

基本情報技術者

[科目B]アルゴリズムとプログラミング
トレーニング問題集（第2版）

はじめに

本トレーニング問題集（以下、本書という）は、国家試験である基本情報技術者の科目B試験に出題される「アルゴリズムとプログラミング」で得点するための実力を養成することを主眼に作成しました。別冊「アルゴリズム テキスト&ドリル」を用いたインプット学習後に本書をお使いください。

「アルゴリズムとプログラミング」は内容の特性から、得点アップのためには継続した学習の積み重ねがとても重要です。それは筋力トレーニングと似ており、「アルゴリズムとプログラミング」の本試験問題を解くための筋肉を、脳に構築していく必要があります。そのための**最大のポイントは、毎日「アルゴリズムとプログラミング」に触れることです。**

本書の制作コンセプトは、『問題文や空欄箇所を最小限に留め、1問当たり5～10分で解ける問題を揃えること』でした。それはひとえに『毎日、本書を開いてほしい』という教員一同の強い願いによるものです。一日に1問でも結構です。**毎日の演習を継続し、ひと通り解き終わったら二順目、三順目と繰り返し演習することで、本書は最大の効果を発揮します。**

本書を学習された皆さんが合格することを心よりお祈り申し上げます。

本書の使い方



1. 本書の位置付け

本書は、別冊「アルゴリズム テキスト&ドリル」の補助教材に位置しており、学習テーマや掲載順序が対応しています。インプット学習後に、本書を使って継続的にアウトプットのトレーニングをしてください。

2. チェック欄の使用法

チェック欄に「○」「△」「×」を付けながら問題を解きましょう。

記号	意味
○	理解して正解できた 復習が不要
△	正解できたが曖昧 二順目に復習が必要
×	正解できなかった 分からなかった

3. 解けない問題に直面した場合

問題が解けずに立ち止まってしまった方は、長く考え込まずに模範解答と解説文を読んで理解することを心がけましょう。「二順目のときには解けるように確認する」ことが重要です。また、どのような理由で問題が解けなかったのか、メモを残すようにしましょう。

《メモの例》

- 問題文の意図が読み取れなかった
- トレースがうまくできなかった
- トレースはできたが空欄にぶつかったときにひらめかなかった

4. 模範解答一覧と解説

本書の後半に、模範解答と解説文を掲載しています。間違えた(分からなかった)問題は、「どの部分で」「どのように」「なぜ間違えたか」を確認するようにしましょう。

チェックシート

第1部 アルゴリズムの表現方法

問題番号	ページ	テーマ	チェック欄 ※実施日(記号)	
練習1	P.12	擬似言語	/ ()	/ ()
練習2	P.12	擬似言語	/ ()	/ ()
練習3	P.13	擬似言語	/ ()	/ ()
練習4	P.13	擬似言語	/ ()	/ ()
練習5	P.14	擬似言語	/ ()	/ ()
練習6	P.14	擬似言語	/ ()	/ ()
練習7	P.15	擬似言語	/ ()	/ ()
練習8	P.16	擬似言語	/ ()	/ ()
練習9	P.17	擬似言語	/ ()	/ ()
練習10	P.18	擬似言語	/ ()	/ ()
練習11	P.19	擬似言語	/ ()	/ ()
練習12	P.20	擬似言語	/ ()	/ ()
練習13	P.21	擬似言語	/ ()	/ ()
練習14	P.22	擬似言語	/ ()	/ ()
練習15	P.23	擬似言語	/ ()	/ ()
練習16	P.24	擬似言語	/ ()	/ ()
練習17	P.25	擬似言語	/ ()	/ ()
練習18	P.26	擬似言語	/ ()	/ ()
練習19	P.27	擬似言語	/ ()	/ ()
練習20	P.28	擬似言語	/ ()	/ ()





第2部 データ構造とアルゴリズム①

問題番号	ページ	テーマ	チェック欄 ※実施日(記号)	
練習21	P.34	配列	/ ()	/ ()
練習22	P.34	配列	/ ()	/ ()
練習23	P.35	配列	/ ()	/ ()
練習24	P.36	配列	/ ()	/ ()
練習25	P.37	配列	/ ()	/ ()
練習26	P.38	配列	/ ()	/ ()
練習27	P.39	配列	/ ()	/ ()
練習28	P.40	配列	/ ()	/ ()
練習29	P.41	配列	/ ()	/ ()
練習30	P.42	配列	/ ()	/ ()
練習31	P.43	配列	/ ()	/ ()
練習32	P.44	配列	/ ()	/ ()
練習33	P.45	配列	/ ()	/ ()
練習34	P.46	配列	/ ()	/ ()
練習35	P.47	配列	/ ()	/ ()
練習36	P.48	配列	/ ()	/ ()
練習37	P.49	配列	/ ()	/ ()
練習38	P.50	配列	/ ()	/ ()
練習39	P.52	配列	/ ()	/ ()

