平成28年秋 データ構造及びアルゴリズム 数値の編集

問8　数値の編集（データ構造及びアルゴリズム）　(H28秋-FE 午後問8)

【解答】

［設問1］　a－イ，b－ア，c－エ
［設問2］　d－イ，e－エ
［設問3］　f－エ

【解説】

指定された編集パターンに従って，数値を編集するプログラムに関する問題である。設問は，数値の編集過程をトレース（追跡）し，編集結果を答える問題，プログラムの空欄を埋める問題，編集処理のケースを特定するものである。いずれも表2「現在の変数・配列要素の内容に応じた更新処理」の内容が理解できれば正解できる問題である。問8は，問題文を正確に理解すれば解答可能な内容になっているので，問題文を理解する力，処理手順やプログラムをトレースする力をしっかり身につけておきたい。

はじめに表2について確認する。表2は，配列Pattern[]，Value[]の要素の内容や変数signifの値に応じて，処理をケース分けしたものである。ケース1をみていく。

ケース1

・Pattern[p]が“▯”。
　添字pの値は0からLength（Pattern[]）（配列の文字列長）－1まで変化する。
・signifがoff。
・Value[v]が“0”。
　添字vの値は0から始まり，ケース1～7の処理を行った場合だけ1ずつ増加する。

このような条件が成り立つときは，次のようになる。

・Pattern[p]を fill 文字（Pattern[]中の先頭の文字）で更新する。
・signifはoffのまま変化しない。

なお，Value[v＋1]（Value[v]の次の要素）の網掛けは，内容を判定しないことを表しているので，ケース分けの条件には含まれない。

続いて，各プログラムを確認する。行番号とともに次に示す。

〔プログラム〕
(行番号)

```
 1  ○関数：Edit（文字型：Pattern[]，文字型：Value[]）
 2  ○文字型：fill
 3  ○論理型：signif
 4  ○整数型：p，v
 5  ・fill ← Pattern[0]
 6  ・signif ← off
 7  ・v ← 0
 8  p：0，p ＜ Length(Pattern[])，1 /* Length()は引数の文字列長を返す */
 9  Pattern[p] = ”▯” or Pattern[p] = ”▮”/* 表2のケース1～7の処理 */
10      現在の変数・配列要素の内容が，表2のケース1～7の
        どれに該当するかを決定し，そのケースに従って
        Pattern[p]とsignifの更新処理を行う。
11      ・v ← v ＋ 1
                                        /* 表2のケース8，9の処理 */
12      signif ＝ off
13      ・Pattern[p] ← fill
14
15
16
```

〔プログラム中の [------] 部分の処理〕
(行番号)

```
21  signif ＝ off
22  Pattern[p] ＝ ”▯” and Value[v] ＝ ”0”
23  /* 何もしない */
24      Value[v ＋ 1] ≠ ”+”
25      ・signif ← on
26
27
28      d
29      ・Pattern[p] ← fill
30      ・Pattern[p] ← Value[v]
31
32      e
33      ・signif ← off
34
35  ・Pattern[p] ← Value[v]
36
```

行番号5：変数fillに配列Pattern[]の先頭の文字を格納
行番号6：変数signifをoffに初期化
行番号7：配列Value[]の添字である変数vを0に初期化
行番号8～16：配列Pattern[]の編集パターンに従って，配列Value[]の数値を表す文字列を編集する処理
　行番号9：ケース分けの条件式
　＜「Pattern[p]＝”▯” or Pattern[p]＝”▮”」が真の場合（ケース1～7）＞
　　行番号10：[------] のプログラム（行番号21～36）を実行
　　行番号11：配列Value[]の添字vの値を1増やす
　＜「Pattern[p]＝”▯” or Pattern[p]＝”▮”」が偽の場合（ケース8，9）＞
　　行番号12：ケース分けの条件式
　　＜「signif＝off」の場合（ケース8）＞
　　　行番号13：Pattern[p]に変数fillの値を格納

