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

令和７年度大学入学共通テスト『試作問題』を解説

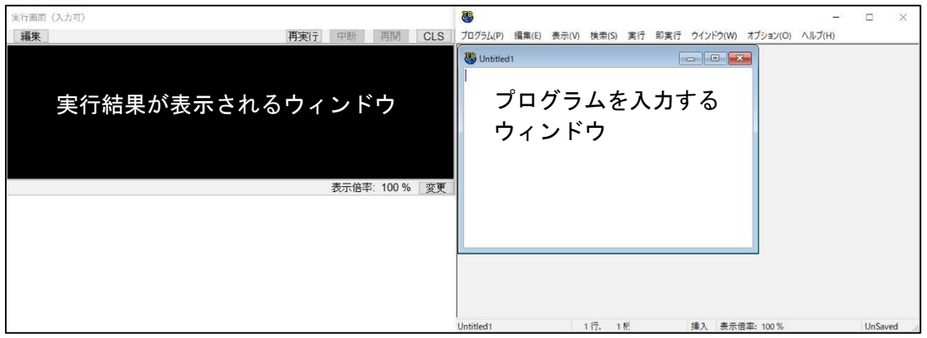
TinyBasic ソフトウェアのダウンロード先

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。



大学入試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。

海洋情報技術〔プログラミング〕

プログラミング問題　第３問

第３問　次の問い（問１～３）に答えよ。（配点　25）

問１　次の生徒（S）と先生（T）の会話文を読み，空欄［ア］に当てはまる数字をマークせよ。また，空欄［イ］～［エ］に入れるのに最も適当なものを，後の解答群のうちから一つずつ選べ。ただし，空欄［ウ］・［エ］は解答の順序は問わない。

S：この前，お客さんが 460 円の商品を買うのに，510 円を払って，釣り銭を 50 円受け取っていたのを見て，授業で勉強したプログラミングで，そんな「上手な払い方」を計算するプログラムを作ってみたいと思いました。

T：いいですね。まず，「上手な払い方」とは何かを考える必要がありますね。

S：普通は手持ちの硬貨の枚数を少なくするような払い方でしょうか。

T：そうですね。ただ，ここでは，客が支払う枚数と釣り銭を受け取る枚数の合計を最小にする払い方を考えてみませんか？ 客も店も十分な枚数の硬貨を持っていると仮定しましょう。また，計算を簡単にするために，100 円以下の買い物とし，使う硬貨は 1 円玉，5 円玉，10 円玉，50 円玉，100 円玉のみで 500 円玉は使わない場合を考えてみましょう。例えば，46 円をちょうど支払う場合，支払う枚数はどうなりますか？

S：46 円を支払うには，10 円玉 4 枚，5 円玉 1 枚，1 円玉 1 枚という 6 枚で払い方が最小の枚数になります。

T：そうですね。一方，同じ 46 円を支払うのに，51 円を支払って釣り銭 5 円を受け取る払い方では，支払いに 2 枚，釣り銭に 1 枚で，合計 3 枚の硬貨のやり取りになります。こうすると交換する硬貨の枚数の合計が最小になりますね。

S：これが上手な払い方ですね。

T：そうです。このように，客と店が交換する硬貨の合計が最小となる枚数，すなわち「最小交換硬貨枚数」の計算を考えましょう。

S：どうやって考えればいいかなぁ。

T：ここでは，次の関数のプログラムを作り，それを使う方法を考えてみまし

ょう。目標の金額を釣り銭無くちょうど支払うために必要な最小の硬貨枚数を求める関数です。

【関数の説明と例】

枚数(金額)… 引数として「金額」が与えられ，ちょうどその金額となる硬貨の組合せの中で，枚数が最小となる硬貨枚数が戻り値となる関数。
例：8 円は「5 円玉が 1 枚と 1 円玉が 3 枚」の組合せで最小の硬貨枚数になるので，**枚数**(8)の値は 4 となる。

T：これは，例えば，**枚数**(46)＝［ア］と計算してくれるような関数です。これを使って最小交換硬貨枚数の計算を考えてみましょう。例えば，46 円支払うのに，51 円払って 5 円の釣り銭を受け取る払い方をした場合，客と店の間で交換される硬貨枚数の合計は，この関数を使うと，どのように計算できますか？

S：［イ］で求められますね。

T：一般に，商品の価格 x 円に対して釣り銭 y 円を 0, 1, 2, . . . と変化させて，それぞれの場合に必要な硬貨の枚数の合計を

枚数(［ウ］)＋**枚数**(［エ］)

