Benesse模試_配列解説

3年情報

問1　ア

を二次元配列として格納することができる。二次元配列の名前を Kekka とし，行の添字を変数 tate，列の添字を変数 yoko とすると，格納された要素は Kekka[tate][yoko] として表され，例えば，添字 1 の行（tate = 1）かつ添字 4 の列（yoko = 4）の要素は，Kekka[1][4] で表すことができる。

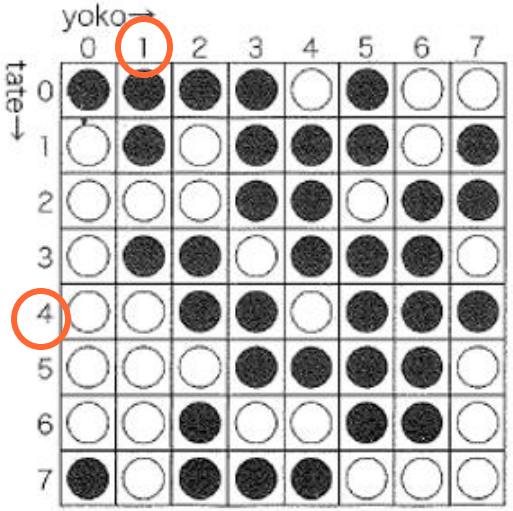


図3　ゲーム終了時の盤面例

●kekka[4][1]は

tateの4、yokoの1を見る

答え　ア　1

問1 イ、ウ

を二次元配列として格納することができる。二次元配列の名前を Kekka とし，行の添字を変数 tate，列の添字を変数 yoko とすると，格納された要素は Kekka[tate][yoko] として表され，例えば，添字 1 の行（tate = 1）かつ添字 4 の列（yoko = 4）の要素は，Kekka[1][4] で表すことができる。

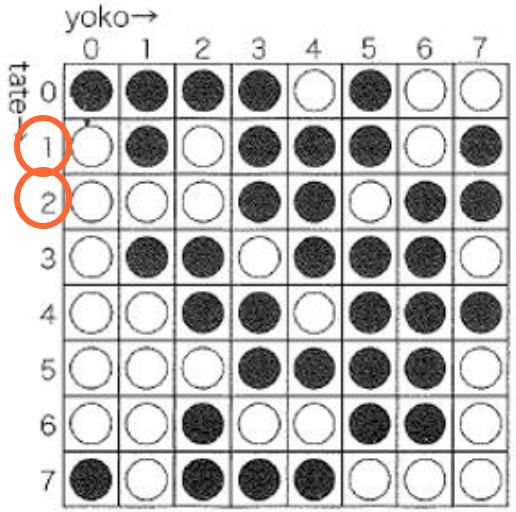


図3　ゲーム終了時の盤面例

●イは１の行の白の個数を数える

答え　イ　３

●ウは２の行の黒の個数を数える

答え　ウ　４

問２　エ

を二次元配列として格納することができる。二次元配列の名前を Kekka とし，行の添字を変数 tate，列の添字を変数 yoko とすると，格納された要素は Kekka[tate][yoko] として表され，例えば，添字１の行（tate = 1）かつ添字 4 の列（yoko = 4）の要素は，Kekka[1][4] で表すことができる。

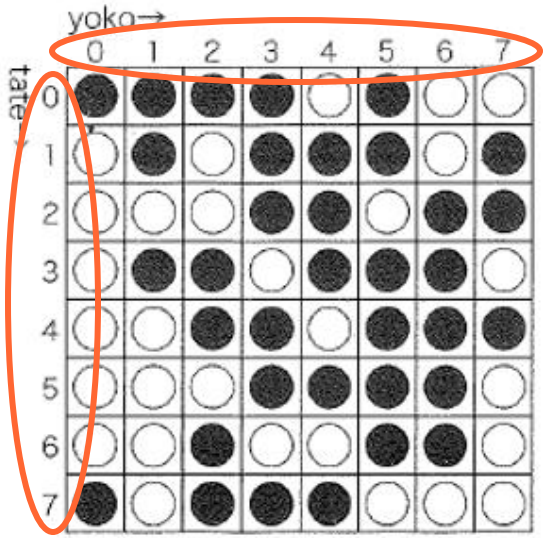


図３　ゲーム終了時の盤面例

●tateもyokoも0から始まり７まで繰り返す

答え　エ　７

問２　オ


```
(01) Kekka = [[ 0, 0, 0, 0, 1, 0, 1, 1 ], (以下, 省略) … ]
(02) shiro = 0
(03) kuro = 0
(04) i = 0
(05) tate を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(06)     yoko を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(07)         shiro = [オ]
(08)         i = i + 1
(09) kuro = [カ] - [キ]
```

