

科目B 直前暗記シート

基本情報技術者 科目B / 式を見た瞬間に「これは何か」が決まるものだけ

| 00 試験がはじまる直前に読む10行 |                       |
|--------------------|-----------------------|
| 配列の番号              | 1から始まる（0からではない）       |
| ÷2の商               | 二分探索の合図（並べ替えではない）     |
| a[j] と a[j+1]      | バブルソート（となりどうしを比べている）  |
| min が出てくる          | 選択ソート（最小の位置を覚えている）    |
| 内側が while + tmp    | 挿入ソート（退避してずらしている）     |
| 交換の不等号が >          | 小さい順に並ぶ／< なら大きい順      |
| スタック               | 逆順で出る／キューはそのまま        |
| 再帰                 | 小さいほうから表で積む。上から展開しない  |
| セキュリティ             | 迷ったら最小権限。「教育するだけ」は不正解 |
| 空欄が2つ              | やさしいほうだけ考える。片方で半分消える  |

| 01 計算の記号 <span>意味が取れないと詰む</span>                                                      |                     |              |
|---------------------------------------------------------------------------------------|---------------------|--------------|
| 書き方                                                                                   | 意味                  | 例            |
| ÷ n の商                                                                                | 割り算の答えの整数部分。端数は切り捨て | 7 ÷ 2 の商 = 3 |
| mod n                                                                                 | 割った余り               | 13 mod 3 = 1 |
| ∧                                                                                     | ビットの論理積。両方1のときだけ1   | —            |
| ∨                                                                                     | ビットの論理和。どちらか1なら1    | —            |
| ▽                                                                                     | ビットの排他的論理和。ちがえば1    | —            |
| >> <<                                                                                 | 右にずらす／左にずらす         | —            |
| ここで落とす ÷ 2 と + 2 の商 は別物です。「の商」と書いてあれば必ず切り捨て。<br>7 ÷ 2 の商 = 3、そのあと × 2 しても 6。7には戻りません。 |                     |              |

ビットで実際に使う形は2つだけ

| 式                                                                                                      | やっていること     | 例                   |
|--------------------------------------------------------------------------------------------------------|-------------|---------------------|
| x ∧ 00000001                                                                                           | 右端の1桁だけ取り出す | 01001011 → 00000001 |
| x ▽ 11111111                                                                                           | 全部の桁をひっくり返す | 01001011 → 10110100 |
| 見分け方 「▽ 11111111」のあとに「+ 00000001」が続いていたら、それは「足すと0になる数」を作っています。<br>(2の補数。ひっくり返して1を足す、という手順そのものを覚えてください) |             |                     |

while と do～while のちがいは

| 書き方                   | 条件を見るのは | 最低の実行回数 |
|-----------------------|---------|---------|
| while (条件) ~ endwhile | 中に入る前   | 0回もありうる |
| do ~ while (条件)       | 中を出たあと  | 必ず1回は通る |

| 02 見た瞬間に決まる対応表                                                                                                 |                 |
|----------------------------------------------------------------------------------------------------------------|-----------------|
| この書き方が出たら                                                                                                      | やっていること         |
| mid ← (lo + hi) ÷ 2 の商                                                                                         | 二分探索（整列済みが前提）   |
| n ← n ÷ 2 の商 をくり返す                                                                                             | 2進数に変換／半分ずつ減らす  |
| mod 10 と ÷ 10 の商 が並ぶ                                                                                           | 数を1桁ずつバラす       |
| mod 2 が 0                                                                                                      | 偶数かどうかの判定       |
| a[要素数 - i + 1]                                                                                                 | 逆順にしている         |
| t←a a←b b←t                                                                                                    | 入れ替え（中身を追う必要なし） |
| out[要素数] を取って足す                                                                                                | 前の合計に足していく（累計）  |
| pow(なにか, 0.5)                                                                                                  | 平方根（ルート）        |
| 見分け方 「0.5乗＝ルート」だけ覚えれば、平方根の問題は全部解けます。理由は考えなくて大丈夫です。<br>ルートの中身が足し算なら、先に足してから外側を0.5乗。 pow(pow(x,2)+pow(y,2), 0.5) |                 |