行番号21：ケース分けの条件式
＜「signif＝off」が真の場合（ケース1～5）＞
　行番号22：ケース分けの条件式
　＜「Pattern[p]＝”▯” and Value[v]＝”0”」が真の場合（ケース1）＞
　　行番号23：処理なし
　＜「Pattern[p]＝”▯” and Value[v]＝”0”」が偽の場合（ケース2～5）＞
　　行番号24：ケース分けの条件式
　　＜「Value[v＋1]≠”+”」が真の場合（ケース2，4）＞
　　　行番号25：変数signifのoffをonに更新
　行番号28：ケース分けの条件式　……　空欄d
　＜空欄dが真の場合＞
　　行番号29：Pattern[p]に変数fillの値を格納
　＜空欄dが偽の場合＞
　　行番号30：Pattern[p]にValue[v]の値を格納
＜「signif＝off」が偽の場合（ケース6，7）＞
　行番号32：ケース分けの条件式　……　空欄e
　＜空欄eが真の場合＞
　　行番号33：変数signifのoffに更新
　行番号35：Pattern[p]にValue[v]の値を格納

［設問1］

与えられた配列Pattern[]及びValue[]を基に関数Editの実行結果を解答する。この時点では，プログラムは，表2のケース1～7の処理の部分が未完成であるため，この部分は表2の内容をトレースして実行結果を求めることになる。

トレースの際は，変数signifには初期値としてoffが格納されている（行番号6），Pattern[]の添字pは0からLength(Pattern[])－1まで変化し，繰返しの都度1ずつ増える（行番号8），Value[]の添字vは0から始まり（行番号7），ケース1～7の場合だけ1増える（行番号11）ことをプログラムで確認しておく。また，条件の組合せや，条件が成り立つ場合の更新内容を表2で正確に確認するよう心掛ける。

・空欄a：Pattern[]の内容“*▯▯,▯▯▯#”，Value[]の内容“00000+”
　まず，Pattern[0]は“*”であり，Pattern[p]は“▯”と“▮”以外の文字なので，ケースは8か9になる。signifには初期値としてoffが格納されており，Value[v]，Value[v＋1]は網掛けで判定の必要がないため，この二つの条件からケース8になることが分かる。ケース8の場合，Pattern[p]はfill文字（Pattern[]中の先頭の文字）で置き換えるので，Pattern[0]は“*”，また，signifはoffのまま次の処理に進む。次に，Pattern[1]は“▯”，signifはoffなので，ケースは1，4，5のいずれかになる。Value[v]の値を確認すると，Value[0]は“0”であり，これらの条件から，該当するケースは1になることが分かる。ケース1の場合，Pattern[p]はfill文字で置き換えるので，Pattern[1]は“*”となる。また，signifはoffのまま次の処理に進む。このとき，Value[]の添字vは1増えて1となる。続いて，Pattern[2]は“▯”，signifはoff，Value[1]は“0”なので，該当するケースは1になり，Pattern[2]は“*”，signifはoff，vは2となる。更にPattern[3]は“,”（“▯”と“▮”以外），signifはoffなので，該当するケースは8になり，Pattern[3]は“*”，signifはoffとなる。このように，Pattern[p]，signif，Value[v]，Value[v＋1]の内容を順にチェックして該当するケースを特定し，各ケースでPattern[p]とsignifの内容がどう更新されるのかを一つずつ確認していく。結果は次の表のようになり，（イ）の“********”が正解である。

現在の内容					更新後	
Pattern[p]	signif	Value[v]	Value[v＋1]	ケース	Pattern[ ]	signif
* (0)	off			8	*▯▯,▯▯▯#	off
▯ (1)	off	0 (0)		1	**▯,▯▯▯#	off
▯ (2)	off	0 (1)		1	***,▯▯▯#	off
, (3)	off			8	****▯▯▯#	off
▯ (4)	off	0 (2)		1	*****▯▯#	off
▯ (5)	off	0 (3)		1	******▯#	off
▯ (6)	off	0 (4)		1	*******#	off
# (7)	off			8	********	off

