プログラミング基礎実習
問題集

新潟大学 工学部
情報工学科
福祉人間工学科
電気電子工学科

単語英日対訳

ANSI standard	ANSI 規格
ASCII code	アスキーコード
I/O	入出力
addition	加算
address	アドレス
address-of operator	アドレス演算子
administrator	管理者
allocate	割り当て
angle brackets	<>括弧
argument	（実）引数
arithmetic operator	算術演算子
array	配列
assign	代入
assignment operator	代入演算子
binary	二進数
binary file	バイナリファイル
binary operator	二項演算子
bit	ビット
bitwise operator	ビット演算子
block	ブロック
body	本体
braces	{}括弧
brackets	[] 括弧
byte	バイト
calculate	計算
carriage return	改行
cast	キャスト
character	文字
class	クラス
code	プログラムコード
colon	:コロン
comarison operator	比較演算子
comma	, コンマ
comment	注釈 コメント
compatible	互換性
compiler	コンパイラ
complement	補数
compute	計算
constant	定数
data type	型
debug	デバッグ
decimal	10 進数
declaration	宣言
decrement	デクリメント
default	デフォルト
definition	定義
dereference operator	間接参照演算子
device	デバイス
digit	数字
dimension	次元
directive	命令
display	表示
division	除算
double quote	"ダブルクオート
dynamic	動的
element	要素
empty statement	空文
enumerate	列挙
equivalent	等価
error	エラー
evaluation	評価
executable	実行可能な
execute	実行
expression	式
field	フィールド
floating number	浮動小数点数
free	解放
function	関数
function call	関数呼び出し
global	大域
header file	ヘッダファイル
hexadecimal	16 進数
high level language	高級言語
implementation	実装
increment	インクリメント
index	添字
indirection operator	間接参照演算子
initialize	初期化
instruction	命令
integer	整数
invoke	関数呼び出し
iteration	反復
keywords reserved	予約語
left value	左辺値
library	ライブラリ
line feed	改行
link	リンク
list	プログラムリスト
local	局所
logical operator	論理演算子
loop	ループ 繰り返し
low level language	低級言語
machine language	機械語
macro	マクロ
member	メンバ
memory	メモリ
mode	モード
modifier	修飾子
modulus	法
multiplication	乗算
negative	負
nest	ネスト 入れ子
newline	改行
null	ヌル
null character	null 文字
null statement	null 文
number	数字
numeric number	数値
object	オブジェクト
object file	オブジェクトファイル
octal	8 進数
operation	演算
operator	演算子
parameter	（仮）引数
parenthes	() 括弧
pass	渡す
pointer	ポインタ
positive	正
preprocessor	プリプロセッサ
print	表示
program	プログラム
programming language	プログラミング言語
prototype	プロトタイプ
quote	' クオート
random access	ランダムアクセス
real number	実数
recursion	再帰
recursive function	再帰関数
reference	参照
relational operator	関係演算子
release	解放
remainder	剰余
return value	返値
right value	右辺値
run	実行
runtime	実行時
scalar	スカラー
scope	スコープ 有効範囲
screen	画面
semicolon	; セミコロン
sort	ソート
specifier	指定子
standard input	標準入力
standard output	標準出力
statement	文
static	静的
storage	記憶
stream	ストリーム
string	文字列
structure	構造体
structured programming	構造化プログラミング
subscript	添字
substitution	代入
subtraction	減算
syntax	構文
terminal	画面
text	テキスト
unary operator	単項演算子
union	共用体
user	ユーザー
value	値
variable	変数
vector	ベクトル
void	void
warning	警告
word	ワード

1. 式の評価

問1

「評価」とは①**計算すること**である。

> 例：1+1 の評価結果は 2

次の C 言語の式を評価せよ。

```
1 + 2
2 - 1
2 * 3
10 / 5
1
2.5
```

```
22 * 3.3
15 - 21
33.3 / 3
10.2 + 5.1
1000
-12.5
```

ここで「*」「/」はそれぞれ乗算（×）、除算（÷）を表す演算子である。（注：C 言語の式の中では×や÷の文字は使えない）

問2

「評価」とは②**判定すること**（大小や同異などを）である。判定結果は、正しい（真）ならば 1、間違っている（偽）ならば 0 で表す。

> 例：1 は 0 より大きく 1>0 は正しいので、1>0 の評価結果は 1（真）である。また 1!=0 の評価結果は 1（真）、1!=1 の評価結果は 0（偽）である。

次の式を評価せよ。

```
0 < 1
3 > -1.5
-3 >= -10
-1 <= -3
3.4 == 3.4
1.0 != 10
```

```
0 < -1
1.0 != 10.0
-3 >= 10
0.34 == 3.4
3 < -1.5
-11.1 <= -33.3
```

ここで「==」「>=」「<=」はそれぞれ等号（＝）、以下（≧）、以上（≦）を表す演算子である。また「!=」は非等号（≠）を表す演算子である。（注：C 言語の式の中では＝≧≦≠等の文字は使えない）

問3

「評価」は基本的に左から行う。

例：1+2+4 の評価の場合。まず 1+2 を評価すると 3 になる。次に 3+4 を評価し、7 になる。

次の C 言語の式を評価せよ。

```
-2 + 4 - 3.5
3 - 4 + 18
3 * 8 / 4
10 / 2
10 / 2 / 5
10 / 2 * 5
1.1 + 2 > 0.1
5 * 2 <= 10.2
3 - 4 == 1
2 - 3 != 1
```

```
-3.5 - 2 + 4
3-4+18
3 * 8/4
100 / 20
10 / 2/5
10/2 >= 5
1.1 + 0.1 < 1.1
55 * 2 <= 110.2
3-4 != 1
2-3 == 1
```

問 4

「評価」には優先順位がある。優先順位が高いものから評価を行う。優先順位が同じ場合、左から評価する。

例：1+2*4 の評価の場合。まず 2*4 を評価すると 8 になる。次に 1+8 を評価し、9 になる。

演算子の優先順位を調べて、次の C 言語の式を評価せよ。

```
-2 + 4 * 3
10 / 2 - 5
2 + 8 * 4 - 2
2 * 4 - 3 + 18 / 2 + 1
0 < 1 + 2
10 - 2 > 5 * 2
2 - 3.5 <= 3 + 1.2
10 - 2 == 5 * 2 - 2
10 + 2 != 5 * 2 + 2
```

```
-2*4 + 3
10-20 /  5
2*8 - 4*2
2*4-3+18 / 2+1
0     +1 < 2
10 - 2>5 * 2
2  - 3.5<=3+  1.2
5 * 2-2 == 10 - 2
10 + 2 != 5 * 2+2
```