「元に戻る」のは、この式が2回出てきたとき

同じ操作を2回すると打ち消し合う——という形が決まっています。下の4つだけです。

| 2回出てきたら元に戻る式                            | なぜ           |
|-----------------------------------------|--------------|
| new[i] ← a[aの要素数 - i + 1]               | 逆順の逆順だから     |
| t←a[p] a[p]←a[q] a[q]←t<br>(同じ p と q で) | 入れ替えの入れ替えだから |

|                               |                        |
|-------------------------------|------------------------|
| x ← x ▽ y<br>(同じ y で)         | ▽ は同じ相手と2回で打ち消す        |
| x ← x + n のあと x ← x - n       | 足して引いているから             |
| ここで落とす 似ているが戻らない式——ここが引っかけです。 |                        |
| x ← x ÷ 2 の商 → x ← x × 2      | 切り捨てで消えた分は復活しない（7→3→6） |
| x ← x << 1 → x ← x >> 1       | 左端の桁が落ちて消える            |

これで取れる 入れ物での「元に戻る」も式ではなく回数で決まります。  
キューは1往復で元のまま／スタックは1往復で逆順、2往復で元通り。

| 03 式だけで並べ替えを見分ける <span>最重要</span> |  |
|-----------------------------------|--|
|-----------------------------------|--|

問題文に「バブルソート」と書いてくれないことがあります。式で形が決まります。

| 3秒で見分ける表                                                                         |                         |                       |                     |
|----------------------------------------------------------------------------------|-------------------------|-----------------------|---------------------|
| ここを見る                                                                            | バブル                     | 選択                    | 挿入                  |
| 比べている相手                                                                          | a[j] と a[j+1]<br>となりどうし | a[j] と a[min]<br>最小候補 | a[j] と tmp<br>退避した値 |
| 内側のループ                                                                           | for                     | for                   | while               |
| 目印になる変数                                                                          | なし                      | min（位置を覚える）           | tmp（1個だけ退避）         |
| 入れ替える場所                                                                          | 内側ループの中<br>何度も交換        | 内側ループの外<br>1周に1回だけ    | ずらして最後に差し込む         |
| 1周で確定するのは                                                                        | 右端                      | 左端                    | 手前が並んでいく            |
| 見分け方 まず「内側が while かどうか」を見る。while なら挿入。<br>for なら、min という変数があるかを見る。あれば選択、なければバブル。 |                         |                       |                     |

バブルソート（昇順）

```
for (i を 1 から aの要素数 - 1 まで 1 ずつ増やす)
  for (j を 1 から aの要素数 - i まで 1 ずつ増やす)
    if (a[j] > a[j + 1])      ← となりどうし
      t ← a[j]
      a[j] ← a[j + 1]
      a[j + 1] ← t          ← 中で何度も交換
    endif
  endfor
endfor
```

選択ソート（昇順）

```
for (i を 1 から aの要素数 - 1 まで 1 ずつ増やす)
  min ← i                    ← 位置を覚える変数
  for (j を i + 1 から aの要素数 まで 1 ずつ増やす)
    if (a[j] < a[min])        ← 最小候補と比べる
      min ← j
    endif
  endfor
  a[i] と a[min] を入れ替える ← 外で1回だけ交換
endfor
```

挿入ソート（昇順）

```
for (i を 2 から aの要素数 まで 1 ずつ増やす)
  tmp ← a[i]                 ← 1個だけ退避
  j ← i - 1
  while (j ≥ 1 and a[j] > tmp) ← 内側が while
    a[j + 1] ← a[j]          ← 後ろへずらす
    j ← j - 1
  endwhile
  a[j + 1] ← tmp              ← 最後に差し込む
endfor
```

1周した時点の中身（{5, 3, 8, 1} を昇順に）

| 方式  | 1周後        | 確定した場所 |
|-----|------------|--------|
| バブル | 3, 5, 1, 8 | 右端の 8  |
| 選択  | 1, 3, 8, 5 | 左端の 1  |