第3部 代表的なアルゴリズム①

問題番号	ページ	テーマ	チェック欄 ※実施日(記号)	
練習40	P.58	探索 (サーチ)	/ ()	/ ()
練習41	P.58	探索 (サーチ)	/ ()	/ ()
練習42	P.59	探索 (サーチ)	/ ()	/ ()
練習43	P.60	探索 (サーチ)	/ ()	/ ()
練習44	P.62	探索 (サーチ)	/ ()	/ ()
練習45	P.63	探索 (サーチ)	/ ()	/ ()
練習46	P.64	探索 (サーチ)	/ ()	/ ()



第4部 データ構造とアルゴリズム②

問題番号	ページ	テーマ	チェック欄 ※実施日(記号)	
練習47	P.68	リスト	/ ()	/ ()
練習48	P.69	リスト	/ ()	/ ()
練習49	P.70	リスト	/ ()	/ ()
練習50	P.72	リスト	/ ()	/ ()
練習51	P.76	木	/ ()	/ ()
練習52	P.78	木	/ ()	/ ()

第5部 代表的なアルゴリズム②

問題番号	ページ	テーマ	チェック欄 ※実施日(記号)	
練習53	P.82	ハッシュ法	/ ()	/ ()
練習54	P.90	整列 (ソート)	/ ()	/ ()
練習55	P.91	整列 (ソート)	/ ()	/ ()
練習56	P.92	整列 (ソート)	/ ()	/ ()
練習57	P.93	整列 (ソート)	/ ()	/ ()
練習58	P.94	整列 (ソート)	/ ()	/ ()
練習59	P.95	整列 (ソート)	/ ()	/ ()
練習60	P.96	整列 (ソート)	/ ()	/ ()
練習61	P.97	整列 (ソート)	/ ()	/ ()
練習62	P.98	整列 (ソート)	/ ()	/ ()
練習63	P.100	整列 (ソート)	/ ()	/ ()
練習64	P.102	整列 (ソート)	/ ()	/ ()
練習65	P.104	整列 (ソート)	/ ()	/ ()
練習66	P.108	文字列処理	/ ()	/ ()
練習67	P.110	文字列処理	/ ()	/ ()
練習68	P.112	文字列処理	/ ()	/ ()
練習69	P.113	文字列処理	/ ()	/ ()
練習70	P.114	文字列処理	/ ()	/ ()
練習71	P.115	文字列処理	/ ()	/ ()
練習72	P.116	文字列処理	/ ()	/ ()
練習73	P.117	文字列処理	/ ()	/ ()
練習74	P.118	文字列処理	/ ()	/ ()
練習75	P.119	文字列処理	/ ()	/ ()
練習76	P.120	文字列処理	/ ()	/ ()
練習77	P.122	文字列処理	/ ()	/ ()

1

擬似言語

擬似言語は、「言語」というよりも、フローチャート流れ図をコンパクトに表現したものと考えればいいでしょう。

アルゴリズムの問題は、この擬似言語で処理手順が書かれていて、指示に従って空欄を埋める形になっていたり、実行結果を求めたりします。

擬似言語の仕様

擬似言語の要素は、「宣言」「注釈(コメント)」「代入」などの基本要素と、「選択」「繰返し」の制御構造、あとは「手続きの呼び出し(関数)」くらいです。

「選択」と「繰返し」の制御構造は、if, if~else, while, do~while, forに対応しています。「選択」と「繰返し」については、P.10、11で確認します。

共通に使用される擬似言語の記述形式

擬似言語を使用した問題では、各問題文中に注記がない限り、次の記述形式が適用されているものとする。

〔擬似言語の記述形式〕

記述形式	説明
○ <u>手続き名又は関数名</u>	手続き又は関数を宣言する。
<u>型名</u> ; <u>変数名</u>	変数を宣言する。
/* <u>注釈</u> */	注釈を記述する。
// <u>注釈</u>	
<u>変数名</u> ← <u>式</u>	変数に <u>式</u> の値を代入する。
<u>手続き名又は関数名</u> (<u>引数</u> , ...)	手続き又は関数を呼び出し、 <u>引数</u> を受け渡す。
if (<u>条件式1</u>) <u>処理1</u> elseif (<u>条件式2</u>) <u>処理2</u> elseif (<u>条件式n</u>) <u>処理n</u> else <u>処理n+1</u> endif	選択処理を示す。 <u>条件式</u> を上から評価し、最初に真になった <u>条件式</u> に対応する <u>処理</u> を実行する。以降の <u>条件式</u> は評価せず、対応する <u>処理</u> も実行しない。どの <u>条件式</u> も真にならないときは、 <u>処理n+1</u> を実行する。 各 <u>処理</u> は、0以上の文の集まりである。 elseif と <u>処理</u> の組みは、複数記述することがあり、省略することもある。 else と <u>処理n+1</u> の組みは一つだけ記述し、省略することもある。
while (<u>条件式</u>) <u>処理</u> endwhile	前判定繰返し処理を示す。 <u>条件式</u> が真の間、 <u>処理</u> を繰返し実行する。 <u>処理</u> は、0以上の文の集まりである。
do <u>処理</u> while (<u>条件式</u>)	後判定繰返し処理を示す。 <u>処理</u> を実行し、 <u>条件式</u> が真の間、 <u>処理</u> を繰返し実行する。 <u>処理</u> は、0以上の文の集まりである。
for (<u>制御記述</u>) <u>処理</u> endfor	繰返し処理を示す。 <u>制御記述</u> の内容に基づいて、 <u>処理</u> を繰返し実行する。 <u>処理</u> は、0以上の文の集まりである。