と計算し，一番小さな値を最小交換硬貨枚数とすればよいのです。

S：なるほど。それで，釣り銭 y はいくらまで調べればよいでしょうか？

T：面白い数学パズルですね。まあ，詳しくは今度考えるとして，今回は 100 円以下の商品なので y は 99 まで調べれば十分でしょう。

［イ］の解答群

⓪ **枚数**(51) ＋ **枚数**(5)　　① **枚数**(46) ＋ **枚数**(5)
② **枚数**(51) － **枚数**(5)　　③ **枚数**(46) － **枚数**(5)

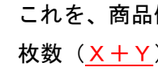
［ウ］・［エ］の解答群

⓪ x　　① y　　② $x + y$　　③ $x - y$

解説

お金を払う際の硬貨の枚数と、お釣りとして受け取る硬貨の枚数を最小にする方法を考えます。

46 円を支払う場合、  と支払えばちょうどとなりますが、

 を支払って、お釣り  をもらう方が、やり取りする枚数は少なくなります。

これを上手な払い方としています。

このプログラムを作成するために、金額を入力すると、枚数が最小となる硬貨枚数が返ってくる関数【枚数】を考えます。

枚数（金額）　とすると、硬貨の枚数が返されます。

枚数（1）　＝ 1 円玉 1 枚　＝ 1 枚
枚数（6）　＝ 5 円玉 1 枚　1 円玉 1 枚　＝ 2 枚
枚数（21）＝ 10 円玉 2 枚　1 円玉 1 枚　＝ 3 枚

では、枚数（46）＝［ア］は何枚になるでしょう。

答えは、上記の図と同様で 6 枚となります。

上手な払い方としては、51 円を支払って、5 円のお釣りをもらうため、
客と店間で交換される硬貨枚数の合計［イ］は、枚数(51)＋枚数(5)　となります。

これを、商品価格をＸ、お釣りをＹとしたとき、枚数（［ウ］）＋枚数（［エ］）は、
枚数（Ｘ＋Ｙ）＋枚数（Ｙ）となります。

46 円分の硬貨を払うよりも、お釣りＹ円をもらってでも、枚数が最小となることを考えます。

今回のプログラムでは 100 円以下の商品を想定しているため、枚数（Ｘ＋Ｙ）＋枚数（Ｙ）のＹを 0～99 まで変化させながら、最も枚数が少ないものを答えとします。

問２　次の文章の空欄［オ］～［コ］に入れるのに最も適当なものを，後の解答群のうちから一つずつ選べ。

Ｓ：まずは，関数「**枚数**(金額)」のプログラムを作るために，与えられた金額ちょうどになる最小の硬貨枚数を計算するプログラムを考えてみます。もう少しヒントが欲しいなぁ。

Ｔ：金額に対して，高額の硬貨から使うように考えて枚数と残金を計算していくとよいでしょう。また，金額に対して，ある額の硬貨が何枚まで使えて，残金がいくらになるかを計算するには，整数値の商を求める演算『÷』とその余りを求める演算『%』が使えるでしょう。例えば，46 円に対して 10 円玉が何枚まで使えるかは［オ］で，その際にいくら残るかは［カ］で求めることができますね。

Ｓ：なるほど！あとは自分でできそうです。

Ｓさんは，先生（Ｔ）との会話からヒントを得て，変数 **kingaku** に与えられた目標の金額（100 円以下）に対し，その金額ちょうどになる最小の硬貨枚数を計算するプログラムを考えてみた(図１)。ここでは例として目標の金額を 46 円としている。

配列 **Kouka** に硬貨の額を低い順に設定している。なお，配列の添字は **0** から始まるものとする。最低額の硬貨が 1 円玉なので **Kouka[0]**の値は **1** となる。

先生（Ｔ）のヒントに従い，高額の硬貨から何枚まで使えるかを計算する方針で，**(4)**～**(6)**行目のような繰返し文にした。この繰返しで，変数 **maisu** に支払いに使う硬貨の枚数の合計が計算され，変数 **nokori** に残りいくら支払えばよいか，という残金が計算される。

実行してみると［ア］が表示されたので，正しく計算できていることが分かる。いろいろな例で試してみたが，すべて正しく計算できていることを確認できた。


```
(1) Kouka = [1,5,10,50,100]
(2) kingaku = 46
(3) maisu = 0, nokori = kingaku
(4) i を［キ］ながら繰り返す：
(5)     maisu = ［ク］＋［ケ］
(6)     nokori = ［コ］
(7) 表示する(maisu)
```