問 5

「評価」には優先順位がある。式中の括弧「(」と「)」で括られた部分は先に評価する。大括弧、中括弧は使えないので、すべて小括弧のみで表す。

例：(1+2)*4 の評価の場合。まず (1+2) を評価すると 3 になる。次に 3*4 を評価し、12 になる。

演算子の優先順位を調べて、次の C 言語の式を評価せよ。

```
-2 * (4 - 3)
24 / ((2 - 5) * 4)
-(8 + (3 * 8)) / 4
10 / (2 * 5)
40 / (-(20 / 5))
2 * (4 - 3 + 17) / (2 + 1)
(1 + (1 + (1 + (1 + (2 * 3)))))
10 < (1 + 2) * 3
```

```
(5 > 2)
(5 > 2) + 1
(5 > 2) + (3 - 2 != 1)
2 == 5 - 3
4 * (2 == 5 - 3) - 1
(4 < 5)
(4 < 5) < 6
4 < (5 < 6)
4 < 5 < 6
2 < 3 > 1
```

問 6

「評価」できる C 言語の式の形は、数学の数式とは異なる。

例：分数は、分母と分子を括弧 () で括り、除算演算子/で表す。

次の C 言語の式を数学の数式に（除算は分数で）書き直せ。また、数学の数式を C 言語の式に書き直せ。

C 言語の式		数学の数式
`-2 + 15 / 3`	→	$-2+\frac{15}{3}$
`10 / 3 + 1`	→	
`2 * 5 / 2 * 5`	→	
`9 / 3 / 3 - 1`	→	
`9 / (3 / 3) - 1`	→	
`1 + 3 * 4 / 2 - 10`	→	
`(1 + 3) * (4 / 2) - 10`	→	
`(1 + 3) / (4 * 2) - 10`	→	
	→	$1 \div 2 \times 4$

$\rightarrow \quad 1 - 2 \div 2 \times 4 + 3$

$\rightarrow \quad (1 - 2) \div 2 \times (4 + 3)$

$\rightarrow \quad 1 + \dfrac{1 - 3}{\frac{2}{4} + 1} - 2$

$\rightarrow \quad 1 + [1 + \{1 + (2 - 3)\}]$

$\rightarrow \quad \dfrac{\frac{1}{4} - \frac{4}{2}}{-(1 - \frac{3}{2})} + \dfrac{6}{2}$

問 7

演算子+, -, *, ==, !=は可換である。

例：1+2 は 2+1 と等しい。

次の二つの C 言語の式の評価結果は同じか違うか。同じなら○、違うなら×で答えよ。

式 1	式 2	○, ×
`1 + 3`	`3 + 1`	
`3 * 5`	`5 * 3`	
`3 + 3 * 5`	`5 * 3 + 3`	
`3 * 7 / 3`	`7 / 3 * 3`	
`8 / 3 / 3`	`8 / (3 * 3)`	
`(3 + 3) / 7`	`3 + 3 / 7`	
`3 + 3 / 7`	`3 + (3 / 7)`	
`8 / 3 * 3`	`8 / (3 * 3)`	
`(5 + 3) / (7 + 3)`	`5 + 3 / 7 + 3`	
`5 + 3 * 7 + 3`	`(5 + 3) * 7 + 3`	
`2 / 1.12 == 4`	`4 == 2 / 1.12`	
`4 != 1.12 + 2`	`2 + 1.12 != 4`	
`3 * 5`	`5*3`	
`1+2 * 3`	`1 + 2*3`	
`3+3 / 7`	`3 + 3/7`	
`12 / 2 / 3`	`12 / 3 / 2`	
`1 - 2`	`-2 + 1`	
`2 * (3 + 4)`	`2 * 3 + 4`	

2. 論理式の評価

二つの真偽の論理関係には論理積（AND）と論理和（OR）がある。それぞれの真偽値表は次の通り。

AND	結果
0 AND 0	0
0 AND 1	0
1 AND 0	0
1 AND 1	1

OR	結果
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	1

C 言語において、AND または OR を表す論理演算子はそれぞれ &&, ||である。

例：(1>0)&&(3>0) の場合、1>0 は 1（真）、3>0 は 1（真）なので、評価結果は 1（真）。(1<0)&&(3>0) の評価結果は 0（偽）。(1<0)||(3>0) の評価結果は 1（真）。(1<0)||(3<0) の評価結果は 0（偽）。

論理演算子の場合も、優先順位が同じなら左から評価する。

問 8

演算子の優先順位を調べて、次の C 言語の式を評価せよ。

```
(0 < 1) && (0 > 1)
3 > 2 - 1 && 5 <= 6 - 1
3 == 2 - 1 || 5 != 6 - 1
((0 < 1) && (0 > 1)) || (0 < 2)
2 > 0 && 0
1 && 3 <= 4
1 && 1
1 && 0
1 || 0 && 1
1 && (0 || 1)
(0 && 0) || 1
0 && (0 || 1)
3 + ((0 > 1) || (0 < 3))
(3 > 1 && 5 >= -1) * 5
```

問 9

否定（NOT）を表す演算子は！である。否定演算子は、式の前に置く。

例：`!(1>0)` の場合、`1>0` は `1`（真）なので、評価結果は `0`（偽）。
`!(1>2)` の場合、`1>2` は `0`（偽）なので、評価結果は `1`（真）。

演算子の優先順位を調べて、次の C 言語の式を評価せよ。

```
(0 < 1) && !(0 > 1)
!(3 > 2 && 5 <= 6)
!(5 != 6 - 1)
!(!(!(2 > 0)))
!0
!(1 && 1)
!!!1
1 && !1
```

問 10

次の式「 `a > 2 && a < 4` 」と等価な式を、`&&`を使わずに書け。
（ヒント：ド・モルガンの法則）

問 11

次の数学の条件式を C 言語の式に書き直せ。また、C 言語の式を数学の条件式に書き直せ。

数学の条件式		C 言語の式
$3 \leq 2$ かつ $4 \geq 5$	$\rightarrow$	`3 <= 2 && 4 >= 5`
$3 > 4$ または $-2 < 6$	$\rightarrow$	
	$\rightarrow$	`5 > 4 && 4 > 3`
$3 \geq 2 \geq 1$	$\rightarrow$	

問 12

「評価」において、「真」は 1 である（真 ⟶1）が、1 だけではなく「0 以外」はすべて「真」である（真 ⟵0 以外）。また、0 の時のみ偽である（偽 ⟷ 0）。

