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３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。

資格の大原　情報処理講座本部

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. チェック欄の使用方法

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	/　(　)	/　(　)

2 配 列

「配列」自体はアルゴリズムの話ではありません。配列というのは変数の一種で、「データ構造」という分野の話です。

なぜアルゴリズムの解説に配列の項があるかというと、探索・並べ替え・文字列処理など、ほとんどのアルゴリズムで、対象になるデータを配列に入れて扱うからです。

配列の扱いに慣れておかないと、アルゴリズム以前の問題として、やっている操作の意味が理解できなくなってしまいます。

配列は中を区切った箱

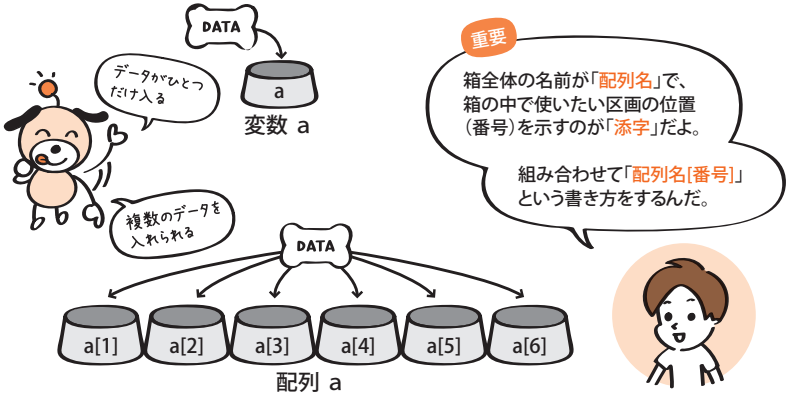
通常の変数は、データがひとつだけ入る箱です。配列も同様にデータを入れる箱と考えていいのですが、箱の中を区切っておくことにより、複数のデータを入れることができます。

配列内の区画には、一連の番号が付けられています。「a」という名前の配列があり、その「3番の区画」を使いたければ、a[3]といった書き方で指定します。この場合、「a」を配列名、[3]の部分を添字(そえじ)と呼びます。

添字は一連の番号なので、[3]と[4]の間にひとつ割り込ませるというように、配列の途中にデータを挿入／削除するのは面倒です。それ以降のデータを全部ひとつずつずらす、という処理が必要になるためです。

配列を扱う問題では、「最大N個のデータ」というように、たいていは最大データ数が指定されています。これが添字(番号)の最大値になるので、しっかり把握しておきましょう。

添字の番号は、0から始まる場合と、1から始まる場合とがあります。必ず問題文で確認してください。

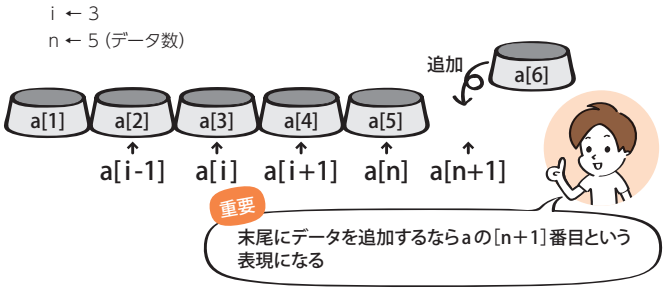


よく使う配列内の位置指定

実際のプログラムやアルゴリズムの問題では、たいていの場合、添字には変数を使います。

そして、例えば変数iで添字を指定したら、その前後という意味で、[i-1]や[i+1]といった添字もよく使われます。こうした書き方にも慣れておきましょう。

1次元配列の添字と位置

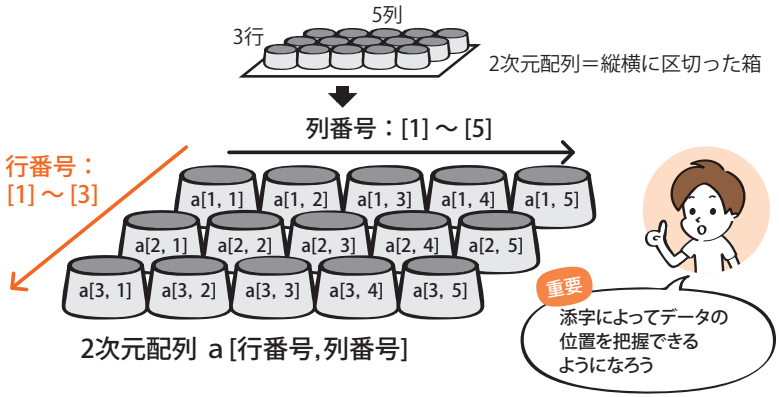


2次元配列にも慣れておこう

a[2]などというように、添字がひとつ付いている配列を「1次元配列」と言います。配列には添字が2つ以上付いたものもありますが、少なくとも、添字が2つ付いた「2次元配列」までは、使いこなせるようになっておきましょう。