図１　目標の金額ちょうどになる最小の硬貨枚数を計算するプログラム

［オ］・［カ］の解答群

⓪ 46 ÷ 10 ＋ 1　　① 46 % 10 － 1
② 46 ÷ 10　　③ 46 % 10

［キ］の解答群

⓪ **5** から **1** まで **1** ずつ減らし　　① **4** から **0** まで **1** ずつ減らし
② **0** から **4** まで **1** ずつ増やし　　③ **1** から **5** まで **1** ずつ増やし

［ク］の解答群

⓪ **1**　　① **maisu**　　② **i**　　③ **nokori**

［ケ］・［コ］の解答群

⓪ **nokori ÷ Kouka[i]**　　① **nokori % Kouka[i]**
② **maisu ÷ Kouka[i]**　　③ **maisu % Kouka[i]**

解説

実際に硬貨の枚数を計算するプログラムを考えます。
金額が 46 円のときの 10 円の枚数を考えるとき、

オ　10 円玉の枚数は、
46 ÷ 10 で ４枚

カ　残りは、
46 % 10 で ６円　となります。 ※演算子%は割った余り（剰余）となります。

プログラムを考えます。
硬貨の枚数を計算するときは、金額の大きい硬貨から行います。
（計算例）46 円の場合

１回目	46 円	÷	100 円硬貨	＝	０枚	残り 46 円
２回目	46 円	÷	50 円硬貨	＝	０枚	残り 46 円
３回目	46 円	÷	10 円硬貨	＝	４枚	残り 6 円
４回目	6 円	÷	5 円硬貨	＝	１枚	残り 1 円
５回目	1 円	÷	1 円硬貨	＝	１枚	残り 0 円

となります。

硬貨の金額は、あらかじめ配列に格納されています。
kouka = [1, 5, 10, 50, 100]
ここで注意したいのは、配列の添え字は 0, 1, 2, 3, 4 となることです。
よって、キは、 i を４から０まで１ずつ減らしながら繰り返す：となります。

繰り返し内の処理を、プログラム的に記述すると、

初期値　maisu = 0 , nokori = kingaku

１回目	nokori 円	÷ kouka[4]円硬貨	＝ ０枚	nokori = nokori % kouka[4]
２回目	nokori 円	÷ kouka[3]円硬貨	＝ ０枚	nokori = nokori % kouka[3]
３回目	nokori 円	÷ kouka[2]円硬貨	＝ ４枚	nokori = nokori % kouka[2]
４回目	nokori 円	÷ kouka[1]円硬貨	＝ １枚	nokori = nokori % kouka[1]
５回目	nokori 円	÷ kouka[0]円硬貨	＝ １枚	nokori = nokori % kouka[0]

となります。


```
(1) Kouka = [1,5,10,50,100]
(2) kingaku = 46
(3) maisu = 0, nokori = kingaku
(4) i を [ キ ] ながら繰り返す：
(5)     maisu = [ ク ] + [ ケ ]
(6)     nokori = [ コ ]
(7) 表示する(maisu)
```

繰り返し内の処理は、maisu に 各硬貨の枚数を加算していくので、
クは maisu 、ケは nokori÷kouka[i] となり、
コは nokori%kouka[i] となります。

この処理を、BASIC プログラムで記述してみましょう。

```
Cls
Dim Kouka(5)
For I=0 To 4 Step 1
  Read Kouka(I)
Next
Data 1,5,10,50,100
kingaku = 46
maisu = 0
nokori = kingaku
For I=4 To 0 Step -1
  maisu = maisu + Int(nokori/kouka(I))
  nokori = nokori Mod kouka(I)
Next
Print maisu
```

配列を宣言します。
硬貨の金額を順番に配列に格納しています。

硬貨の金額の大きいほうから処理します。
Basic では、割り算の結果に少数が含まれるため、少数以下切り捨てをする Int()で囲います。
また、剰余は％ではなく、Mod を使用します。

問３　次の文章を参考に，図２のプログラムの空欄［サ］～［タ］に入れるのに最も適当なものを，後の解答群のうちから一つずつ選べ。ただし，空欄［ス］・［セ］は解答の順序は問わない。

Ｔ：プログラム（図１）ができたようですね。それを使えば，関数「**枚数**(金額)」のプログラムができます。関数の引数として与えられる金額の値をプログラム（図１）の変数 **kingaku** に設定し，**(7)** 行目の代わりに変数 **maisu** の値を関数の戻り値とすれば，関数「**枚数**(金額)」のプログラムとなります。では，その関数を使って最小交換硬貨枚数を計算するプログラムを作ってみましょう。ここでも，100 円以下の買い物として考えてみます。

【関数の説明】(再掲)