図４　プログラム１

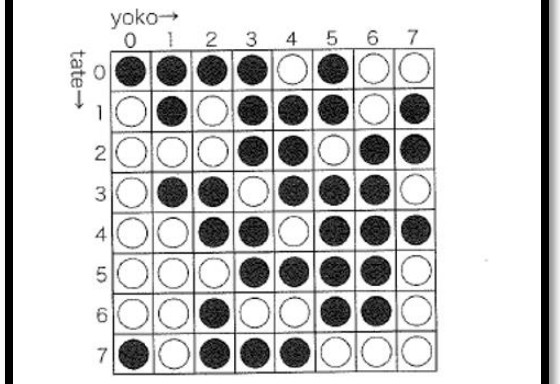


図３　ゲーム終了時の盤面例（再掲）

●shiroの初期値は0

●kekkaに収納されている

●kekka[tate][yoko]は
kekka[0][0]からスタート

●黒だったら０が足され
白なら１が足される

答え　オ　②

問２　カ、キ


```
(01) Kekka = [[ 0, 0, 0, 0, 1, 0, 1, 1 ], (以下, 省略) … ]
(02) shiro = 0
(03) kuro = 0
(04) i = 0
(05) tate を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(06)     yoko を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(07)         shiro = [オ]
(08)         i = i + 1
(09) kuro = [カ] - [キ]
```

図４　プログラム１

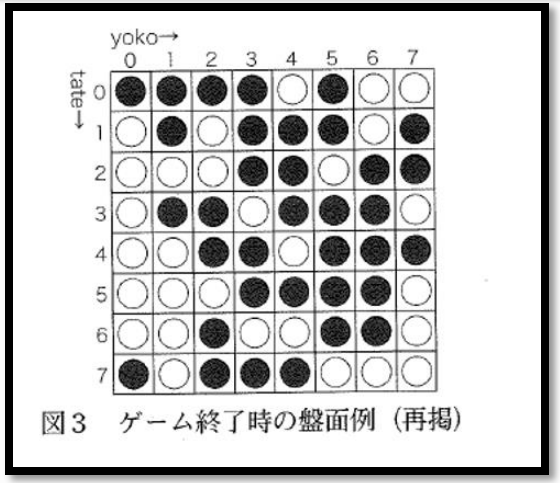


図３　ゲーム終了時の盤面例（再掲）

●全体の回数ー白の個数で求める

●白の個数は変数shiroにある

答え　　キ　①

●全体の個数はiを追っていく

答え　　カ　③

問２　ク、ケ


```
(01) Kekka = [[ 0, 0, 0, 0, 1, 0, 1, 1 ], (以下, 省略) … ]
(02) shiro = 0
(03) kuro = 0
(04) i = 0
(05) tate を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(06)     yoko を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(07)         shiro = [オ]
(08)         i = i + 1
(09) kuro = [カ] - [キ]
(10) もし [コ] ならば：
(11)     「黒の勝ち」と表示する
(12) そうでなくてもし [サ] ならば：
(13)     「引き分け」と表示する
(14) そうでなければ：
(15)     「白の勝ち」と表示する
(16) 二次元配列 Kekka のすべての要素を 0 として初期化する
```

図５　プログラム２

●ｉは全体の個数である

●０を含むことに注意

答え　クケ　６４

問３コ、サ


```
(01) Kekka = [[ 0, 0, 0, 0, 1, 0, 1, 1 ], (以下, 省略) … ]
(02) shiro = 0
(03) kuro = 0
(04) i = 0
(05) tate を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(06)     yoko を 0 から [エ] まで 1 ずつ増やしながら繰り返す：
(07)         shiro = [オ]
(08)         i = i + 1
(09) kuro = [カ] - [キ]
(10) もし [コ] ならば：
(11)     「黒の勝ち」と表示する
(12) そうでなくてもし [サ] ならば：
(13)     「引き分け」と表示する
(14) そうでなければ：
(15)     「白の勝ち」と表示する
(16) 二次元配列 Kekka のすべての要素を 0 として初期化する
```

図５　プログラム２

●黒が勝つケースはkuroが多いとき

●引き分けるケースはkuroとshiroが同じ個数の時

答え　コ　０
　　サ　②

●全体の個数は64ある。
　黒を求める計算式は（全体の個数）－（白の数）

●もしすべてのマスが埋まらずに白の数が30、黒の数も29で終わったと考えると黒の数は64-30=34になるので黒の勝ちになってしまう。

答え　シ　②

問1 イ

まずAさんは配列を利用し，クラス名を Kurasumei，獲得点を Tensu として，表1のデータを格納することにした。また，全体のクラス数の値を変数 kurasu_num に格納した。このようにして図1のプログラムを書いたAさんは，想定したデータが正しく出力されるか試してみることにした。その結果，Tensu[4] は［ア］と予想どおりに表示された。