ここで落とす 途中経過を答える問題では、確定する側が逆なのが決め手です。  
バブルは右から、選択は左から埋まっていきます。

昇順か降順かは不等号だけで決まる

| 交換の条件 | 結果 |
|-------|----|
|-------|----|

|                       |          |
|-----------------------|----------|
| if (a[j] > a[j+1]) 交換 | 小さい順（昇順） |
| if (a[j] < a[j+1]) 交換 | 大きい順（降順） |

見分け方 大きいものを後ろに送れば、小さい順に並ぶ。だから > なら昇順。

スタックとキュー

| 入れ物  | 1,2,3 の順に入れて全部出すと | ひとことで |
|------|-------------------|-------|
| スタック | 3, 2, 1           | 逆順になる |
| キュー  | 1, 2, 3           | そのまま  |

| 名前   | 式の目印                 | 条件   |
|------|----------------------|------|
| 線形探索 | for で先頭から1つつ         | 整列不要 |
| 二分探索 | mid ← (lo+hi) ÷ 2 の商 | 整列必須 |

ここで落とす 問題文に「昇順に整列されている」と書いてあるか必ず確認。書いていなければ、二分探索を前提にした選択肢はすべて誤りです。

04 再帰と、木をたどる順番

再帰は「下から積む」

止まる条件を先に見つけて、小さい値から順に表を作る。これだけです。

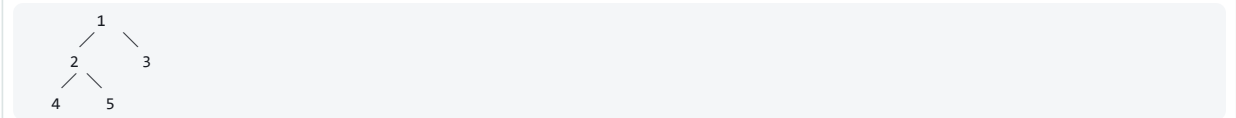
```
f(0) = 1          ← 止まる条件
f(1) = 1 × f(0) = 1
f(2) = 2 × f(1) = 2
f(3) = 3 × f(2) = 6
f(4) = 4 × f(3) = 24
```

ここで落とす 上から f(4) = 4 × (3 × f(2)) と展開すると、かっこが増えて必ず迷子になります。

| 式                      | 正体         | 値の並び                 |
|------------------------|------------|----------------------|
| f(n) = n × f(n-1)      | 階乗         | 1, 2, 6, 24, 120     |
| f(n) = f(n-1) + f(n-2) | フィボナッチ     | 1, 1, 2, 3, 5, 8, 13 |
| f(n) = f(n-1) + n      | 1からnまでの合計  | 1, 3, 6, 10, 15      |
| 自分を2回呼ぶ                | 木をたどる／分割統治 | 枝分かれする               |

「木をたどる」とは何か

木とは、上から下へ枝分かれしていく形にデータを並べたものです。  
科目Bでは、その全部の点を、という順番で読み上げるかが問われます。



問題はこの形で出ます。自分自身を呼び出す（再帰）関数です。

```
order(n):
  if (子が2つある)
    order(左の子)
    n を出力          ← この位置がすべてを決める
    order(右の子)
  else
    n を出力
endif
```

| 「n を出力」の位置 | たどる順番      | 上の木だと         |
|------------|------------|---------------|
| いちばん先頭     | 自分 → 左 → 右 | 1, 2, 4, 5, 3 |
| 真ん中（左と右の間） | 左 → 自分 → 右 | 4, 2, 5, 1, 3 |
| いちばん最後     | 左 → 右 → 自分 | 4, 5, 2, 3, 1 |

これで取れる 選択肢の「1つ目の数字」だけ見れば半分以上消えます。  
出力が先頭なら、最初に出るのはいちばん上（根）。  
出力が真ん中か最後なら、最初に出るのはいちばん左下。  
木の絵を全部たどる必要はありません。