> **枚数**(金額)… 引数として「金額」が与えられ，ちょうどその金額となる硬貨の組合せの中で，枚数が最小となる硬貨枚数が戻り値となる関数。

Ｓさんは，図２のようなプログラムを作成した。変数 **kakaku** に与えられる商品の価格に対して，釣り銭を表す変数 **tsuri** を用意し，妥当な **tsuri** のすべての値に対して交換する硬貨の枚数を調べ，その最小値を求めるプログラムである。なお，ここでは例として商品の価格を 46 円としている。

このプログラムでは，先生（Ｔ）のアドバイスに従い，釣り銭無しの場合も含め，99 円までのすべての釣り銭に対し，その釣り銭になるように支払う場合に交換される硬貨の枚数を求め，その最小値を最小交換硬貨枚数として計算している。

最小値の計算では，これまでの払い方での最小枚数を変数 **min_maisu** に記憶しておき，それより少ない枚数の払い方が出るたびに更新している。**min_maisu** の初期値には，十分に大きな値として 100 を用いている。100 円以下の買い物では，使う硬貨の枚数は 100 枚を超えないからである。


```
(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)
```

図２　最小交換硬貨枚数を求めるプログラム

このプログラムを実行してみたところ 3 が表示された。46 円を支払うときの最小交換硬貨枚数は，支払いで 50 円玉が 1 枚，1 円玉が 1 枚，釣り銭で 5 円玉が 1 枚の計 3 枚なので，正しく計算できていることが分かる。同様に，**kakaku** の値をいろいろと変えて実行してみたところ，すべて正しく計算できていることを確認できた。

［サ］，［ソ］・［タ］の解答群

⓪ **maisu**　① **min_maisu**　② **shiharai**　③ **tsuri**

［シ］の解答群

⓪ **0**　① **1**　② **99**　③ **100**

［ス］・［セ］の解答群

⓪ 枚数(**shiharai**)　① 枚数(**kakaku**)　② 枚数(**tsuri**)
③ **shiharai**　④ **kakaku**　⑤ **tsuri**

解説

BASIC プログラムで記述したプログラムを関数化します。

```
Function MAISU(Kingaku)
  Dim Kouka(5)
  For I=0 To 4 Step 1
    Read Kouka(I)
  Next
  Data 1,5,10,50,100
  maisu = 0
  nokori = kingaku
  For I=4 To 0 Step -1
    maisu = maisu + Int(nokori/kouka(I))
    nokori = nokori Mod kouka(I)
  Next
  MAISU = maisu
End Function
```

Kingaku は関数呼び出し時に受け取るため、ここで記述する。

関数の戻り値として指定する。

これで、プログラム中から、
maisu = MAISU(46)
のように記述すれば、46 円の硬貨枚数が maisu に格納されるようになる。

やり取りする硬貨の枚数が最小となる支払額を求めるプログラムを考える。

〔変数〕　商品の価格　kakaku
　　お釣り　tsuri
　　最小の枚数　min_maisu

お金を支払ってお釣りをもらう際に、やり取りする硬貨の枚数が最小のものを探す。つまり、支払う金額は kakaku + tsuri となり、お釣りとして受け取る金額は tsuri である。このため、MAISU(kakaku + tsuri)+MAISU(tsuri) が最小となるものを探せばよいこととなる。

商品の価格 kakaku は変わらないため、お釣りの額 tsuri を 0 から 99 まで変動させながら、やり取りする硬貨の枚数 maisu を計算し、最小の枚数が解となる。


```
(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)
```

図２　最小交換硬貨枚数を求めるプログラム

サは tsuri 、シは 0 、スは 枚数(shiharai)、セは 枚数(tsuri) となる。

そして、maisu が、今までの最小値 min_maisu より少なければ、
maisu を 最小値とするため、ソは maisu、タは min_maisu となる。

BASIC プログラムで、これを記述すると以下のようになる。

```
kakaku = 46
min_maisu = 100
For tsuri = 0 To 99 Step 1
  shiharai = kakaku + tsuri
  maisu = MAISU(shiharai) + MAISU(tsuri)
  If maisu < min_maisu Then
    min_maisu = maisu
  End If
Next
Print min_maisu
```

```
Function MAISU(Kingaku)
  Dim Kouka(5)
  For I=0 To 4 Step 1
    Read Kouka(I)
  Next
  Data 1,5,10,50,100
  maisu = 0
  nokori = kingaku
  For I=4 To 0 Step -1
    maisu = maisu + Int(nokori/kouka(I))
    nokori = nokori Mod kouka(I)
  Next
  MAISU = maisu
End Function
```

関数部分