なお，すべての配列の添字は0から始まっているものとする。例えば，Kurasumei[0] の値は "1A" である。

```
(1)  Kurasumei = ["1A", "1B", "1C", "1D", "2A", "2B", "2C",
                  "2D", "3A", "3B", "3C", "3D"]
(2)  Tensu = [42, 24, 13, 27, 11, 49, 65, 67, 54, 60, 65, 3]
(3)  kurasu_num = [イ]
```

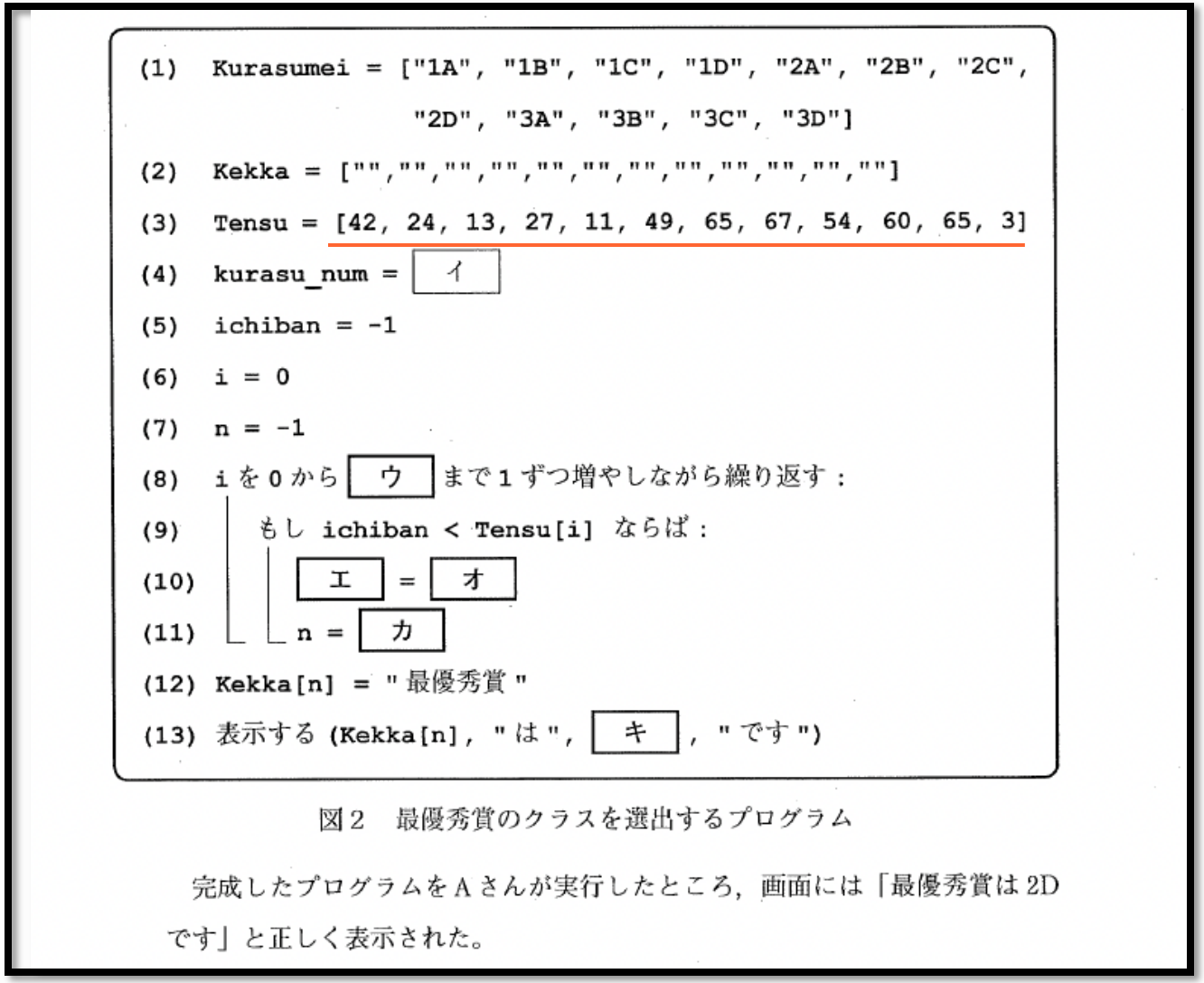
図1　データを配列と変数に格納するプログラム

［ア］・［イ］の解答群

⓪ 3　① 4　② 11　③ 12　④ 13　⑤ 27　⑥ 49

●kurasu_numには全体のクラス数が入る

答え　イ　③



```
(1)  Kurasumei = ["1A", "1B", "1C", "1D", "2A", "2B", "2C",
                  "2D", "3A", "3B", "3C", "3D"]
(2)  Kekka = ["","","","","","","","","","","",""]
(3)  Tensu = [42, 24, 13, 27, 11, 49, 65, 67, 54, 60, 65, 3]
(4)  kurasu_num = [ イ ]
(5)  ichiban = -1
(6)  i = 0
(7)  n = -1
(8)  i を 0 から [ ウ ] まで 1 ずつ増やしながら繰り返す：
(9)      もし ichiban < Tensu[i] ならば：
(10)         [ エ ] = [ オ ]
(11)         n = [ カ ]
(12) Kekka[n] = "最優秀賞"
(13) 表示する (Kekka[n], "は", [ キ ], "です")
```

図2　最優秀賞のクラスを選出するプログラム

完成したプログラムをAさんが実行したところ，画面には「最優秀賞は2Dです」と正しく表示された。

●繰り返す回数は0から11まで

●全体のクラス数が12なので-1

答え　③kurasu_num-1

```
(1)  Kurasumei = ["1A", "1B", "1C", "1D", "2A", "2B", "2C",
                  "2D", "3A", "3B", "3C", "3D"]
(2)  Kekka = ["","","","","","","","","","","",""]
(3)  Tensu = [42, 24, 13, 27, 11, 49, 65, 67, 54, 60, 65, 3]
(4)  kurasu_num = [ イ ]
(5)  ichiban = -1
(6)  i = 0
(7)  n = -1
(8)  i を 0 から [ ウ ] まで 1 ずつ増やしながら繰り返す：