例：`2&&(3>0)` の評価結果は `1`（真）。`!2` の評価結果は（2 が真だからその逆となり）`0`（偽）。

演算子の優先順位を調べて、次の C 言語の式を評価せよ。

```
2 && 1
-2 || !8
3 + 4 * 2 || 4 - 3 - 1
(3 + (0 > 1)) || (0 < 3)
!0 + !2
!(2 + 3) * 4
1 + 2 + (3 < 4)
1 + 2 + (3 < 4) < 2
3 && 2 && -1 && 8 && 0
(3 > 1) && 5
3 > (1 && 5)
```

問 13

正負を表す演算子はそれぞれ +, - である。-を連続して適用する場合（または+を連続して適用する場合）には空白をあける（--,++等は不可、+-,-+等は可）。

例：(- -2) の評価結果は 2

次の C 言語の式を評価せよ。

```
-(2)
-(-2)
- - 2
- - -2
+2
+(-2)
+ - 3
```

```
+(-(+(-2)))
+-+-2
3 + - 2
-+-+3+-+-2
+ + - + 3
4 / (-2)
4 / -2
4 / -(2 - 1)
```

問 14

次の C 言語の式は正しくない。間違いを指摘せよ。

```
3 <> 2
4 > = 2
4 => 2
3 !== 1 + 2
4 - 1 = = 3
2 =< 3
4 \ 2
--2
+--+2
```

3. 変数、代入

「変数」とは評価結果を保持するために使う記号である。変数に評価結果の値を保持させることを「代入」と呼び、代入演算子「=」を使う。

=の左辺に変数を書き、右辺に評価する式を書く。

> 例：変数 x に 2 を代入するには x=2 と書く。
> y=3+4 と書くと、3+4 の評価結果 7 が y に代入される。

y=3+4 のような、変数に代入するように書いたものを「代入文」と呼ぶ。

問 15

それぞれ変数に式の評価結果を代入する代入文を書け。

注意：**C 言語における「=」は代入を意味し、数学における等号記号「=」とは全く意味が異なる。**

変数	評価する式	代入文
y	`3 + 4`	`y = 3 + 4`
x	`-3`	
x	`1 - 2`	
x	`1 + 2 + 3 + 4 * 5`	
w	`-2 * (4 - 2)`	
x	`0 < 1`	
y	`3 > (1 && 5)`	
x	`1 - 2 == 4`	

問 16

ある代入文によって変数に値を実際に代入することを、代入文を「実行」する、または代入を「実行」すると言うことがある。

代入は、なにより先にまず右辺を評価する。代入が実行された「後」の変数は、評価結果の値を保持している。次の代入文を実行した後に変数が保持している値を書け。

代入文	値
`x = 1 + 2`	3
`y = 3 - 4 + 18`	
`y = 10 / 2 / 5`	
`y = (2 - 3 != 1)`	
`y = 1 + 2 * (3 + 4)`	
`y = (1+(1+(1+(1+(2*4)))))`	
`y = 2*(4*(3*(1+2)+1)+1)+1`	

問 17

「変数」を「評価」すると、その変数が保持している値が評価結果となる。評価されるだけなので、変数が保持している値は変わらない。

> 例：変数 x の保持している値が 3 のとき、x の評価結果は 3 であり、x+1 の評価結果は 4 である。

代入文「x = 1 + 2」と「y = 4 / 2」によって変数 x と y に値が代入されているとき、次の C 言語の式を評価せよ。

```
x
y
x + 2
2 + x * 3
x == 3
6 / 2 != x
(x > 1) && 5
9 / x / x
```

```
x + y
x + y + y
x * (y + y)
2*(x*(y+1))
y == x
x * y > 0
y - 1 > x && 0 < x
```

問 18

同じ変数に何度も代入を実行するたびに、保持している値は変わる。代入文が実行された時に、変数の値が書き換えられる。**代入されたことのない変数の値は不定である。**代入文を連続して実行する場合には、代入文と代入文の間に区切り文字「;」（セミコロン）が必要である。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での変数 x と y の値を書け。

数学における「変数」は、ある決まった数を表すために用いるので、同じ変数の値は途中で勝手に変わること、変えることはできない。

しかし、C 言語における「変数」は、評価結果を保持するために用いる記号であり、同じ変数に何度も代入を実行するたびに、保持している値は変わる。

代入文	x の値	y の値
`x = 1 + 2;`	3	不定
`y = x + 1;`	3	4
`x = 4 / 2;`	2	4
`y = x * 3;`	2	6
`x = 10 / (2 * 5);`		
`y = x > 2;`		
`x = 1;`		
`y = x + 1;`		
`x = y + 1;`		
`y = x + 1;`		
`x = y + 1;`		
`y = x + 1;`		
`x = y + 2;`		
`y = x + 2;`		
`x = y + 3;`		
`y = x + 4;`		
`x = y - 10;`		
`y = x + 1;`		

問 19

「代入」も「評価」の一つである。代入文を評価すると、変数に代入された値が評価結果となる。通常の代入文と同じように、変数には値が代入されている。

例：x=2 の評価結果は 2 である。

次の C 言語の式を評価せよ。

```
x = 3
x = 3 * 2 + 1
y = 3 + 2
y = (3 + 2)
(y = 3) + 2
6 / (x = 3) + 1
(y = 3) + ((x = 4) > 1) + 2
(y = (x = 2 * 3)) + 1
```

問 20

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。不定であれば不定と書け。

代入文	x の値	y の値	z の値
`x = 1;`			
`y = 2;`			
`z = 3;`			
`x = y + z * 2;`			
`x = (y = (z = 7));`			
`x = y = z = 8;`			
`x = y = z + 6;`			
`z = (x == y);`			
`z = (x = y);`			
`z = (x = (y == 1));`			
`x = 3 + 2;`			
`y = x = 3 + 2;`			
`y = (x = 3) + 2;`			

問 21

次の C 言語の式は正しくない。間違いを指摘せよ。

```
x = y = 3 + 1 = z
(x = y) = 3
4 = z + 1
```

4. 変数の初期化

問 22

何も代入されていない変数は値が不定であり、その変数は「初期化されていない」と言う。評価される式の中に、値が不定の変数が含まれている場合には、評価結果全体が不定となる。

例：変数 x の値が不定のとき、x+2 の評価結果は不定。

以下の代入文が上から順番に実行されるとき、各代入文が実行された時点での各変数の値を書け。不定であれば不定と書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3;`			
`y = x;`			
`x = y + 2 * z;`			
`y = x + 6;`			

以後の問題では「不定であれば不定と書け」という指示は省略するが、今後、変数の値が不定であれば「不定」と書くこと。

問 23

以下の代入文が上から順番に実行されるとき、各代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`z = -4;`			
`x = z * 2 + y;`			
`z = x / 6;`			
`x = 2 ;`			
`z = y + x * 6;`			