注記　（）内の数字は配列の添字を表す。
　　網掛け部分は，内容を判定しない。
　　更新後のPattern[]の下線は，処理対象となった文字位置を表す。

・空欄b：Pattern[]の内容“*▯▯▯.▯▯#”，Value[]の内容“00012-”
　空欄aと同様にPattern[]，signif，Value[v]，Value[v＋1]の内容にしたがってトレースすると，次の表のようになる。ケースが1，8の場合の処理は空欄aで確認済みなので，ここではケースが4，6，9になるPattern[5]，Pattern[6]，Pattern[7]の場合について確認する。
　まず，Pattern[5]の値は“▯”，signifの値はその前の処理の結果offである。そして，Value[v]（v＝3）は“1”なので，ケースは4か5になる。この場合，Value[v＋1]も判定する必要があるのでValue[4]を確認すると“2”であり，Value[v＋1]は“＋”以外なので，該当するケースは4になる。ケース4の場合，Pattern[p]はValue[v]で置き換えるので，Pattern[5]は“1”，signifはonに更新され，vは1増えて4になって次の処理へ進む。
　次に，Pattern[6]は“▯”，signifはon，Value[4]は“2”なので，ケースは6か7のいずれかになる。このとき，Value[v＋1]であるValue[5]は“－”なので，Value[v＋1]が“＋”以外のケース6が該当する。ケース6の場合，Pattern[p]はValue[v]で置き換えるためPattern[6]は“2”，signifはonのまま，vは5になって次の処理へ進む。
　続いて，Pattern[7]は“#”であり，“▯”と“▮”以外の文字なのでケースは8か9になる。signifがonであることから，該当するケースは9になる。ケース9の場合，Pattern[p]はそのまま残すためPattern[7]は“#”，signifはonになり，処理を終わる。したがって，（ア）の“*****12#”が正解である。

平成28年秋 データ構造及びアルゴリズム 数値の編集

問8　数値の編集（データ構造及びアルゴリズム）　(H28秋-FE 午後問8)

【解答】

［設問1］　a－イ，b－ア，c－エ
［設問2］　d－イ，e－エ
［設問3］　f－エ

【解説】

指定された編集パターンに従って，数値を編集するプログラムに関する問題である。設問は，数値の編集過程をトレース（追跡）し，編集結果を答える問題，プログラムの空欄を埋める問題，編集処理のケースを特定するものである。いずれも表2「現在の変数・配列要素の内容に応じた更新処理」の内容が理解できれば正解できる問題である。問8は，問題文を正確に理解すれば解答可能な内容になっているので，問題文を理解する力，処理手順やプログラムをトレースする力をしっかり身につけておきたい。

はじめに表2について確認する。表2は，配列Pattern[]，Value[]の要素の内容や変数signifの値に応じて，処理をケース分けしたものである。ケース1をみていく。

ケース1

・Pattern[p]が“▯”。
　添字pの値は0からLength（Pattern[]）（配列の文字列長）－1まで変化する。
・signifがoff。
・Value[v]が“0”。
　添字vの値は0から始まり，ケース1～7の処理を行った場合だけ1ずつ増加する。