2次元配列は、ひとつの箱を縦横に区切ったもの、と考えてください。縦方向と横方向に各々番号を付けて、その組み合わせで箱の中の区画の位置を指定します。

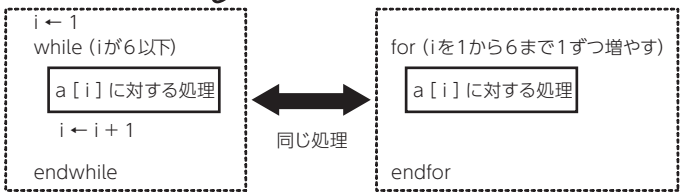
添字は、縦横の位置を示す数字を[]で挟んで、a[3, 2]などというように、変数名の後に2つ並べて書きます。

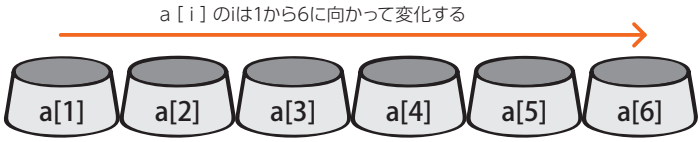


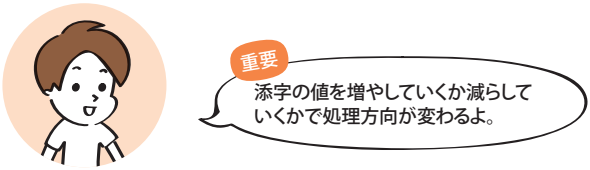
「繰返し」処理と配列

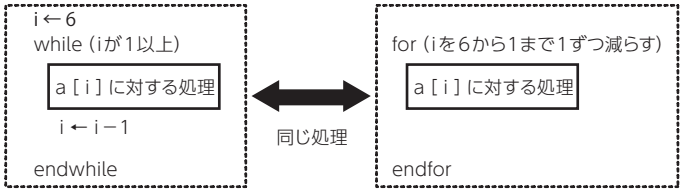
実際のプログラムでは、配列のデータは「繰返し」で処理することが多くなります。アルゴリズムの記述を見て、配列をどのような順番で処理しているのか、把握できるようにしておきましょう。

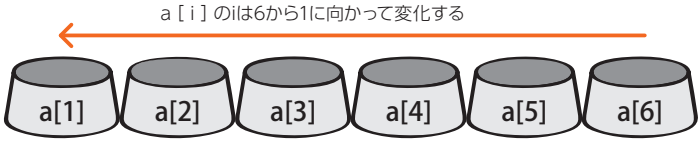
1次元配列の処理方向









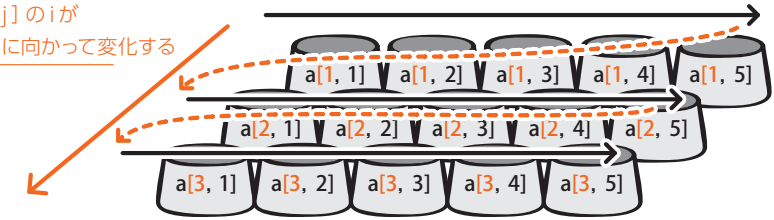


2次元配列の処理方向

```
i ← 1
j ← 1
while (iが3以下)
  while (jが5以下)
    a[i, j]に対する処理
    j ← j + 1
  endwhile
  i ← i + 1
  j ← 1
endwhile
```

a[i, j]のjが1から5に向かって変化する

a[i, j]のiが
1から3に向かって変化する



重要

添字が2つあるから、
どの添字を変えるとどの方向に
動いていくのか、変数と添字の
関連で把握しておこう。

練習 21

解答➡ P.138
標準学習時間 5min

10進数n(0≦n≦255)を8桁の2進数に変換する。2進数は下位桁から順に,配列の要素nishin[1]からnishin[8]に格納する。

例　n=157(=10011101_2)のとき

	1	2	3	4	5	6	7	8
nishin	1	0	1	1	1	0	0	1

○ tobin(整数型：n, 整数型の配列：nishin)
　整数型：i
　for（iを1から8まで1ずつ増やす）
　　nishin[(1)] ← n mod 2
　　(2)
　endfor