問 24

代入文の実行順序は重要であり、順序が変わると変数の値も変わってしまう。

以下の代入文が上から順番に実行されるとき、各代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3;`			
`x = y;`			
`z = x + 2;`			
`y = x;`			

次に、以下の代入文が上から順番に実行されるとき、各代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3;`			
`y = x;`			
`z = x + 2;`			
`x = y;`			

上の二つの例は、どの代入文の順序が違うのか。

問 25

変数 x と y の値を「入れ換える」ことができる。そのためには、別の変数を仲介に使う。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3;`			
`z = x;`			
`x = y;`			
`y = z;`			

入れ換える前の
x の値：_____, y の値：_____

入れ換えた後の
x の値：_____, y の値：_____

問 26

変数 z と y の値を入れ換えるように下線部を埋めよ。また、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`z = 1;`			
`y = 3;`			
`_____;`			
`_____;`			
`_____;`			

問 27

変数 x,y,z の値がそれぞれ 2,15,87 のとき、この値を順に入れ換える。
以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`w = x;`			
`x = y;`			
`y = z;`			
`z = w;`			

入れ換える前の
x の値：_____, y の値：_____,
z の値：_____

入れ換えた後の
x の値：_____, y の値：_____,
z の値：_____

問 28

代入演算子「=」による、左辺の変数への代入は、右辺の評価が全て終わってから行われる。したがって、左辺の変数と同じものが右辺に登場していても、その変数の値が右辺の評価に使われてから、左辺の変数として値が代入される。

例：x の値が 2 のとき、代入文 x=x+2 の実行後の x の値は 4 である。C 言語での代入演算子「=」は数学での等号「＝」とは全く意味が異なることに注意。

以下の代入文が上から順番に実行されるとき、それぞれの「代入文が実行された後の」各変数の値を書け。

代入文	x の値	y の値
`y = 2;`		
`x = 3;`		
`x = y + x;`		
`y = 2;`		
`y = y + 1;`		
`y = y + 1;`		
`y = y + 1;`		
`y = y + 1;`		
`x = 1;`		
`x = x + 2;`		
`x = x + 2;`		
`x = x + 2;`		
`x = x + 2;`		
`x = 2;`		
`x = x + 2;`		
`x = x + 2;`		
`x = x + 2;`		
`x = x * 4;`		
`y = 1;`		
`y = y * 2;`		
`y = y * 3;`		
`y = y * 4;`		
`y = 1000;`		
`y = y / 10;`		
`y = y / 10;`		
`y = y / 10;`		
`x = x / 2;`		
`x = x / 2;`		
`x = x / 2;`		
`x = x / 2;`		

5. 数学の数式とC言語の式・関数

問 29

数学において、変数と数の積は、通常は乗算の記号「×」を書かない。しかしC言語では、必ず乗算の演算子「*」が必要である。

例：数式 $2x+1$ を表すC言語の式は `2*x+1`。

次のC言語の式を数学の数式に（除算は分数で）書き直せ。また、数学の数式をC言語の式に書き直せ。

C言語の式		数学の数式
`-2 + 3 * x / 3`	→	$-2+\dfrac{3x}{3}$ （$-2+x$ も可）
`y * 5 / z * 5`	→	
`x * x / 3 * y + y`	→	
`9 / x / y - 1`	→	
`9 / (z * z) - 1`	→	
`x + y * z / w - u`	→	
`(1 + v) * (u / 2) - 10`	→	
`7 * x * y * z * a * b`	→	
	→	$2x+1$
	→	ax^2+bx+c
	→	$\dfrac{vt^2}{2}$

$$\rightarrow \quad h - g\frac{t^2}{2}$$

$$\rightarrow \quad 1 + \frac{2x - m}{\frac{3a}{2} + b} - 2c$$

$$\rightarrow \quad 1 + [x + \{y + (2 - z)\}]$$

$$\rightarrow \quad \frac{4x - 2u}{-(1 - 3w)} + \frac{6}{2}$$

問 30

数学における初等関数（例えば sin, exp など）や演算（例えば微分積分）や定数（例えば π, e など）は、C 言語の式中で使えるものもあれば使えないものもある。

C 言語の式中で使えるものの大部分は、**C 言語の「関数」**として用意されている。関数には関数名と**引数（ひきすう）**があり、関数名の後に引数を括弧「`()`」で括る。

関数 $f(x)$ があるとき、数学では「x の関数 f」と呼び、C 言語では `f(x)` を「引数 `x` をとる関数 `f`」と呼ぶ。

> 例：数式 $\sin x$ を表す C 言語の式は `sin(x)`、$\sqrt{x}$ は `sqrt(x)`、$\sqrt[3]{x}$ は `cbrt(x)`。ここで `sin`, `sqrt` 等は C 言語の関数名、`x` は引数である。

C 言語で用意されている数学関数を調べて、次の C 言語の式を数学の数式に書き直せ。また、数学の数式を C 言語の式に書き直せ。

C 言語の式		数学の数式
`x + log(2)`	$\rightarrow$	$x + \log_e 2$
`x + log10(20)`	$\rightarrow$	
`2 * exp(3 * x) / (2 * a)`	$\rightarrow$	
`2 * a * fabs(x) + b`	$\rightarrow$	
`sqrt(b * b - 4 * a * c)`	$\rightarrow$	

`sqrt(x * x + y * y)` →

`2 * cbrt(2 * c)` →

`sin(x) * sin(x) / 2 - 4` →

`cos(x / 2) / 2 * y` →

`pow(x + 2 * y, 3)` →

`floor(z) + ceil(y)` →

→ $2|x| + \sin x$

→ $1 + (y + 3x)^{15} + 4y$

→ $4\sqrt{\dfrac{k}{m} - 1}$

→ $\sin^{-1}(x) - 1$

→ $\log_e x^3$

→ $1 - e^{-\frac{t}{2r}}$

→ $\dfrac{\tan a + \tan b}{1 - \tan a}$

→ $\tan^{-1} \dfrac{2a}{b}$

問 31

数学では変数を a_n や v' や I_R などと書くことができる。しかし C 言語では変数の書き方に

制約がある。

C 言語の変数の変数名は、

- 一文字目が必ずアルファベットで始まる（大文字小文字どちらでもよい。つまり a〜z、A〜Z）。
- 二文字目以降はアルファベット、数字、アンダーバー（_）を組合せる。それ以外は使用不可。

例：X0, x0, x_1, An, an, bn, c_1000th, I_R, test_variable_name, theta, angle, Omega, Lambda, delta, epsilon, distance, c, C, E, CE などは変数名にできる。