〔演算子と優先順位〕

演算子の種類	演算子	優先度
式	() ,	高 ↑
単項演算子	not + -	
二項演算子	乗除 mod × ÷	↓ 低
	加減 + -	
	関係 ≠ ≤ ≥ < = >	
	論理積 and	
	論理和 or	

注記 演算子 . は、メンバ変数又はメソッドのアクセスを表す。
演算子 mod は、剰余数を表す。

〔論理型の定数〕

true, false

擬似言語の基本的な形

擬似言語の前半には、変数や関数などの「宣言」部があります。まずここを確認し、使用する変数を把握しましょう。また、各変数の初期値を代入している部分も、必ずチェックしておきましょう。

記述例

○ 宣言
型名：変数名
for
繰返し
endfor
if
選択
endif

重要

1行目は○から始まり、「:」がある行は
利用する変数を準備している



擬似言語の問題を攻略するためには、「選択」「繰返し」「関数」の記述に慣れて、動きを読み取ることが不可欠です。一つのプログラムは、一般的に次のような構成になります。

○ 整数型：summation (整数型：n)
整数型：i, m, p, ans

宣言

m ← n ÷ 2 /*mは加算回数*/
p ← 1 + n /*pは先頭と末尾の和*/
ans ← 0
i ← 1

初期値

while (i が m 以下)
ans ← ans + p
i ← i + 1
endwhile

繰返し

if ((m × 2) が n と等しくない) /*nは奇数か*/
(1)
endif

選択
(条件判断)

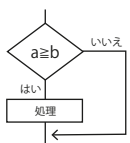
return ans

処理

制御構造の使用例

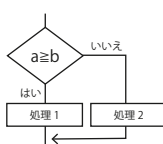
●選択1 (if)

aがb以上なら「処理」を実行

擬似言語	フローチャート
<pre>if (aがb以上) 条件式 処理 endif</pre> 重要 選択はif 	

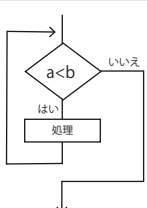
●選択2 (if～else)

aがb以上なら「処理1」を実行、そうでなければ「処理2」を実行

擬似言語	フローチャート
<pre>if (aがb以上) 条件式 処理 1 else 処理 2 endif</pre> 重要 基本になる 2分岐の形 	

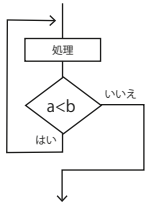
●繰返し1 (while)

先頭判定でaがb未満なら「処理」を繰り返す

擬似言語	フローチャート
<pre>while (aがbより小さい) 条件式 処理 endwhile</pre> 重要 条件が真、つまり成立 していれば繰り返す 	

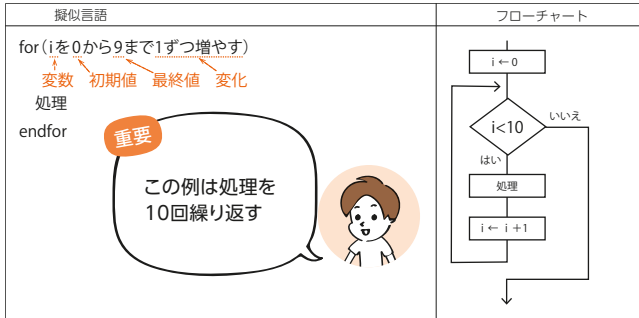
●繰返し2 (do～while)

