさい。

問題２　じゃんけんの出し手を、グーを１、チョキを２、パーを３として
　　５０回シミュレーションし、それぞれの出し手の回数と割合を表示しなさい。
　　Basic プログラムで作成し、疑似コードも記述しなさい。

⑧繰り返し(While)

Basic プログラム

```
I = 1
While(I<10)
  I = I + 1
WEnd
Print I
```

疑似コード

```
I = 1
I<10 の間繰り返す：
 I = I + 1
表示する(I)
```

※I の初期値を１として、10 回繰り返しています。　For I=1 To 10 Step 1 と同じ処理
　While()は条件を満たしている間繰り返されます。
　繰り返し回数が分からない場合に用いられる命令です。

例）１～順番に加算していき、100 を超えたら終了します。終了時、Gokei と I の値を表示します。

Basic プログラム

```
I = 1
Gokei = 0
While(Gokei<=100)
  Gokei = Gokei + I
  I = I + 1
WEnd
Print Gokei , I
```

疑似コード

```
I = 1
Gokei = 0
Gokei<=100 の間繰り返す：
 | Gokei = Gokei + I
 | I = I + 1
表示する(Gokei , I)
```

例）配列 Ten[]に何人分かのテストの点数(0～100)が記録されており、合計点と平均点表示する。
　　なお、配列 Ten[]の最後には、終わりを示す、999 が格納されているものとする。

Basic プログラム

```
Dim Ten(100)
For I=0 To 5
  Read Ten(I)
Next
Data 78,92,27,45,60,81,999
-------------------------------
I = 0
Gokei = 0
While(Ten(I)<>999)
  Gokei = Gokei + Ten(I)
  I = I + 1
WEnd
Print ”合計：”;Gokei
Print ”平均：”;Gokei/I
```

疑似コード

```
Ten = [78,92,27,45,60,81,999]
-------------------------------
I = 0
Gokei = 0
Ten[I]!=999 の間繰り返す：
 | Gokei = Gokei + Ten[I]
 | I = I + 1

表示する(”合計：”,Gokei)
表示する(”平均：”,Gokei/I)
```

※点線より上の部分で、配列 Ten[]に点数を格納しています。
　６人分の点数を格納していますが、人数は分からないものとしてプログラムを組みます。

⑨関数

関数は、値を渡すことによって、計算や処理を行い、結果を返してくれるプログラムのまとまりです。

メインプログラム(Basic)

```
Ten = 27
Kekka = Hantei(Ten)
If Kekka=1 Then
  Print "赤点"
Else
  Print "合格"
End If
```

関数　※テストの点数を受け取って、赤点なら１，合格なら０を返す関数

```
Function Hantei(Tensu)
  Kekka = 0
  If Tensu<30 Then
    Kekka = 1
  End If
  Hantei = Kekka
End Function
```

※疑似コードでも同様に関数を使用することができます。
あらかじめ用意されている関数を使用するときは、メインプログラム部分だけが示され、
関数を作成する問題の場合は、関数内のプログラムも示されます。
※関数に渡す値を**引数**といい、返される値を**戻り値**といいます。

疑似コード

```
Ten = 27
Kekka = 判定(Ten)
もし Kekka=1 ならば：
    表示する("赤点")
そうでなければ：
  └ 表示する("合格")
```

```
関数 判定(Tensu)
  | Kekka = 0
  | もし Tensu<30 ならば：
  |   └ Kekka = 1
  └ Kekka を返す
```

※関数があらかじめ用意され、関数内のプログラムが問われない場合は、□部分は示されないことがあります。

問題

問題１　引数で値を渡すと、１～引数までを足した結果が戻り値として返される関数(Kasan)を作成しなさい。
Basic プログラムで作成し、疑似コードも記述しなさい。

問題２　２つの値を引数で渡すと、最大公約数が戻り値として返される関数(Koyaku)を作成しなさい。最大公約数の求め方は、ユークリッドの互除法を使用する。
ユークリッドの互除法による処理は、「海洋情報技術」教科書を参照。
Basic プログラムで作成し、疑似コードも記述しなさい。

⑩配列（二次元）

Basic プログラム

```
Dim A(8,3)
For I=0 To 2
  For J=0 To 7
    Read A(J,I)
  Next
Next
Data 50,80,70,20,10,40,60,30
Data  1, 5, 4, 3, 8, 7, 2, 6
Data 18,19,20,21,22,23,24,25
```

疑似コード

```
A = [ [50,80,70,20,10,40,60,30] ,
      [ 1, 5, 4, 3, 8, 7, 2, 6] ,
      [18,19,20,21,22,23,24,25] ]
```

※横８・縦３の２次元配列を定義し、初期値を代入しています。

50	80	70	20	10	40	60	30
1	5	4	3	8	7	2	6
18	19	20	21	22	23	24	25

上記の続き

Basic プログラム

```
Print A(2,1)
Print A(3,2)
Print A(7,0)
```

疑似コード

```
表示する(A[2,1])
表示する(A[3,2])
表示する(A[7,0])
```

※今回、二次元配列の値は A(列,行) で参照しています。A(行,列)でプログラミングする場合もあります。
上記プログラムの表示結果は　4
21
30
となります。　※配列の添え字は 0～となります。

※配列の処理は、繰り返し命令 For を使うことで効率的に行うことができます。
全ての合計を求める。

Basic プログラム

```
Gokei = 0
For I=0 To 2
  For J=0 To 7
    Gokei = Gokei + A(J,I)
  Next
Next
Print "合計：";Gokei
```

疑似コード

```
Gokei = 0
Ｉを０から２まで１ずつ増やしながら繰り返す：
| Ｊを０から７まで１ずつ増やしながら繰り返す：
└└ Gokei = Gokei + A[J,I]
表示する("合計：",Gokei)
```

問題

問題１　各行ごとの小計も表示しなさい。
Basic プログラムで作成し、疑似コードも記述しなさい。

問題２　表中の最小値を表示するプログラムを作成しなさい。
Basic プログラムで作成し、疑似コードも記述しなさい。