第59回

C言語プログラミング能力認定試験

2　級

正答・解説

サーティファイ
情報処理能力認定委員会

＜解説＞

問１

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
イ	ア	イ	ア	ア	ア	ア	イ

(1) C言語では注釈/*と*/を入れ子にすることはできない。
(3) C言語では，#include 前処理指令によって読み込む対象のソースファイル内に，さらに#include 前処理指令を記述することができる。
(8) 文字列中に改行を指定する場合は「¥n」と記述する。

問２

(9)	(10)	(11)	(12)	(13)	(14)
ウ	イ	エ	イ	ウ	ア

ファイルの入出力を行うには，標準ライブラリ関数 fopen を呼び出し，アクセス対象のファイルに関する情報を保持する構造体 FILE(空欄 9)のポインタを取得する。その後は，取得したポインタを指定して，対象ファイルの入出力を行う。

ここで，関数fopenは，次の二つの引数を指定する。

・ファイルの名前
・ファイルの使い方を表すモード

ファイルの使い方を表すモードには，読込み("r(空欄10)")，書込み("w(空欄11)")，追加("a")が指定できる。関数fopenは，処理に失敗した場合，NULL(空欄12) を返す。

オープンしたファイルに文字を出力するには，標準ライブラリ関数fputc(空欄13)を呼び出す。また，入出力を終えたファイルを閉じるには，標準ライブラリ関数fclose(空欄14)を呼び出す。

問３

(15)	(16)	(17)	(18)	(19)
オ	イ	ウ	エ	エ

(15) x は広域変数 x(＝2)，y は広域変数 y(＝3)，z は広域変数 z(＝4)を参照する。したがって，x + y + z は 2 + 3 + 4 となり，標準出力には 9 と出力される。
(16) 関数 func1 の引数 x には広域変数 x(＝2)の値が設定される。関数 func1 内では，y は局所変数 y(＝1)，z は広域変数 z(＝4)を参照する。関数 func1 内で「x--」により引数 x の値はデクリメントされ 1 となる。したがって，関数 func1 の戻り値 x + y + z は 1 + 1 + 4 となり，標準出力には 6 と出力される。
なお，デクリメントされるのは引数xの値であって，広域変数xの値は変化しない。

(17) 関数 func2 の引数 y には広域変数 x(＝2)の値が設定される。関数 func2 内では，z は広域変数 z(＝4)を参照する。関数 func2 内で「x += 4」により広域変数 x(＝2)の値は 4 が加算され 6 となる。したがって，x + y + z は 6 + 2 + 4 となり，標準出力には 12 と出力される。

(18) 関数 func3 の引数 z には広域変数 y(＝3)の値が設定される。関数 func3 内で，「z += 7」により，引数 z の値は 7 が加算され 10 となる。y は広域変数 y(＝3)を参照する。「y -= 5」により，広域変数 y(＝3)は 5 が減算され-2 となる。x は広域変数 x(＝6)を参照する。したがって，x + y + z は 6 + (-2) + 10 となり，標準出力には 14 と出力される。

なお，7が加算されるのは引数zの値であって，広域変数zの値は変化しない。

(19) 広域変数xは，(17)の関数func2の呼び出しにより6に変化している。広域変数yは，(18)の関数func3の呼び出しにより-2に変化している。広域変数zは，初期値の4のままである。したがって，x + y + z は 6 + (-2) + 4となり，標準出力には8と出力される。

問４

(20)	(21)	(22)	(23)	(24)	(25)
ウ	イ	ア	オ	イ	ウ

(20) ～ (22) C言語のビット演算子には，

| 演算子 ･･･ ビットごとの論理和（設問20）

& 演算子 ･･･ ビットごとの論理積（設問21）

^ 演算子 ･･･ ビットごとの排他的論理和（設問22）

などがある。

(23)

	ビット列	16進数
x	0010 1010 1001 1111	2a9f
y	1111 0000 0101 1011	f05b
x | y	1111 1010 1101 1111	fadf