末尾判定でaがb未満なら「処理」を繰り返す

擬似言語	フローチャート
<pre>do 処理 while (aがbより小さい) 条件式</pre> 重要 末尾判定なので最低1回は処 理を実行する 	

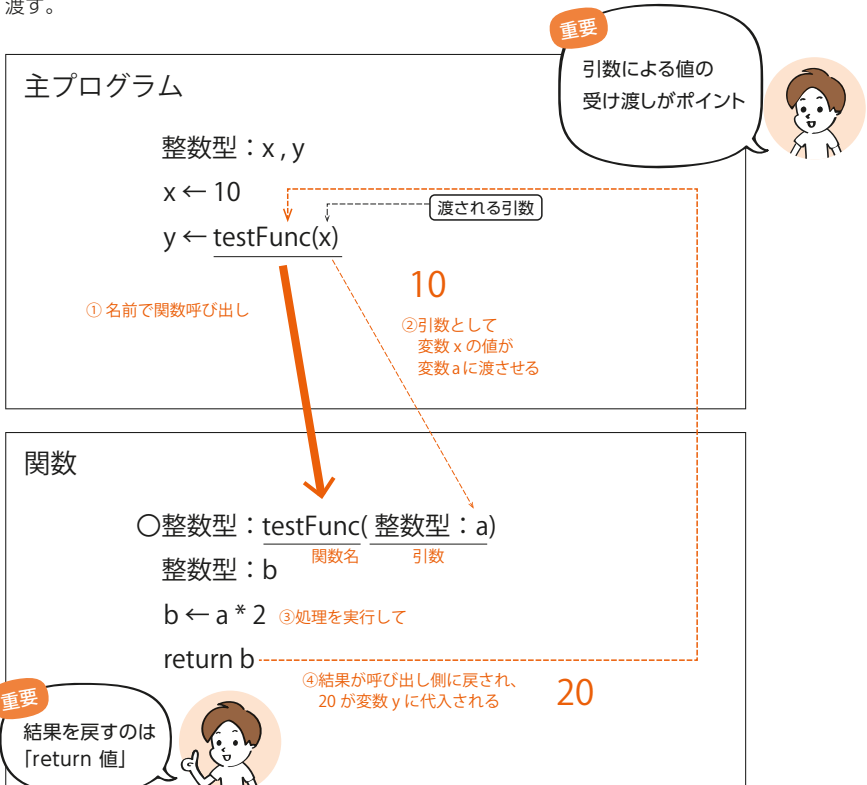
●繰返し3(for)

()の中に、繰返しをコントロールするための変数、初期値と終了値、変化の仕方を書く



●関数

関数は冒頭の宣言部で関数名と引数を宣言し、それに合わせた関数名で呼び出して、引数データを渡す。



練習

1

解答 →
P.126標準
学習時間
5min

引数tenに受け取った100点満点のテストの点数の評価を,"A"(80点以上の場合),"B"(60点以上80点未満の場合),"C"(それ以外の場合)のいずれかで呼出し元に返す関数seisekiを定義する。

```
○ 文字型 : seiseki(整数型 : ten)
if (  (1)  )
    return "A"
elseif (  (2)  )
    return "B"
else
    return "C"
endif
```

解答群 ア tenが79より大きい
ウ tenが80より大きい
オ tenが60より小さい
キ tenが59以上

イ tenが79以上
エ tenが59以下
カ tenが59より大きい
ク tenが60より大きい

練習

2

解答 →
P.126標準
学習時間
5min

引数tenに受け取った100点満点のテストの点数の評価を,"A"(80点以上の場合),"B"(60点以上80点未満の場合),"C"(それ以外の場合)のいずれかで呼出し元に返す関数seisekiを定義する。

```
○ 文字型 : seiseki(整数型 : ten)
if (  (1)  )
    return "C"
elseif (  (2)  )
    return "A"
else
    return "B"
endif
```

解答群 ア tenが79より大きい
ウ tenが80より大きい
オ tenが60以下
キ tenが59以上

イ tenが79以上
エ tenが59以下
カ tenが59より大きい
ク tenが60より大きい

引数yに受け取った4桁の正の整数値を西暦年とみなし、それがうるう年か否かを判定する。うるう年とは、西暦年が次の条件を満たす年である。

- (1) 400で割り切れる。
- (2) 4で割り切れ、100で割り切れない。

```

○ isleap(整数型 : y)
  論理型 : leap ← false
  if (y mod 400)が0と等しい    /* modは剰余演算子 */
    leap ← true
  elseif (y mod 4)が0と等しい
    if (y mod 100)が0と等しくない
      leap ← true
    endif
  endif
  if ( (1) )
    "うるう年でない" と出力
  else
    "うるう年である" と出力
  endif

```

解答群 ア leapがtrueと等しい イ (not leap)がtrueと等しい

入力装置から0以上の数値を変数dataに読み込み、その和を求める。数値は0件以上入力され、負の数値が入力されるまで処理を繰り返す。

```

実数型 : ans, data
ans ← 0.0
dataに数値を入力
while (dataが0.0以上)
  (1)
  dataに数値を入力
endwhile
ansを出力

```

解答群 ア ans ← data イ data ← ans + data
 ウ ans ← ans + data



入力装置から、負の数値が入力されるまで数値を変数dataに読み込み、その平均を求める（負の数値は含めない）。

実数型 : sum, data, avg

整数型 : n

sum ← 0

n ← 1

dataに数値を入力

while (dataが0.0以上)

 sum ← sum + data

 n ←

 dataに数値を入力

endwhile

n ←

if (nが0でない)

 avg ← sum ÷ n

endif

解答群 ア n - 1 イ n + 1



1～10の値を変数ansに加算する。

整数型 : ans, atai

ans ← 1

for (ataiを 1ずつ増やす)

 ans ← ans + atai

endfor

解答群 ア 0から9まで イ 1から10まで ウ 2から10まで



1からnまでの整数の和を, $n/2$ 回の加算の繰返しで求める。

○ 整数型 : summation (整数型 : n)