(9)      もし ichiban < Tensu[i] ならば：
(10)         [ エ ] = [ オ ]
(11)         n = [ カ ]
(12) Kekka[n] = "最優秀賞"
(13) 表示する (Kekka[n], "は", [ キ ], "です")
```

図2　最優秀賞のクラスを選出するプログラム

完成したプログラムをAさんが実行したところ，画面には「最優秀賞は2Dです」と正しく表示された。

●Tensu[i]を順番に見ていき
ichibanより大きければその値を
ichibanに入れる

●その際その地点で得点が大きい
iをnに入れる

答え　エ　0
　　オ　Tensu[i]

答え　カ　i

```
(1)  Kurasumei = ["1A", "1B", "1C", "1D", "2A", "2B", "2C",
                  "2D", "3A", "3B", "3C", "3D"]
(2)  Kekka = ["","","","","","","","","","","",""]
(3)  Tensu = [42, 24, 13, 27, 11, 49, 65, 67, 54, 60, 65, 3]
(4)  kurasu_num = [ イ ]
(5)  ichiban = -1
(6)  i = 0
(7)  n = -1
(8)  i を 0 から [ ウ ] まで 1 ずつ増やしながら繰り返す：
(9)      もし ichiban < Tensu[i] ならば：
(10)         [ エ ] = [ オ ]
(11)         n = [ カ ]
(12) Kekka[n] = "最優秀賞"
(13) 表示する (Kekka[n], "は", [ キ ], "です")
```

図２　最優秀賞のクラスを選出するプログラム

完成したプログラムをＡさんが実行したところ，画面には「最優秀賞は 2D です」と正しく表示された。

●一番得点が大きい添字番号がnに入っている

●それをクラスに当てはめる

答え

キ　①kurasumei[n]

問3 ク・ケ・コ

●やりたいことは一旦全てのクラスに敢闘賞を当てはめる

0　　1　　2・・

●Kekka=[“敢闘賞”、“敢闘賞”、“敢闘賞”]

●その過程でもしichiban<tensu[i]に当てはまるクラスがあれば
最優秀賞に書き換える

問3 ク・ケ・コ

●まずIAは42点でichiban>-1より大きいので最優秀賞が代入される

0　　1　　2・・

●Kekka=[“最優秀賞”、“敢闘賞”、“敢闘賞”]

●次に42点を超えてくるのは49点の2Bなので最優秀賞が代入される
このときIAの最優秀賞が消えて、代わりに敢闘賞が代入されなければいけないがそのようなプログラムの記述はないのでIAの最優秀賞は残る

●同じように65点の2Cは49点の2Bを超えるが2Cの最優秀賞は残る

答え　0 IA ⑤2B ⑥2C