変数名の制約に注意して、自分で分かりやすい変数名を考えながら、次の数学の数式を C 言語の式で表せ。

C 言語の式		数学の数式
`a0 + a1 + a2 + delta_x`	$\rightarrow$	$a_0 + a_1 + a_2 + \delta x$
	$\rightarrow$	$2gx_0 \cos\dfrac{\theta}{2}$
	$\rightarrow$	$\dfrac{4}{3}\Omega^2 + \dfrac{k_0 q Q}{\Lambda}$
	$\rightarrow$	$\mu(v_{20} - v_{21})$
	$\rightarrow$	$\sqrt{\dfrac{Gm\lvert M\rvert}{R\ell_1\varepsilon_0}}$

問 32

数学で使用する様々な記号（「±」「′」「$\widehat{\ }$」「$\bar{\ }$」など）は、C 言語で使えないものが多い。そのため、数学の記号の意味を解釈して、その数式を表現するための新たな変数名を考えなければならない。例えば、$|x|$ は C 言語において絶対値関数を使って `fabs(x)` で絶対値が計算できる。しかし、微分を意味する x' は対応する C 言語の関数はないため、`x_dash` などと書く（ただし微分計算はできない）。

変数名の制約に注意して、自分で分かりやすい変数名を考えながら、次の数学の数式を C 言語の式で表せ。

C 言語の式		数学の数式
`2 * a + b_dash` `2 * a - b_dash`	$\rightarrow$	$2a \pm b'$

$\rightarrow \quad -b \pm \sqrt{b^2 - 4ac}$

$\rightarrow \quad y - x' \sin x$

$\rightarrow \quad v'' + 2\dot{u} - 3\hat{z}$

$\rightarrow \quad \frac{1}{2}\overline{V}^2 Gm + |\overline{V}|$

問 33

数学における方程式は、そのままではC言語では使えない。

そのため、数学の数式の意味を解釈して、その数式を表現するためのC言語での式と変数を考え、「式の評価結果を変数に代入する」形式にしなければならない。

> 例：方程式 $2x + 3y = 1$ を解く場合、y の値がわかっているときには、方程式を x について解き、$x = \frac{1-3y}{2}$ と変形すると x を求めることができる。これに相当する代入文は `x=(1-3*y)/2;` である。

変数 `y, b, c` に値がすでに代入されているとき、次の数学の方程式を解き x を求めるための、C言語の代入文を書け。

C言語の代入文		数学での方程式
`x=(1-3*y)/2;`	$\leftrightarrow$	$2x + 3y = 1$
	$\leftrightarrow$	$4x - y^2 + 8 = 0$
	$\leftrightarrow$	$\frac{1}{2}x + 2\sin\frac{\theta}{2} = b$
	$\leftrightarrow$	$\frac{1}{2}x^2 + bx + c = 0$

問 34

方程式 $x^2 - x - 6 = 0$ を解の公式を用いて解く。この解を計算するC言語の代入文の組を考える。

次の代入文が上から順番に実行されるとき、それぞれの代入文が実行された後での各変数の値を書け。

代入文	x1	x2
`x1 = (-(-1) + sqrt((-1)*(-1) - 4*(-6))) / 2;`		
`x2 = (-(-1) - sqrt((-1)*(-1) - 4*(-6))) / 2;`		

問 35

方程式 $x^2 - x - 6 = 0$ を解く。この解を計算する C 言語の代入文の組を考える。

次の代入文が上から順番に実行されるとき、それぞれの代入文が実行された後での各変数の値を書け。

代入文	D	x1	x2
`D = sqrt((-1) * (-1) - 4 * (-6));`			
`x1 = (-(-1) + D) / 2;`			
`x2 = (-(-1) - D) / 2;`			

問 36

方程式 $x^2 - x - 6 = 0$ を解く。$x^2 + bx + c = 0,\ b = -1,\ c = -6$ として、この解を計算する C 言語の代入文の組を考える。

次の代入文が上から順番に実行されるとき、それぞれの代入文が実行された後での各変数の値を書け。

代入文	b	c	D	x1	x2
`c = -6;`					
`b = -1;`					
`D = sqrt(b * b - 4 * c);`					
`x1 = (-b + D) / 2;`					
`x2 = (-b - D) / 2;`					

問 37

方程式 $x^2 - x - 6 = 0$ を解く。$x^2 + bx + c = 0,\ b = -1,\ c = -6$ として、この解を計算する C 言語の代入文の組を考える。

次の代入文が上から順番に実行されるとき、それぞれの代入文が実行された後での各変数の値を書け。

代入文	b	c	D	x1	x2
`b = -1;`					
`c = -6;`					
`D = b * b - 4 * c;`					
`D = sqrt(D);`					
`x1 = -b + D;`					
`x1 = x1 / 2;`					
`x2 = -b - D;`					
`x2 = x2 / 2;`					

6. 型

数学における「数」には自然数/整数/有理数/実数/複素数があるが、通常は変数 x が実数なのか整数なのかということは気にしない。

しかし、**C 言語では**全ての変数や数字がどの種類の「数」なのかを常に考えなければならない。この種類のことを「型」と呼ぶ。変数の型は、一度決めたら変更できない。

重要な型には次の二つがある。

- 整数型 `int`（整数 integer の頭文字をとったもの）
- 実数型 `float`（浮動小数点 floating point の一単語をとったもの。実数は real number であるが）

整数型を `int` 型、実数型を `float` 型と呼ぶこともある。

注意：前章までの解説や問題では、説明を簡単にするために、変数の型の概念を無視しているため、実際の計算とは多少異なることがある。

問 38

整数型の変数 x に整数 2 を代入する（x=2）と、x の値は 2 であるが、実数 2.1 や 2.8 を代入すると、小数部分は切捨てられて、代入後の x の値はやはり 2 である。

変数 x,y,z が整数型（int 型）で、以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3.2;`			
`z = x;`			
`z = -1.9;`			

問 39

実数型の変数 x に実数 2.1 を代入する（x=2.1）と、x の値は 2.1 であるが、整数 2 を代入すると、代入後の x の値は 2.0 である。

変数 x,y,z が実数型（float 型）で、以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2;`			
`x = 3.2;`			
`z = x;`			
`z = -1.9;`			

以後は「評価結果が実数型であれば小数点を付けること」とは明記しないが、付けること。

問 40

C 言語の式中の数字（2.1, -9 など）は、すべて整数型か実数型である。数字に小数点が含まれる場合は実数型、小数点がなければ整数型である。

実数型の数を書く場合、左端や右端の 0 は省略することができる。

> 例：2 や-9 や 1 は整数型、1.9 や 0.9 や.9 や-.9 や 2.0 や-2. は実数型である。

変数 x,y が整数型（int 型）、変数 z が実数型（float 型）で、以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 2.;`			
`x = 1.2;`			
`x = .2;`			
`z = 1.9;`			
`x = z;`			
`z = 0.9;`			
`x = z;`			
`z = .9;`			
`z = -.9;`			
`z = x;`			