解答群　ア　i　　イ　8－i　　ウ　8＋i
　　エ　9－i　　オ　9＋i　　カ　n ← n × 2
　　キ　n ← n ÷ 2

練習 22

解答➡ P.138
標準学習時間 5min

10進数n(0≦n≦255)を8桁の2進数に変換する。2進数は下位桁から順に,配列の要素nishin[8]からnishin[1]に格納する。

例　n=157(=10011101_2)のとき

	1	2	3	4	5	6	7	8
nishin	1	0	0	1	1	1	0	1

○ tobin(整数型：n, 整数型の配列：nishin)
　整数型：i
　for（iを1から8まで1ずつ増やす）
　　nishin[(1)] ← n mod 2
　　(2)
　endfor

解答群　ア　i　　イ　8－i　　ウ　8＋i
　　エ　9－i　　オ　9＋i　　カ　n ← n × 2
　　キ　n ← n ÷ 2

練習 23

解答➡ P.139

標準学習時間 10min

要素番号(添字)1～nの位置に値が格納されている配列aを対象とする,次の二つの手続を定義する。

•手続insert

要素番号m(1≦m≦n)以降の値をそれぞれ一つ後ろにずらし,要素番号mの位置に値dを追加する

•手続delete

要素番号m+1(1≦m≦n)以降の値をそれぞれ一つ前にずらすことにより,要素番号mの位置の値を削除する

なお,配列aは十分な領域が確保されているものとする。

○ insert(整数型の配列：a, 整数型：n, 整数型：m, 整数型：d)

```
整数型：i
i ← n
while（i がm以上）
  (1)
  i ← i − 1
endwhile
a[m] ← d
n ← n + 1
```

○ delete(整数型の配列：a, 整数型：n, 整数型：m)

```
整数型：i
i ← m
while（i がnより小さい）
  (2)
  i ← i + 1
endwhile
n ← n − 1
```

解答群　ア　a[i − 1] ← a[i]　　イ　a[i] ← a[i − 1]

　　ウ　a[i + 1] ← a[i]　　エ　a[i] ← a[i + 1]

練習 24

解答➡ P.139

標準学習時間 5min

配列aの1番目からn番目までに格納されたn個の値を，mが示す要素数分，左方向にローテート（巡回シフト）する。ここで，配列の要素番号は1から始まる。

例　n=7, m=3の場合

ローテート前

	[1]	[2]	[3]	[4]	[5]	[6]	[7]
a	10	20	30	40	50	60	70

ローテート後

	[1]	[2]	[3]	[4]	[5]	[6]	[7]
a	40	50	60	70	10	20	30

```
○ rotate_l(整数型の配列：a, 整数型：n, 整数型：m)
  整数型：t, i , j
  for（ i を1からmまで1ずつ増やす）
    t ← a[1]                          /* 先頭要素の退避 */
    for（ j を2からnまで1ずつ増やす）
      (1)
    endfor
    (2)                               /* 退避した要素の挿入 */
  endfor
```

解答群　ア　a[j] ← a[j − 1]　　イ　a[j] ← a[j + 1]
　　ウ　a[j − 1] ← a[j]　　エ　a[j + 1] ← a[j]
　　オ　a[1] ← t　　カ　a[n] ← t

練習 25

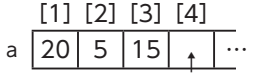
解答➡ P.140

標準学習時間 5min

配列を用い，後入れ先出しのデータ構造であるスタックに対する操作，pushとpopをシミュレートする。ここで，配列の要素番号は1から始まり，処理中に配列範囲を超える状態は生じないものとする。

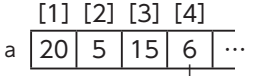
例

push(6)の実行



6 その時点の末尾の直後にデータを格納

pop( )の実行



その時点の末尾のデータを返す

大域：整数型：sp ← 1
大域：整数型の配列：a

○ push(整数型：x)

(1)

○ 整数型：pop( )

(2)

解答群

ア a[sp + 1] ← x
sp ← sp + 1

イ sp ← sp + 1
a[sp] ← x

ウ a[sp] ← x
sp ← sp + 1

エ return a[sp − 1]

オ sp ← sp − 1
return a[sp]