整数型 : i, m, p, ans

$m \leftarrow n \div 2$

/* mは加算回数, 端数は切り捨て */

$p \leftarrow 1 + n$

/* pは先頭と末尾の和 */

$ans \leftarrow 0$

$i \leftarrow 1$

while (i が m 以下)

$ans \leftarrow ans + p$

$i \leftarrow i + 1$

endwhile

if ($m \times 2$ が n と等しくない)

/* nは奇数か */

(1)

endif

return ans

解答群

ア $ans \leftarrow ans - 1$

イ $ans \leftarrow ans + 1$

ウ $ans \leftarrow ans - i$

エ $ans \leftarrow ans + i$



引数dに受け取った正の整数値の0の桁を全て5に置き換えた値を返す。

例 d=1002030のとき1552535を, d=12345のとき12345を返す。

○ 整数型 : zero2five (整数型 : d)

整数型 : x, y, z

y ← 0

z ← 1

while (dが0より大きい)

x ← d mod 10

/* xはdの1の位の値 */

if (xが0と等しい)

y ← y + 5 × z

else

y ← y + x × z

endif

d ← (1)

z ← (2)

endwhile

return y

解答群 ア d ÷ 10 イ d × 10

 ウ z ÷ 10 エ z × 10

1からnまでの整数を順番に表示する。ただし、その整数が3で割り切れるときは"Fizz"を、5で割り切れるときは"Buzz"を、3でも5でも割り切れるときは"FizzBuzz"を、それぞれ整数の代わりに表示する。

- (1) nの値は100とする。
- (2) aをbで割った余りは、 $a \bmod b$ で求める。
- (3) 整数や文字列を表示するごとに改行する。
- (4) 表示や改行は、次の例に従って行う。

処理内容	プログラムでの記述方法
変数iの内容を表示する	iを表示
文字列"abc"を表示する	"abc"を表示
改行する	改行を実行

```

○ fizzbuzz ( )
  整数型 : i                               /* 1から数えるカウンタ */
  整数型 : n
  n ← 100
  i ← 1
  while ( i が n 以下 )
    if ( (1) )
      "Fizz" を表示
    if ( (2) )
      "Buzz" を表示
    endif
  else
    if ( (2) )
      "Buzz" を表示
    else
      i を表示
    endif
  endif
  改行を実行
  i ← i + 1
endwhile

```

解答群 **ア** ($i \bmod 3$) が 0 と等しくない
 イ ($i \bmod 3$) が 0 と等しい
 ウ ($i \bmod 5$) が 0 と等しくない
 エ ($i \bmod 5$) が 0 と等しい



n 以下の素数を出力する。素数とは、1とその数自身でしか割り切れない正の整数である。ここでは、次のことは既知とする。

- (1) 最小の素数は2である。
- (2) 2以外の素数は必ず奇数である。
- (3) 2より大きい素数でない奇数 α は、 $3 \sim \alpha - 1$ の範囲の数の中の奇数を約数にもつ。つまり、この範囲のいずれかの奇数で割り切れる。

○ primenumber(整数型: n)

整数型: i, j, flg

2を出力

/* 最初の素数2を出力 */

for ()

flg ← 1

j ← 3

while ((jが $i - 1$ 以下) and (flgが1と等しい))

if (($i \bmod j$)が0と等しい) /* modは剰余演算子 */

flg ← 0

endif

j ← j + 2

endwhile

if ()

i を出力

endif

endfor

- 解答群
- ア i を2から n まで1ずつ増やす
 - イ i を2から n まで2ずつ増やす
 - ウ i を3から n まで2ずつ増やす
 - エ i を3から n まで3ずつ増やす
 - オ flgが0と等しい
 - カ flgが1と等しい
 - キ flgが1と等しくない

解答解説

模範解答一覧

練習1	(1)ア (2)カ
練習2	(1)エ (2)ア
練習3	(1)イ
練習4	(1)ウ
練習5	(1)イ (2)ア
練習6	(1)ウ
練習7	(1)エ
練習8	(1)ア (2)エ
練習9	(1)イ (2)エ
練習10	(1)ウ (2)カ
練習11	(1)イ (2)エ
練習12	(1)ア (2)エ
練習13	(1)イ (2)オ
練習14	(1)エ (2)イ
練習15	(1)ウ
練習16	(1)ア (2)イ (3)カ
練習17	(1)イ (2)カ
練習18	(1)ア (2)サ
練習19	(1)ウ (2)イ
練習20	(1)ア (2)カ (3)エ
練習21	(1)ア (2)キ
練習22	(1)エ (2)キ
練習23	(1)ウ (2)エ
練習24	(1)ウ (2)カ
練習25	(1)ウ (2)オ
練習26	(1)エ (2)イ