このような条件が成り立つときは，次のようになる。

・Pattern[p]を fill 文字（Pattern[]中の先頭の文字）で更新する。
・signifはoffのまま変化しない。

なお，Value[v＋1]（Value[v]の次の要素）の網掛けは，内容を判定しないことを表しているので，ケース分けの条件には含まれない。

続いて，各プログラムを確認する。行番号とともに次に示す。

〔プログラム〕
(行番号)

```
 1  ○関数：Edit（文字型：Pattern[]，文字型：Value[]）
 2  ○文字型：fill
 3  ○論理型：signif
 4  ○整数型：p，v
 5  ・fill ← Pattern[0]
 6  ・signif ← off
 7  ・v ← 0
 8  p：0，p ＜ Length(Pattern[])，1 /* Length()は引数の文字列長を返す */
 9  Pattern[p] = ”▯” or Pattern[p] = ”▮”/* 表2のケース1～7の処理 */
10      現在の変数・配列要素の内容が，表2のケース1～7の
        どれに該当するかを決定し，そのケースに従って
        Pattern[p]とsignifの更新処理を行う。
11      ・v ← v ＋ 1
                                        /* 表2のケース8，9の処理 */
12      signif ＝ off
13      ・Pattern[p] ← fill
14
15
16
```

〔プログラム中の [------] 部分の処理〕
(行番号)

```
21  signif ＝ off
22  Pattern[p] ＝ ”▯” and Value[v] ＝ ”0”
23  /* 何もしない */
24      Value[v ＋ 1] ≠ ”+”
25      ・signif ← on
26
27
28      d
29      ・Pattern[p] ← fill
30      ・Pattern[p] ← Value[v]
31
32      e
33      ・signif ← off
34
35  ・Pattern[p] ← Value[v]
36
```

行番号5：変数fillに配列Pattern[]の先頭の文字を格納
行番号6：変数signifをoffに初期化
行番号7：配列Value[]の添字である変数vを0に初期化
行番号8～16：配列Pattern[]の編集パターンに従って，配列Value[]の数値を表す文字列を編集する処理
　行番号9：ケース分けの条件式
　＜「Pattern[p]＝”▯” or Pattern[p]＝”▮”」が真の場合（ケース1～7）＞
　　行番号10：[------] のプログラム（行番号21～36）を実行
　　行番号11：配列Value[]の添字vの値を1増やす
　＜「Pattern[p]＝”▯” or Pattern[p]＝”▮”」が偽の場合（ケース8，9）＞
　　行番号12：ケース分けの条件式
　　＜「signif＝off」の場合（ケース8）＞
　　　行番号13：Pattern[p]に変数fillの値を格納

行番号21：ケース分けの条件式
＜「signif＝off」が真の場合（ケース1～5）＞
　行番号22：ケース分けの条件式
　＜「Pattern[p]＝”▯” and Value[v]＝”0”」が真の場合（ケース1）＞
　　行番号23：処理なし
　＜「Pattern[p]＝”▯” and Value[v]＝”0”」が偽の場合（ケース2～5）＞
　　行番号24：ケース分けの条件式
　　＜「Value[v＋1]≠”+”」が真の場合（ケース2，4）＞
　　　行番号25：変数signifのoffをonに更新
　行番号28：ケース分けの条件式　……　空欄d
　＜空欄dが真の場合＞
　　行番号29：Pattern[p]に変数fillの値を格納
　＜空欄dが偽の場合＞
　　行番号30：Pattern[p]にValue[v]の値を格納
＜「signif＝off」が偽の場合（ケース6，7）＞
　行番号32：ケース分けの条件式　……　空欄e
　＜空欄eが真の場合＞
　　行番号33：変数signifのoffに更新
　行番号35：Pattern[p]にValue[v]の値を格納

［設問1］

与えられた配列Pattern[]及びValue[]を基に関数Editの実行結果を解答する。この時点では，プログラムは，表2のケース1～7の処理の部分が未完成であるため，この部分は表2の内容をトレースして実行結果を求めることになる。

トレースの際は，変数signifには初期値としてoffが格納されている（行番号6），Pattern[]の添字pは0からLength(Pattern[])－1まで変化し，繰返しの都度1ずつ増える（行番号8），Value[]の添字vは0から始まり（行番号7），ケース1～7の場合だけ1増える（行番号11）ことをプログラムで確認しておく。また，条件の組合せや，条件が成り立つ場合の更新内容を表2で正確に確認するよう心掛ける。