問 41

評価結果の値にも型がある。

式の中が整数型（の数や変数）のみであれば、評価結果は整数型である。

式中に実数型（の数や変数）が一つでもあれば、その部分の評価結果は実数型になる。

> 例：1+2*3.0 は、2*3.0 の部分が先に評価されるので、この部分の評価結果が 6.0 になる。そして 1+6.0 が評価されるので、式全体の評価結果は実数型の 7.0 である。

除算演算子/は特殊である。整数型同士の割算の評価結果は、整数である「商」である。実数型を含む割算の評価結果は、通常の割算の結果である小数で、実数型である。

> 例：5/2 の評価結果は整数型の 2 である。

> 例：5.0/2.0 の評価結果は実数型の 2.5 である。

次の C 言語の式の評価結果とその型を書け。

```
-2
-2+4
-2+4-3.5
-2+4-3.
5 / 2
5. / 2
5 / 2.0
5. / 2.
```

```
1.5 * 2
1.5 * 2 / 3
1.5 * (2 / 3)
1.5 * (2 / 3.)
2/5
10*2/5
2/5*10
```

問 42

変数 x,y が整数型（int 型)、変数 z が実数型（float 型）で、以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`y = 1 / 2;`			
`y = 4 / 3;`			
`x = 3 * 2.5 + 5 / (2 - 1.);`			
`z = x + 1;`			
`z = x + 1.;`			
`z = z + .1;`			
`z = z + .1;`			
`z = z + .1;`			
`y = z * 2 + x;`			
`y = (z + 1) / 2;`			
`y = 1 / 2 * (z + 1);`			
`y = 1 / 2.  * (z + 1);`			

7. 変数宣言と型変換

C 言語では、式中で使う変数は全て前もって「使う」ことを宣言しておかなければならない。これを「変数宣言」と呼ぶ。変数宣言でそれぞれの変数の型を決める。

- 整数型（int 型）の変数 x を宣言するには、`int x;` と書く。変数 x と y と z を同時に宣言するには、`int x, y, z;` と書く。
- 実数型（float 型）の変数 x を宣言するには、`float x;` と書く。変数 a と b と c を一度に宣言するには `float a, b, c;` と書く。

前章までは、説明を簡単にするために変数宣言を省略していた。今後は必ず宣言を行うこと。

問 43

全ての変数宣言は、**代入文の前で**行う。
以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。その時点で宣言されていなければ、未定義と書け。

代入文	x の値	y の値	z の値
`int x;`			
`float y, z;`			
`y = 2.;`			
`x = 1.2;`			
`x = .2;`			
`z = 1.8;`			
`x = z * 2;`			
`z = z * 0.5;`			
`x = x + y;`			

今後は「その時点で宣言されていなければ未定義と書け」という指示は省略するが、今後、変数が宣言されていない場合は「未定義」と書け。

問 44

全ての変数宣言は、代入文の前で行う。
以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。
また前問とどう値が変わるのかに注目せよ。

代入文	x の値	y の値	z の値
`int z;`			
`float y;`			
`int x;`			
`y = 2.;`			
`x = 1.2;`			
`x = .2;`			
`z = 1.8;`			
`x = z * 2;`			
`z = z * 0.5;`			
`x = x + y;`			

問 45

次のプログラムには間違いがある。訂正せよ。

```
int a, b;
a = 2;
b = a + 2;
int c;
c = a + b;
```

問 46

C 言語には、第二の実数型（double 型）がある。double 型は float 型とほぼ同じである。int 型や float 型を含む式中に double 型が一つでもあれば、その部分の評価結果は double 型になる。

変数 x,y が int 型、z が float 型、a,b が double 型の時、次の C 言語の式の評価結果の「型」を書け。

```
x - 2
x - 2 * y
5 / a
x + 1 + 5 / z * 2.
x + 1 + 5 / a * 2.
x + y + z + a + b
```

float 型と double 型の違いは表現できる数値の範囲が異なる（次章を参照）。

問 47

C 言語では、関数にも型がある。関数 f(x) の引数も関数の型も実数の場合、「実数引数 x をとる実数関数 f」と呼ぶ。

変数 x,y が int 型、z が float 型、a,b が double 型の時、C 言語の数学関数の型を調べて、次の C 言語の式の評価結果の型を書け。

```
x - log(2)
x - 2 / fabs(z) * y
sin(z) * cos(x)
1 / 2. * exp(a * b)
floor(z) + ceil(a)
```

問 48

C 言語では、評価の型を変換することができる（ただし変数の型は変更できない）。型名を括弧「()」で括り、式の直前に書くと、評価結果の型を変換する。

整数型を実数型に変換する場合は、整数の値はそのまま実数になる。実数型を整数型に変換する場合は、実数の値の小数点以下は切捨てられて、整数部分だけが整数型として残る。

また、単に 2.0 や 1.5 のように小数点を付けた実数を書くと、double 型になる。2 と書くだけでは int 型である。

例：2.0+1.2 の評価結果は 3.2 で型は double である。
(int)(2.0+1.2) の評価結果は 3 で型は int である。
2.0+(int)1.2 の評価結果は 3.0 で型は double である。

変数 x,y が int 型、z が float 型、a が double 型の時、次の C 言語の式の評価結果の型を書け。

```
(float)(x - 2) * a
(float)x - 2 * a
(float)x - 2 * (float)a
(float)x - 2 * (double)a
(float)(x - 2 * a)
5 / 2
5 / 2.
5 / (float)(exp(x))
5 / (float)exp(x)
(float)1 / (float)2
(float)(1 / 2)
(float)1.0 / 2.0
(float)x / (float)y
(float)x / (double)y
(int)floor(z) + (int)ceil(a)
```

問 49

変数 x, y が int 型、z が float 型、a が double 型とする。

次の C 言語の式を、評価結果の値とその型が同じになるように、最も簡単な形式に書き換えよ。すでにもっとも簡単な形式になっているならばそのままでよい。

例：5.0/(float)2 は 5./2 と等価である。

```
(float)(x - 2) * a
(float)x - 2 * a
(float)x - 2 * (float)a
(float)x - 2 * (double)a
(float)(x - 2 * a)
5 / 2
5 / 2.
5 / (float)(exp(x))
5 / (float)exp(x)
(float)1 / (float)2
(float)(1 / 2)
(float)1.0 / 2.0
(float)x / (float)y
(float)x / (double)y
(int)floor(z) + (int)ceil(a)
```