カ return a[sp]
sp ← sp − 1

練習 26

解答➡ P.140

標準学習時間 10min

n個の整数値が格納された配列aについて，降順（大きい順）に付けた順位を配列bに求める。ここで，配列の要素番号は1から始まる。

例

	1	2	3	4	5	6
a	30	55	20	30	40	60

のとき，

	1	2	3	4	5	6
b	4	2	6	4	3	1

※4位のデータ30が二つあるため，20の順位は6となる。

① 順位の初期値を1とする。

② 配列aの先頭から順に，一つ次の要素以降と大小を比較し，小さい方の順位を一つ下げる。

```
○ order(整数型の配列：a, 整数型の配列：b, 整数型：n)
  整数型：i , j
  for（i を1からnまで1ずつ増やす）
    b[i] ← 1                          /* 順位の初期値を1とする。*/
  endfor
  for（i を1からn − 1まで1ずつ増やす）
    for（j を i + 1からnまで1ずつ増やす）
      if (a[i]がa[j]より大きい)
        (1)
      else
        if (a[i]がa[j]より小さい)
          (2)
        endif
      endif
    endfor
  endfor
```

解答群　ア　b[i] ← b[i] − 1　　イ　b[i] ← b[i] + 1
　　ウ　b[j] ← b[j] − 1　　エ　b[j] ← b[j] + 1

練習 27

解答➡ P.140

標準学習時間 5min

n次多項式

$$f(x)=a_nx^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0$$

の値を求める。その際，上式を

$$f(x)=(\cdots((a_nx+a_{n-1})x+a_{n-2})x+\cdots+a_1)x+a_0$$

と変形し，内側の括弧内から計算を繰り返す。

ここで，配列の要素番号は0から始まり，係数a_0～a_nの値は，配列aのa[0]～a[n]に格納されているものとする。

○ 実数型：polynomial(実数型の配列：a, 実数型：x, 整数型：n)
　整数型：i
　実数型：y
　y ← a[n]
　i ← n − 1
　while（i が0以上）
　　y ← (1)
　　i ← i − 1
　endwhile
　return y

解答群　ア　x + a[i]　　イ　y × a[i] × x
　　ウ　y × x + a[i]　　エ　y × a[i] + x

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)イ (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)ウ
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

2 配列

練習 21

問題➡ P.34

解答 (1)ア　i　(2)キ　n ← n ÷ 2

<考え方>
例えば、10進数157を8桁の2進数10011101に変換して、下位桁から順に配列nishin[1]～nishin[8]に格納する際の処理イメージは、次のようになる。

被除数		除数		商	剰余		
157	÷	2	=	78	1	➡	nishin[1]へ格納
78	÷	2	=	39	0	➡	nishin[2]へ格納
39	÷	2	=	19	1	➡	nishin[3]へ格納
19	÷	2	=	9	1	➡	nishin[4]へ格納
9	÷	2	=	4	1	➡	nishin[5]へ格納
4	÷	2	=	2	0	➡	nishin[6]へ格納
2	÷	2	=	1	0	➡	nishin[7]へ格納
1	÷	2	=	0	1	➡	nishin[8]へ格納

(1)　剰余関数modを用いて求めた剰余は、求めた順にnishin[1]、nishin[2]、…、nishin[8]に格納する。したがって、1から8まで1ずつ変化するiをそのまま添字に用いればよい。

(2)　剰余は剰余関数modによって直接求めるため、上記のような除算は行っていない。次回の計算では、今回の除算の商が変数nに格納されていなければならないので、n÷2の除算を実行し、その商をnに格納しておく。

練習 22

問題➡ P.34

解答 (1)エ　9 − i　(2)キ　n ← n ÷ 2

<考え方>
設問の内容は前問と同じだが、変換結果の格納の仕方が前問とは逆になる。つまり、最初に求まる剰余をnishin[8]に格納し、最後に求まる剰余をnishin[1]に格納する。

(1)　剰余関数modを用いて求めた剰余を、nishin[8]、nishin[7]、…、nishin[1]に格納するためには、1から8まで1ずつ変化するiに対し、9− iを添字とすればよい。

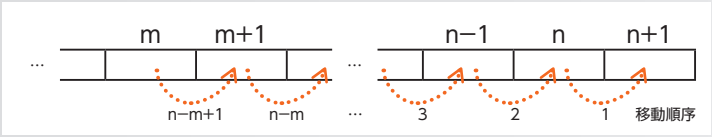
(2)　剰余は剰余関数modによって直接求めるため、上記のような除算は行っていない。次回の計算では、今回の除算の商が変数nに格納されていなければならないので、n÷2の除算を実行し、その商をnに格納しておく。

練習 23

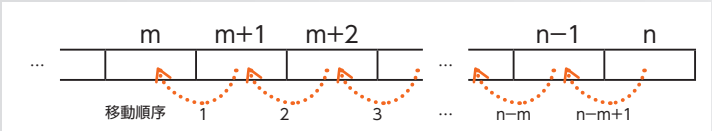
問題➡ P.35

解答 (1)ウ　a[i＋1] ← a[i]　(2)エ　a[i] ← a[i＋1]