練習27	(1)ウ
練習28	(1)キ (2)オ (3)イ
練習29	(1)ア
練習30	(1)ア
練習31	(1)ア (2)エ (3)ウ (4)オ
練習32	(1)ア (2)オ
練習33	(1)ア (2)カ (3)エ
練習34	(1)オ
練習35	(1)エ
練習36	(1)ウ (2)カ
練習37	(1)エ (2)ク
練習38	(1)イ (2)エ (3)ウ (4)カ
練習39	(1)ア (2)オ
練習40	(1)エ
練習41	(1)イ
練習42	(1)ウ
練習43	(1)イ (2)ウ
練習44	(1)イ (2)ア (3)ウ (4)ク
練習45	(1)ウ (2)イ
練習46	(1)カ (2)ア (3)ア
練習47	(1)ウ
練習48	(1)オ (2)エ
練習49	(1)エ (2)カ
練習50	(1)ア (2)エ
練習51	(1)ア (2)イ
練習52	(1)ク (2)エ (3)カ

練習53	(1)ウ (2)カ
練習54	(1)イ (2)ウ
練習55	(1)イ (2)オ (3)ケ
練習56	(1)ア
練習57	(1)ウ
練習58	(1)エ (2)キ
練習59	(1)ウ
練習60	(1)イ (2)エ
練習61	(1)カ (2)ウ
練習62	(1)オ (2)ア (3)カ
練習63	(1)ウ (2)ア (3)イ (4)ア
練習64	(1)イ (2)カ (3)ク (4)ケ
練習65	(1)ク (2)カ (3)ウ (4)キ
練習66	(1)ウ
練習67	(1)ウ (2)カ
練習68	(1)ア
練習69	(1)エ
練習70	(1)オ (2)ウ
練習71	(1)ウ
練習72	(1)イ
練習73	(1)イ (2)ウ (3)オ
練習74	(1)エ
練習75	(1)ウ (2)ア
練習76	(1)カ (2)オ (3)ア
練習77	(1)イ (2)エ (3)ウ

1 擬似言語

練習

1

問題 →
P.12

解答 (1)ア tenが79より大きい (2)カ tenが59より大きい

<考え方>

if文により、受け取った点が80点以上かどうかを判断する。

・80点以上の場合、呼出し元に"a"を返す。

80点以上ではない場合、elseif文により60点以上かどうかを判断する。

・60点以上の場合、呼出し元に"b"を返す。

60点以上でもない場合、呼出し元に"c"を返す。

- (1) 真と判定された場合に"a"が返されることより、引数tenに受け取った点数が80点以上の場合に真と判定される条件式、すなわち「tenが80以上」が当てはまる。ただし、これは選択肢の中にはないため、同等の意味をもつ「tenが79より大きい」を選ぶ。
- (2) 真と判定された場合に"b"が、偽と判定された場合には"c"が返されることにより、引数tenに受け取った点数が60点以上、すなわち「tenが59より大きい」が当てはまる。

練習

2

問題 →
P.12

解答 (1)エ tenが59以下 (2)ア tenが79より大きい

<考え方>

設問の内容は前題と同じだが、条件を逆に考えていく。

if文により、受け取った点が、60点未満かどうかを判断する。

・60点未満の場合、呼出し元に"c"を返す。

60点未満ではない場合、elseif文により80点以上かどうかを判断する。

・80点以上の場合、呼出し元に"a"を返す。

80点以上ではない場合、呼出し元に"b"を返す。

- (1) 真と判定された場合に"c"が返されることにより、引数tenに受け取った点数が60点未満、すなわち59点以下の場合に真と判定される「tenが59以下」が当てはまる。
- (2) 真と判定された場合に"a"が、偽と判定された場合には"b"が返されることから、引数tenに受け取った点数が80点以上、すなわち79点を上回る場合に真と判定される「tenが79より大きい」が当てはまる。

練習 3

問題 →
P.13

解答 (1) **イ** (not leap) がtrueと等しい

<考え方>

受け取った4桁の整数値を西暦年号とみなし、うるう年か否かを判断する。剰余(余り)を求めるために演算子「mod」を使用する。/*で始まり*/で終わる記述は注釈(コメント)であり、実行命令ではない。注釈は問題を解く上でのヒントになることが多いため、気を付けて見るようにしよう!

論理型変数leapの初期値をfalse(平年を意味する)としておき、西暦年号が400で割り切れるかどうかを判断する。

・割り切れる場合は論理型変数leapをtrue(うるう年を意味する)に変更する。

400で割り切れない場合は、4で割り切れるかどうかを判断し、4で割り切れる場合には、100で割り切れないかどうかを判断する。

・4で割り切れ、かつ100で割り切れない場合、論理型変数leapをtrueに変更する。

論理型変数leapに設定された値(真理値)により、うるう年か否かを判断する。

・うるう年でない場合(leapがfalse)は、「うるう年でない」を出力

・そうでない場合(leapがtrue)は、「うるう年である」を出力

例えば、西暦2000年は、400で割り切れるため、うるう年である。一方、1900年や2100年などは、400では割り切れず、4でも100でも割り切れるため、平年と判断される。