・空欄a：Pattern[]の内容“*▯▯,▯▯▯#”，Value[]の内容“00000+”
　まず，Pattern[0]は“*”であり，Pattern[p]は“▯”と“▮”以外の文字なので，ケースは8か9になる。signifには初期値としてoffが格納されており，Value[v]，Value[v＋1]は網掛けで判定の必要がないため，この二つの条件からケース8になることが分かる。ケース8の場合，Pattern[p]はfill文字（Pattern[]中の先頭の文字）で置き換えるので，Pattern[0]は“*”，また，signifはoffのまま次の処理に進む。次に，Pattern[1]は“▯”，signifはoffなので，ケースは1，4，5のいずれかになる。Value[v]の値を確認すると，Value[0]は“0”であり，これらの条件から，該当するケースは1になることが分かる。ケース1の場合，Pattern[p]はfill文字で置き換えるので，Pattern[1]は“*”となる。また，signifはoffのまま次の処理に進む。このとき，Value[]の添字vは1増えて1となる。続いて，Pattern[2]は“▯”，signifはoff，Value[1]は“0”なので，該当するケースは1になり，Pattern[2]は“*”，signifはoff，vは2となる。更にPattern[3]は“,”（“▯”と“▮”以外），signifはoffなので，該当するケースは8になり，Pattern[3]は“*”，signifはoffとなる。このように，Pattern[p]，signif，Value[v]，Value[v＋1]の内容を順にチェックして該当するケースを特定し，各ケースでPattern[p]とsignifの内容がどう更新されるのかを一つずつ確認していく。結果は次の表のようになり，（イ）の“********”が正解である。

現在の内容					更新後	
Pattern[p]	signif	Value[v]	Value[v＋1]	ケース	Pattern[ ]	signif
* (0)	off			8	*▯▯,▯▯▯#	off
▯ (1)	off	0 (0)		1	**▯,▯▯▯#	off
▯ (2)	off	0 (1)		1	***,▯▯▯#	off
, (3)	off			8	****▯▯▯#	off
▯ (4)	off	0 (2)		1	*****▯▯#	off
▯ (5)	off	0 (3)		1	******▯#	off
▯ (6)	off	0 (4)		1	*******#	off
# (7)	off			8	********	off

注記　（）内の数字は配列の添字を表す。
　　網掛け部分は，内容を判定しない。
　　更新後のPattern[]の下線は，処理対象となった文字位置を表す。

・空欄b：Pattern[]の内容“*▯▯▯.▯▯#”，Value[]の内容“00012-”
　空欄aと同様にPattern[]，signif，Value[v]，Value[v＋1]の内容にしたがってトレースすると，次の表のようになる。ケースが1，8の場合の処理は空欄aで確認済みなので，ここではケースが4，6，9になるPattern[5]，Pattern[6]，Pattern[7]の場合について確認する。
　まず，Pattern[5]の値は“▯”，signifの値はその前の処理の結果offである。そして，Value[v]（v＝3）は“1”なので，ケースは4か5になる。この場合，Value[v＋1]も判定する必要があるのでValue[4]を確認すると“2”であり，Value[v＋1]は“＋”以外なので，該当するケースは4になる。ケース4の場合，Pattern[p]はValue[v]で置き換えるので，Pattern[5]は“1”，signifはonに更新され，vは1増えて4になって次の処理へ進む。
　次に，Pattern[6]は“▯”，signifはon，Value[4]は“2”なので，ケースは6か7のいずれかになる。このとき，Value[v＋1]であるValue[5]は“－”なので，Value[v＋1]が“＋”以外のケース6が該当する。ケース6の場合，Pattern[p]はValue[v]で置き換えるためPattern[6]は“2”，signifはonのまま，vは5になって次の処理へ進む。
　続いて，Pattern[7]は“#”であり，“▯”と“▮”以外の文字なのでケースは8か9になる。signifがonであることから，該当するケースは9になる。ケース9の場合，Pattern[p]はそのまま残すためPattern[7]は“#”，signifはonになり，処理を終わる。したがって，（ア）の“*****12#”が正解である。