05 セキュリティ 4問=200点

科目Bの20問のうち4問。アルゴリズムを解かなくても知識だけで取り切れる、いちばん割のいい場所です。

選択肢を切る4つのものさし

|   |                                 |
|---|---------------------------------|
| ① | 根本的か、その場しのぎか → 根本的なほうが正解        |
| ② | 技術で防げているか、人の注意頼みか → 技術のほうが正解    |
| ③ | その脅威に本当に効くか → ずれた対策が必ず混ぜてある     |
| ④ | やりすぎて業務が止まらないか → 「全面禁止」は基本的に不正解 |

ここで落とす 「教育する」「注意喚起する」だけの選択肢は、たいてい不正解。  
人の注意力に頼る対策は確実性がないためです。技術的に防ぐ選択肢があれば、そちらが正解になります。

これで取れる 迷ったら「最小権限の原則」に近いものを選ぶ。  
「必要な人に、必要な範囲だけ、必要な期間だけ」——この考えに近い選択肢が正解になることが非常に多いです。

そのまま答えになる10行

| 状況            | 正解の方向                           |
|---------------|---------------------------------|
| ノートPCやUSBの紛失  | 暗号化しておく（パスワードだけでは足りない）          |
| マルウェアに感染した    | まずネットワークから切り離す                  |
| ランサムウェア対策     | バックアップを切り離して保管（つなぎっぱなしは一緒にやられる） |
| 社長から急ぎの送金指示   | 電話など別の経路で本人に確認                  |
| 退職者のアカウント     | 退職手続に削除を組み込む（誓約書ではない）           |
| 共有フォルダが全員閲覧可  | 必要な人だけに権限を絞る                    |
| 1人で申請から実行まで   | 職務分掌で申請者と承認者を分ける                |
| 外部委託・クラウド利用   | 責任は移らない。監督義務が残る                 |
| 公開Webサーバの置き場所 | DMZ（社内LANに置かない）                 |
| インシデントが起きた    | 検知 → 初動（止める）→ 復旧 → 原因究明         |

ここで落とす 感染したPCの電源をいきなり切るのは不正解。メモリに残る調査の手がかりが消えてしまいます。  
正しい初動は「LANケーブルを抜く・無線を切る」です。

防御の道具（何が防げて、何が防げないか）

| 道具        | 役割                            |
|-----------|-------------------------------|
| ファイアウォール  | IPとポートで通信を遮断（Webアプリ攻撃は防げない）   |
| WAF       | 通信の中身を見てWebアプリ攻撃を防ぐ           |
| IDS / IPS | 検知だけがIDS／遮断までするのがIPS          |
| 暗号化       | 盗み見だけを防ぐ。改ざん・なりすまし・SQL注入は防げない |
| 多要素認証     | パスワードが漏れても侵入を防げる              |

06 解き方の型

長文は上から順に読まない

|   |                              |
|---|------------------------------|
| ① | 設問を先に読む——何を答えるのか（値か、式か、組合せか） |
| ② | 1行目だけ読む——この関数が何をやるものか。途中は飛ばす |
| ③ | 「例えば」を探す——あれば数字を紙に書き写す       |
| ④ | プログラムは空欄の前後3行だけ見る            |
| ⑤ | 選択肢を具体例に当てはめて確かめる            |

これで取れる 「例えば」は出題者が用意した検算用のデータです。  
選択肢を1つつ当てはめて、その例の答えになるものを選べば、プログラムを全部理解しなくても正解できます。

空欄の場所で、見るところが変わる

| 空欄の場所           | やること                             |
|-----------------|----------------------------------|
| while / for の条件 | 具体例の数字を入れて2周だけ回す。ちょうど0になる場合を必ず試す |
| return の式       | 具体例の答えは分かっているので、代入して合うものを選ぶ      |
| 配列への代入          | 表を書いて1～2周まわす                     |
| a と b の2か所      | やさしいほうだけ考える。片方決まれば選択肢が半分以下       |

ここで落とす プログラムを最初から最後まで理解しようとしな。空欄の周りだけ見て、具体例で確かめる——これが正しい戦い方です。