- (1) 論理型変数leapの初期値はfalseであるが、うるう年の条件を満たす場合にtrueに切り替わる。つまり、leapは、うるう年ならばtrueに、うるう年でなければfalseになる。空欄の条件が真と判定された場合に「うるう年でない」と出力するため、空欄の条件は「leapがfalseと等しい」となるが、これは選択肢の中にはないため、演算子notにより論理変数leapの真偽を反転させ、それがtrueのときに「うるう年でない」と出力する。

なお、論理型変数はそれ自身が真偽のいずれかを保持するため、条件式で「trueと等しい／等しくない」、「falseと等しい／等しくない」のような比較を行わず、論理型変数だけを条件式に指定することもできる。

練習 4

問題 →
P.13

解答 (1) **ウ** ans ← ans + data

<考え方>

繰返し処理の直前にある「dataに数値を入力」で、入力装置から一つ目の数値を変数dataに読み込む。二つ目以降の数値は、繰返し内にある「dataに数値を入力」で入力されるが、負の数値が入力されると、繰返しの継続条件「dataが0.0以上」が成立しなくなり、繰返しが終了する。それまでに入力された数値の和は変数ansに求める。数値は0件以上入力されるため、1回目でも負の数値が入力された場合は、繰返し処理は1回も実行されない。

- (1) 変数ansの初期値は0.0であるため、二つ目の数値を入力する前に一つ目の数値を変数ansに加算する必要がある。以降、変数dataに入力される負ではない数値を次々に加算していく。

練習 5

問題 →
P.14

解答 (1)イ $n + 1$ (2)ア $n - 1$

<考え方>

設問の内容は前題と似ているが、和ではなく平均を求める。平均は、入力された非負の数値の和をそれらの個数で割ることにより求める。

各変数が保持する値は次のとおりである。

変数	保持する値
sum	入力された数値の和
n	入力された数値の個数
data	入力された数値
avg	入力された数値の平均

- 入力された数値の平均を求めるためには、数値の合計と個数が必要となる。合計は変数sumに求めるので、空欄(1)を含む処理が、個数を変数nに求めるための処理に該当する。数値の個数は、数値を読み込む都度nに1を加算することで求めることができる。
- 直前の繰返し処理が終了した時点で、変数dataには負の数値が入力されており、変数nにはその負の数値の個数が余分に加算されている。したがって、それを除くために、nから1を引く。

練習 6

問題 →
P.14

解答 (1)ウ 2から10まで

<考え方>

繰返し処理において変数ataiの値を変化させながら変数ansに加算していくことで、1～10の合計を変数ansに求める。変数ansの初期値が1である点に注意する。

- 変数ansの初期値が1であるため、繰返し処理では、変数ansに加算する変数ataiが2から10まで1ずつ変化するようにする。

解答 (1) ① ans ← ans + i

<考え方>

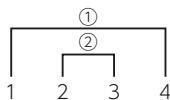
設問の内容は前題とほぼ同じだが、1～nまでの整数の和を、n/2回の加算の繰返しで求める。変数pの使い方が重要である。

1からnまでの整数の和を、

$$1+2+3+\cdots+(n-2)+(n-1)+n$$

のように、先頭+末尾、先頭から2番目+末尾から2番目、…と数値を二つずつ組み合わせて加算することで、いずれの加算結果も変数pに設定した1 + nとなり、加算の回数は変数mに設定したn ÷ 2回となる。

ただし、nが奇数のときには、中央の値が加算されずに繰返しが終了するため、繰返し終了後にnが奇数か否かを判定し、奇数の場合には中央の値を加算する。

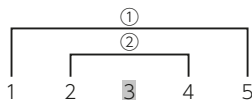


$$\textcircled{1} 1+4=5$$

$$\textcircled{2} 2+3=5$$

2 回足し算をし、10 を求める

n が偶数の場合



$$\textcircled{1} 1+5=6$$

$$\textcircled{2} 2+4=6$$

2 回足し算をし、12 を求め、3 を加える

n が奇数の場合

- (1) nが奇数のとき、変数mには端数0.5が切り捨てられた整数が求まる。その結果、中央に位置する値が加算されずに繰返し処理が終了するため、繰返し処理の終了後、 $m \times 2 \neq n$ が成り立つか否かを調べ、成り立つときにはnが奇数であるので、まだ加算されていない中央に位置する値をansに加算する。

nが奇数のとき、中央に位置する値は $\frac{n+1}{2}$ であり、繰返し処理では、その前後の数値の組合せまでが加算される。

$$\cdots + \frac{n-1}{2} + \frac{n+1}{2} + \frac{n+3}{2} + \cdots$$

繰返し処理での最後の加算

その後、i が1増えて $\frac{n+1}{2}$ になった時点で繰返しが終了するため、変数ansにはiを加算すればよい。

練習 8

問題 →
P.16

解答 (1)ア $d \div 10$ (2)エ $z \times 10$

<考え方>

引数dに受け取った正の整数値の0の桁を全て5に書き換えた値を返す。

数値に含まれる全ての0を5に置き換えるためには、数値を1桁ずつ処理する必要があるが、dの値が可変である、つまり最上位の位が特定できないため、上位桁からの処理では処理が煩雑になる。一方、下位桁からの処理であれば、dが幾つであっても、常に1の位から処理を開始でき、処理が簡潔になる。