(24)

	ビット列	16進数
x	0010 1010 1001 1111	2a9f
y	1111 0000 0101 1011	f05b
x & y	0010 0000 0001 1011	201b

(25)

	ビット列	16進数
x	0010 1010 1001 1111	2a9f
	1111 1111 1111 1111	ffff
x ^ 0xffff	1101 0101 0110 0000	d560

問５

(26)	(27)	(28)	(29)
エ	ウ	イ	ウ

(26) ～ (28)

α に記述される演算子は &&(空欄26)，β に記述される演算子は ||(空欄27)，γ に記述される演算子は ?:（条件演算子）(空欄28) となる。

(29) p = 10, q = -1, r = 8 であるから，条件式「p > r || (q == r && p < r)」は，「真 または（偽 かつ 偽）」となり，式全体として「真」と評価される。また，条件式「(p > r || q == r) && p < r」は，「（真 または 偽） かつ 偽」となり，式全体として「偽」と評価される。次に，論理演算子&&と論理演算子||の優先順位は，論理演算子&&の方が高いので，「if (p > r || q == r && p < r)」は「if (p > r || (q == r && p < r)) 」と等価になる。したがって，「a = 1, b = 0, c = 1」となる。

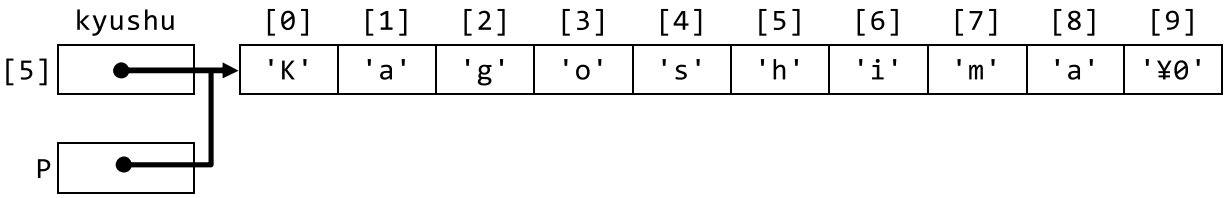
問６

(30)	(31)	(32)	(33)	(34)
ア	オ	ア	オ	エ

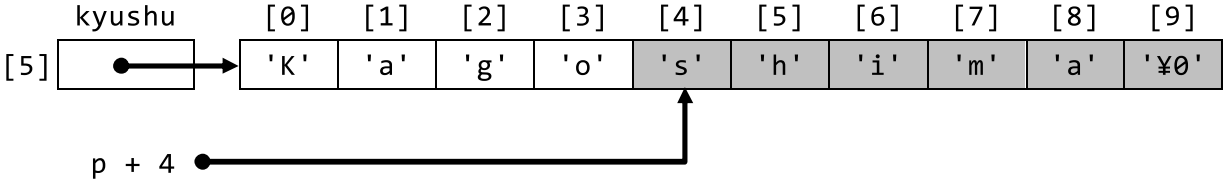
char型ポインタを要素にもつ配列kyushuの各要素は以下のように初期化される。

kyushu	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0] →	'F'	'u'	'k'	'u'	'o'	'k'	'a'	'¥0'		
[1] →	'N'	'a'	'g'	'a'	's'	'a'	'k'	'i'	'¥0'	
[2] →	'S'	'a'	'g'	'a'	'¥0'					
[3] →	'K'	'u'	'm'	'a'	'm'	'o'	't'	'o'	'¥0'	
[4] →	'O'	'i'	't'	'a'	'¥0'					
[5] →	'K'	'a'	'g'	'o'	's'	'h'	'i'	'm'	'a'	'¥0'
[6] →	'M'	'i'	'y'	'a'	'z'	'a'	'k'	'i'	'¥0'	