問 50

以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`int x;`			
`float y, z;`			
`y = (int)2.2;`			
`x = 1.2 * y;`			
`z = (float)1 / 2 + y;`			
`z = (float)(1 / 2) + y;`			
`z = (float)(1 / 2 + y);`			
`z = z + (float)1 / 10;`			
`z = z + (float)1 / 10;`			
`z = z + (float)1 / 10;`			
`x = (int)((float)1 / 2 + y);`			
`x = x + y;`			

8. 型の表現できる範囲

数学では、実数とは無限大から無限小まで任意の精度の数を指す。

しかしプログラムでの実数型 (float,double) は、無限大や無限小や非常に大きい数や非常に小さい数を表すことはできない。

float 型は整数部分 1 桁と小数部分 6 桁を合計して 7 桁まで、double 型は整数部分 1 桁と小数部分 14 桁の合計 15 桁までの実数を表現する。これを**仮数部**と呼ぶ。

これに、10 の冪乗をかけ、実数を表現する。float なら $10^{-38} \sim 10^{38}$、double なら $10^{-308} \sim 10^{308}$ をの範囲の数値（**指数部**と呼ぶ）を仮数部にかける。

普通の実数は、仮数部と指数部をかけた形で表現する。仮数部と指数部はともに負の数も表現できる。

> 例：$0.002234(2.234 \times 10^{-3})$ や $22340(2.2340 \times 10^{4})$ は float でも double でも表現できる。
> $0.123456789(1.23456789 \times 10^{-1})$ は double では表現できるが、float ではすべてを表すことができず、有効数字以下は切捨てられて $0.1234567(1.234567 \times 10^{-1})$ となる。

double 型は、float 型よりほぼ倍の精度で実数を表現できるため、倍精度（double precision）と呼ぶ。double 型の名前の由来である。

問 51

次の実数は float 型と double 型で表現できるか。できれば○を、できなければ×を書け。

	float	double
1.2345		
0.001245		
123.45		
123.45×10^{42}		
0.000000002468		
0.00000000246802468		
$0.000000002468 \times 10^{-35}$		
1234.56×10^{-20}		
12345678901234567890		
$12345678.90 \times 10^{-200}$		

問 52

プログラム中で実数を書く場合、10 の冪乗を e で表し、$10^7 = 1 \times 10^7$ を `1e+7` と書く。

> 例：$0.00223 = 2.23 \times 10^{-3}$ は `2.23e-3`、20×10^7 を `20e7` または `20e+7` と書く。

次の実数は float 型と double 型で表現できるか。できれば○を、できなければ×を書け。

	float	double
`1.2345`		
`0.001245`		
`123.45`		
`123.45e42`		
`0.000000002468`		
`0.00000000246802468`		
`0.000000002468e-35`		
`1234.56e-20`		
`12345678901234567890`		
`12345678.90e-200`		

問 53

float 型で表せない範囲の double 型の数値を float 型の変数に代入したり、float 型に変換したりすると、その値は不定になる。表せる範囲であれば、値は変わらない。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`float x;`			
`double y, z;`			
`y = 1.23e20;`			
`z = 1.0e+20;`			
`x = y + z;`			
`y = x * 2;`			
`x = y * z;`			
`y = x / 2;`			

一般に、代入や型の変換で、代入先の変数の型や変換後の型が表せない数値の場合、値は不定になる。

問 54

float 型と float 型の演算の評価結果が float 型で表せない範囲の数値である場合、その値は不定になる。表せる範囲であれば、値は変わらない。

double 型も同様である。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`float x, y;`			
`double z;`			
`x = 1.2e20;`			
`y = 2.0e20;`			
`z = x * y;`			
`z = 1.2e200;`			
`z = z * 2.0e+200 + x;`			
`x = x + 2.3;`			

以後も、変数の値が不定であれば「不定」と書け。

問 55

int 型も表現できる整数の範囲に限りがある。通常は −2147483648 から 2147483647 までの整数である。

int 型で表せない範囲の float 型や double 型の実数値を int 型の変数に代入したり、int 型に変換したりすると、その値は不定になる。表せる範囲であれば、小数部分が切り捨てられた整数になる。int 型と int 型の演算の評価結果が int 型で表せない範囲の数値である場合、その値は不定になる。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`float x;`			
`double y;`			
`int z;`			
`y = 1000000000.2;`			
`z = y;`			
`z = y * 3;`			
`x = -1.23e+10;`			
`z = x * 2 + 10;`			
`z = (int)(y * 4);`			

問 56

正の整数だけを扱う場合には、専用の整数型がある。unsigned int 型（符号無し unsigned）である。unsigned int も表現できる整数の範囲に限りがあり、0 から 4294967295 までの整数である。

unsigned int 型で表せない範囲の実数値や int 型の整数値（負の数なども含まれる）を unsigned int 型の変数に代入したり、unsigned int 型に変換したりした結果は、不定である。unsigned int 型と unsigned int 型の演算の評価結果が unsigned int 型で表せない範囲の数値である場合、その値は不定になる。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`int x;`			
`double y;`			
`unsigned int z;`			
`y = 1000000000.2;`			
`z = y * 3;`			
`z = y * y;`			
`x = 10;`			
`z = x * 2 - 30;`			
`z = (unsigned int)(x * 2 - 30);`			

問 57

小さな整数だけを扱う場合には、専用の整数型がある。short int 型と unsigned short int 型である。short int 型は −32768～32767、unsigned short int 型は 0～65535 までの整数を表現できる。int や unsigned int と同様に、表現できない範囲の数値を代入された場合には不定になる。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`short int x;`			
`double y;`			
`unsigned int z;`			
`x = 100;`			
`y = 1000.0;`			
`x = y * 3000;`			
`x = 100;`			
`z = y * y;`			
`x = z;`			
`x = 100;`			
`z = x - 200;`			

問 58

整数型には次の三種類がある。

int 型 は実際には signed int 型（符号付き signed）の省略形である。signed 型は signed int 型の省略形である。unsigned 型は unsigned int 型の省略形である。

short 型 は short int 型の省略形である。short int 型は signed short int 型の省略形である。unsigned short 型は unsigned short int 型の省略形である。

long 型 は long int 型の省略形である。long int 型は signed long int 型の省略形である。unsigned long 型は unsigned long int 型の省略形である。

次の表にある型名の省略形の正式な型名を書け。また、正式な型名の最も短い省略形を書け。

型名の省略形	正式な型名
`short int`	
`unsigned short`	
`int`	
`unsigned`	
`long`	
`unsigned long`	
	`signed short int`
	`signed long int`
	`unsigned short int`