具体的には、

- ①dが0になるまで、10で割りながら
- ②その時点の1の位の値を取り出し、それが0ならば5に、0でないならば1の位そのものに
- ③取り出した値の元々の位に相当する値を掛けて加算

という処理を行えばよい。

例えば、 $d=1002030$ を 1552535 に変換する場合の処理イメージは、次のようになる。

d	x(dの1の位)	z(位取り)	加算する値	y(合計)
1002030	0	1	5×1	5
100203	3	10	3×10	35
10020	0	100	5×100	535
1002	2	1000	2×1000	2535
100	0	10000	5×10000	52535
10	0	100000	5×100000	552535
1	1	1000000	1×1000000	1552535

- (1) 元の数値の1の位、10の位、100の位、…を、10で割ったときの剰余($d \bmod 10$)で変数xに取り出すために、繰返しの都度、dを10で割ることで、対象となる桁を1の位に移動させる。
- (2) 取り出すのは1の位の値であるが、実際には、それぞれ元の1の位の値、10の位の値、100の位の値、…であるため、位取りの値を示す変数zを、繰返しの都度10倍する。

練習 9

問題 →
P.17

解答 (1)イ ($i \bmod 3$)が0と等しい (2)エ ($i \bmod 5$)が0と等しい

<考え方>

1から $n(=100)$ までの整数について、3又は5で割り切れる、つまり3又は5で割った余りが0か否かを調べ、その結果により文字列又は数字を表示する。

- (1) 条件が成立した場合に"Fizz"を表示するので、iが3で割り切れたときに真と判定される、[($i \bmod 3$)が0と等しい]が当てはまる。
- (2) 条件が成立した場合に"Buzz"を表示するので、iが5で割り切れたときに真と判定される[($i \bmod 5$)が0と等しい]が当てはまる。

練習 10

問題 →
P.18

解答 (1)ウ iを3からnまで2ずつ増やす (2)力 flgが1と等しい

<考え方>

2以外の素数は全て奇数であるため、一つ目の素数2を出力後、3以上の奇数のみを対象に、3からその奇数-1までの範囲の奇数で割ったときの余りを求める。範囲内の全ての奇数に対する余りを求め終わるか、又は余りが0になったら、繰返しを終了する。

繰返しが終了した時点で変数flgが初期値の1のままであれば、範囲内の全ての奇数では割り切れなかったことを示し、iは素数と判定できる。一方、変数flgが0に変わっていたら、範囲内のいずれかの奇数で割り切れたことを示し、iは素数ではないと判定できる。

素数でない奇数 α は、3から $\alpha-1$ の範囲の奇数を約数にもつ。例えば、9が素数かどうかの判断は以下のように考える。

・9が3、5、7のいずれかで割り切れるかどうかを調べる。この中に一個でも約数があれば、つまり9を割り切る値があれば、9は素数ではない。

⇒9は3で割り切れるため、素数ではない。

- (1) 最小の素数2は直前で出力済みなので、素数か否かを判定される数を示す変数iが、それ以降のn以下の奇数の値をとる、すなわち、3からnまで2ずつ増加していくようにする。
- (2) 変数iの値が素数でなかった場合には、内ループ内で変数flgに0が設定され、内ループが終了する。一方、変数iの値が素数であった場合には、内ループ終了時点の変数flgの値は1のままとする。
この条件式が真の場合にiを出力しているため、iが素数であったときに真となる「flgが1と等しい」が当てはまる。

練習 11

問題 →
P.19

解答 (1)イ x^2 (2)エ $x1 + i \times \text{delta}$

<考え方>

$y=f(x)=x^2$ 、 x 軸、直線 $x=x_1$ 、及び直線 $x=x_2$ で囲まれる領域の面積 s を、問題で与えられた数式を用いて計算する。この方法は「台形則」と呼ばれる、数値積分の解法の一つである。

この種のプログラムでは、数式とプログラムとを対応付けることで、数式自体が理解できなくてもも解答が可能である。

- (1) 面積 s を求める際、範囲の両端における関数値の和「 $f(x_1)+f(x_2)$ 」は、 n の値によらず常に同じであるため、 n の値ごとにその都度計算するのではなく、あらかじめ計算し、変数uに格納しておく。
- (2) この処理を含む繰返し処理は、

$$\sum_{i=1}^{n-1} f(x_1 + i \times \delta)$$

の部分の結果を変数tに求める処理である。

$\sum_{i=1}^{n-1}$ は、iを1からn-1まで1ずつ変えながら、後ろに記述されたものの合計を求める計算を示す記号であり、本問では

$$f(x_1 + 1 \times \delta) + f(x_1 + 2 \times \delta) + \cdots + f(x_1 + (n-1) \times \delta)$$

となり、プログラムでは