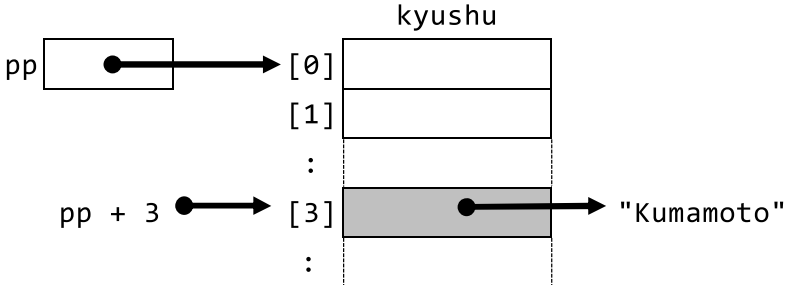
(30)「p = kyushu[5]」により，p は"Kagoshima"を指し示す。したがって，標準出力には「Kagoshima」と出力される。



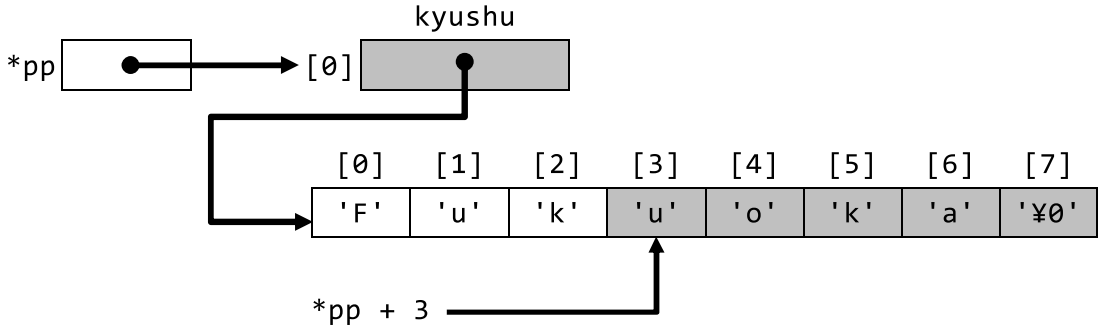
(31)「p + 4」は kyushu[5][4]の位置を指し示す。したがって，標準出力には「shima」と出力される。



(32)「pp = kyushu」により pp は，配列 kyushu の先頭要素を指し示すことになるので，「pp + 3」は kyushu[3]を指し示す。したがって，「*(pp + 3)」は文字列"Kumamoto"を指し示すので，標準出力には「Kumamoto」と出力される。



(33)「*pp」はkyushu[0]を指し示す。したがって，「*pp + 3」は文字列"Fukuoka"の4文字目の位置を指し示すので，標準出力には「uoka」と出力される。



(34) 外側のfor文による繰り返し処理において，pは順にkyushu[0]～kyushu[5]を指し示す。内側のfor文による繰り返し処理において，

i = 0のとき，qは順にkyushu[1]～kyushu[6]
i = 1のとき，qは順にkyushu[2]～kyushu[6]
i = 2のとき，qは順にkyushu[3]～kyushu[6]

i = 3のとき，qは順にkyushu[4]～kyushu[6]
i = 4のとき，qは順にkyushu[5]～kyushu[6]
i = 5のとき，qはkyushu[6]

を指し示す。if文で，「*(p + 1) == *(q + 1)」が「真」のときcntをインクリメントしている。「*(p + 1) == *(q + 1)」とは，pが示す文字列の2文字目とqが示す文字列の2文字目が等しいかを判定しているので，以下に示すときcntがインクリメントされる。

① pが"Fukuoka"を指し示し，qが"Kumamoto"を指し示すとき（i=0, j=3）
② pが"Nagasaki"を指し示し，qが"Saga"を指し示すとき（i=1, j=2）
③ pが"Nagasaki"を指し示し，qが"Kagoshima"を指し示すとき（i=1, j=5）
④ pが"Saga"を指し示し，qが"Kagoshima"を指し示すとき（i=2, j=5）
⑤ pが"Oita"を指し示し，qが"Miyazaki"を指し示すとき（i=4, j=6）