<考え方>
配列要素を一つ後ろ、又は一つ前にずらす処理は、ずらす順序を誤ると、要素の上書きが起こり、配列内容を壊してしまう。
配列の要素番号m～nの値をそれぞれ一つ後ろにずらす処理は、後ろから前に向かって行う必要がある。



一方、配列の要素番号m～nの値をそれぞれ一つ前にずらす処理は、前から後ろに向かって行う必要がある。



(1)　最初の移動が「a[n ＋ 1] ← a[n]」、最後の移動が「a[m ＋ 1] ← a[m]」であり、変数 i はnからmまで変化するので、「a[i ＋ 1] ← a[i]」とすればよいことが分かる。
(2)　最初の移動が「a[m] ← a[m ＋ 1]」、最後の移動が「a[n － 1] ← a[n]」であり、変数 i はmからn－1まで変化するので、「a[i] ← a[i ＋ 1]」とすればよいことが分かる。

練習 24

問題➡ P.36

解答 (1)ウ　a[j － 1] ← a[j]　(2)カ　a[n] ← t

<考え方>
配列要素のローテート（巡回シフト）は、配列内容を前方、又は後方に指定要素数分ずらす際、端からはみ出す要素が反対側の端から挿入される処理である。
本問では左方向にローテートするため、端からはみ出す要素はその時点の先頭要素a[1]であり、1回のローテートごとにa[1]を退避しておき、a[2]からa[n]を順次一つ前にずらした後、退避しておいた要素をa[n]に挿入、という処理をm回繰り返す。

(1)　空欄(1)を含む内ループでは、a[2]からa[n]を順次一つ前にずらす処理を行う。つまり、
　　a[1]←a[2]、a[2]←a[3]、a[3]←a[4]、…、a[n-1]←a[n]
のような代入処理をこの順番で行う。これは、2からnまで1ずつ変化する j を用いて
　　a[j -1]←a[j]と表すことができる。
(2)　要素a[2]～a[n]のそれぞれを1要素分左側にずらした後、変数tに退避しておいた元のa[1]を、空きとなった右端の要素a[n]に格納することで、1回分のローテートが完了する。

練習 25

問題➡ P.37

解答 (1)ウ　a[sp] ← x
sp ← sp + 1
(2)オ　sp ← sp − 1
return a[sp]

<考え方>
配列を用いてスタックに対する二つの操作(pushとpop)をシミュレートする際、スタックの先頭を示す「スタックポインタ」の働きをする変数の初期値に注意する必要がある。本問ではspがそのための変数であり、その初期は1であるが、初期値が0のこともあり得る。

(1)　スタックポインタを示す変数spの初期値が1であるので、最初のデータがa[1]に格納されるようにするためには、a[sp]に格納後にspに1を加える必要がある。
(2)　変数spは、配列内の末尾のデータの次の位置を指している。つまり、a[sp]にはデータが格納されていない。そのため、取り出す際には、まずspから1を引いて末尾データに位置づけ、その位置のデータを返す必要がある。

練習 26

問題➡ P.38

解答 (1)エ　b[j] ← b[j] + 1　(2)イ　b[i] ← b[i] + 1

<考え方>
配列aの要素に対して降順に順位付けするために、本問では「順位の初期値を1位として、要素の大小を比較し、小さい方の順位を下げる」という方法を用いているが、「順位の初期値を最下位(要素数)として、要素の大小を比較し、大きい方の順位を上げる」という方法もある。ただし、この方法では、等しかった場合には双方の順位を上げる必要がある。

(1)　「a[i] > a[j]」が成り立つ場合、順位b[j]に1を加えることにより、要素番号 j の値の順位を一つ下げる。
(2)　「a[i] < a[j]」が成り立つ場合、順位b[i]に1を加えることにより、要素番号 i の値の順位を一つ下げる。

練習 27

問題➡ P.39

解答 (1)ウ　y × x + a[i]

<考え方>
n次多項式の値を、問題文で指定された方法に従って計算する。各項の係数の値は配列aに格納されているものを利用する。
a_nに相当するa[n]が変数yの初期値に設定され、変数 i には初期値n−1が設定される。したがって、1回目に1番内側のカッコ内を計算するためには、「y × x + a[i]」という計算を行い、その結果をyに格納すればよい。これを i を0まで1ずつ減らしながら繰り返すことで、