したがって，cntの値は5となり，標準出力には「5」と出力される。

問 7

(35)	(36)	(37)	(38)	(39)
イ	ア	エ	ア	エ

(35) ＜関数 Sort の処理＞(2)の「グループ間隔が 0 より大きい間」の判定条件である。グループ間隔は変数 span で管理されているので，「span が 0 より大きい」とき「真」となる式とすればよい。したがって，「span > 0」となる。

(36) ＜関数 Sort の処理＞(5)の「グループ内で大小比較要素の前位置（以下，比較前位置という）をグループ先頭位置で初期化し，限界位置より小さい間，グループ間隔の値の分を加算しながら(6)の処理を繰り返す」ための反復処理である。比較前位置は変数 k，グループ先頭位置は変数 i，限界位置は変数 j，グループ間隔は変数 span で管理されているので，変数 k を i で初期化し，k が j より小さい間，k に span を加算する for 文とすればよい。したがって，「k = i; k < j; k += span」となる。

(37) ＜関数 Sort の処理＞(6)の「比較前位置の配列要素と「比較前位置 ＋ グループ間隔」の配列要素を比較」する判定条件である。比較前位置の配列要素は a[k]，「比較前位置 ＋ グループ間隔」の配列要素は a[k + span]で参照できる。また，判定条件が「真」のときに，値の交換処理をしているので，条件式は，「a[k]が a[k + span]より大きい」とき「真」となればよい。したがって，「a[k] > a[k + span]」となる。

(38) ＜関数 Sort の処理＞(6)の「要素の値を交換する」処理である。a[k]と a[k + span]の値を交換するには，ワーク変数 w に交換前の a[k + span]を退避した後，a[k + span]に a[k]の値を代入し，a[k]に w を代入すればよい。したがって，「a[k + span] = a[k]」となる。

(39) ＜関数 Sort の処理＞(7)の「グループ間隔を半分にする」処理である。したがって，「span /= 2」となる。

問８

(40)	(41)	(42)	(43)	(44)
イ	エ	ウ	イ	イ

(40) ＜共通関数 Find の関数仕様＞（処理内容）の「一致しなかったら -1 を返却する」処理が「真」となる判定条件である。直前の for 文による反復処理は，管理エリアの「識別子」と引数で指定された「識別子」が一致したら，break 文にて「i が BCOUNT と等しくなる前」に終了する。つまり，for 文による反復処理が終了した時点で「i が BCOUNT と等しい」場合，一致する識別子がなかったことになる。したがって，「i == BCOUNT」となる。

(41) ＜関数 Add の関数仕様＞（処理内容）(2 － 3) の「ワーク変数から 256 を減算する」処理である。したがって，「wlen -= BSIZE」となる。

(42) ＜関数 Add の関数仕様＞（処理内容）(4 － 3) の「ワーク変数に文字列長を格納する。ただし，文字列長が 256 以上の場合は，ワーク変数に 256 を格納する」処理である。つまり，文字列長が 256 未満の場合は文字列長をそのまま，そうでない場合は，256 をワーク変数（wlen）に格納する。その処理を，ここでは条件演算子（? :）を使って記述している。したがって，「len < BSIZE ? len : BSIZE」となる。

(43) ＜関数 Get の関数仕様＞（処理内容）(4) の「文字列格納位置に管理エリアの要素番号位置の「文字列長」を加算する」処理である。したがって，「pos += mArea[i].len」となる。

(44) ＜関数 Get の関数仕様＞（処理内容）(6)の「要素番号位置が-1 でない間」，および＜関数 Remove の関数仕様＞（処理内容）(4)の「要素番号位置が-1 でない間」の判定処理である。したがって，「i != -1」となる。