問 59

全ての整数型の数値は double 型で表現できる。しかし、すべての数値を double 型にすることは適切ではない。

その理由は、整数型の変数や数値が必要なときがあることと、その変数が何の数値を表しているのかが分からなくなるからである。

例：物の個数は 0 個,1 個,2 個,... であるので、unsigned int 型で表す。

次の数値を表現するのに最も適切な型を書け。

表す数値	型
りんごの個数（個）	`unsigned short int`
銀行口座の残金（円）	
切手一枚の値段（円）	
駅からの距離 (km)	
100m 走のタイム (s)	
気温 (°C)	
西暦（年）	
身長 (cm)	
体重 (kg)	
先週と今週の体重の差 (kg)	
自然数 n	
虚数の実部	
国家予算（約 64 兆円）	
新潟市の人口（約 50 万人）	
巻町の町税収入（約 27 億円）	

9. 文字型

C 言語では、文字型という型がある（char 型, “character” の頭文字に由来）。これは半角文字のアルファベットや数字や記号などの一文字だけを表現する（全角の日本語文字は不可）。文字「a」を式の中で表すには、「'」（クォート quote, shift+7 キー）で括って「'a'」と書く。

問 60

文字型を表している式には○を、間違っているものには×を書け。

式	正誤
'b'	
'h'	
'zz'	
' B'	
'#'	
‘d’	
"d"	
'&'	
'B'	
’B ’	
'あ'	

問 61

''で括られた文字の評価結果は、整数になる。つまり、文字と整数が対応している。

例：'a' の評価結果は整数で 97 である。

この文字と整数の対応はあらかじめ決められており、変更できない。ただし全くでたらめな整数と対応しているわけではなく、例えば a~z は通し番号、0~9 は通し番号、A~Z は通し番号になっている。

例：'b' の評価結果は整数で 98 である。

注意：'b' が 98 であることを丸暗記するのではなく、「文字（数字）」と「数値」の違いをはっきり理解すること。

次の式の評価結果を書け。

式	評価結果
'a'	97
'b'	
'f'	
'0'	
'5'	53
'9'	
'A'	
'G'	
'H'	72

問 62

文字型（char 型）の変数を宣言するには、「`char x;`」と書く。整数型や実数型の変数と同様に、評価結果を代入することができる。

文字型の変数が保持する値は整数であり、その評価結果も整数である。

以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値
`char x;`	
`x = 'a';`	
`x = 'b';`	
`x = 97;`	
`x = 'a' + 1;`	
`x = 'b' - 'a';`	
`x = 'b' / 2 + 1;`	

問 63

文字型（char 型）は、実際には整数を表す。char 型にも符号付きと符号無しの型がある。signed char 型は −127～128 の整数を、unsigned char 型は 0～255 の整数を表現する。

整数型（int 型）の整数や実数型の数値と代入や型変換をした場合、unsigned char 型または signed char 型で表せない範囲の数値である場合、その値は不定になる。

以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	z の値
`unsigned char x;`			
`int y,z;`			
`x = 'a';`			
`y = 97;`			
`x = y;`			
`z = 1;`			
`x = y + z;`			
`x = x * 5;`			
`x = y - 120;`			
`x = 'b' - 100 + z * 10;`			

問 64

以下の変数宣言と代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値と、その整数値が表す文字を書け。対応する文字がなければ横線―を書け。

代入文	x の値	x の値が表す文字
`unsigned char x;`		`------`
`int y;`		`------`
`x = 'a';`		
`x = x + 1;`		
`x = x + 1;`		
`x = x + 1;`		
`x = 'a' + 3;`		
`x = 'f' - 'b';`		
`x = 'A';`	65	
`x = 'A' + 3;`		
`x = 'F' - 'B';`		
`x = 'a' - 'A' + 'A';`		
`x = 'a' - 'A' + 'C';`		
`x = 'b' - 'B' + 'C';`		
`x = 'a' - ('a' - 'A');`		
`y = 'a' - 'A';`		
`x = 'c' - y;`		
`x = 'C' + y;`		

問 65

次の式の評価結果を書け。

式	評価結果
`'a' == 'a'`	
`'b' < 'a'`	
`'0' + 1 <= '0'`	
`'5' != 5`	
`'9' - 9 == '1' - 1`	
`'a' > 100`	

問 66

代入文「`x = 'b'`」によって文字型変数 x に値が代入されているとき、次の式を評価せよ。

式	評価結果
`x == 'a'`	
`'a' == x - 1`	
`x + 1 <= 'b'`	
`'e' - 'd' != x - 'a'`	
`'B' == x - ('a' - 'A')`	
`!('b' == x)`	

10. 表示するためのprintf関数

前章までは、式評価の結果は計算されるだけで、出力というものを考えていなかった。

出力するためには、関数 printf（表示関数 print function から）を使う。printf は評価結果を、指示された形式で画面上に表示（出力）する。

printf 関数は、引数の書き方によって何を出力するのかを指定する。

問 67

まず、引数を一つだけのとき、printf 関数は文字列を表示する。この場合、引数には「"」（ダブルクォート double quote, shift+2 キー）で括った文字列を与え、その中の部分（文字列）は画面にそのまま表示される。

改行を表す記号\n は、実際には表示されず、その部分で画面上の表示が改行される。printf で\n を表示することを改行すると言う。改行せずに連続して printf で表示すると、一行で表示される。

例：

```
printf("How are \nyou?");
printf("Fine, thank you.");
```

を実行すると、

```
How are
you?Fine, thank you.
```

と表示される。

以下の printf が上から順番に実行されるとき、printf が実行された時点で表示される内容を書け。

```
printf("This is");
printf("a test.\n Next,");
printf(" you say \n");
printf("hello world\n");
```

_____

_____

_____

問 68

以下の内容を出力する printf を書け。

```
I am a student.
This is a pen.
```

問 69

char 型の文字（例えば'a'）の評価結果は全て整数（'a' は 97）になる。式の中では文字はいつも整数であるが、逆に、人間がその数値が表す文字を見るには、つまり数値を文字にするにはどうしたらよいのか。

そのために、printf 関数に引数を二つ与える。

一つ目の引数は、どのような形式で（数字なのか文字なのか、実数なのか整数なのか、10 進数なのか 16 進数なのか）表示するかを指定するための文字列である。二つ目の引数は、評価結果を表示する式である。

char 型の整数が表す文字を表示したい場合には、第一引数を"%c"と書く（c は char 型の c）。char 型の整数を数字として表示したい場合には、第一引数を"%d"と書く（d は 10 進数 decimal の d）。つまり %c もしくは %d を「"」（ダブルクォート, shit+2 キー）で括る。

例：printf("%c", 97); は文字 a を表示し、printf("%d", 97); は数字 97 を表示する。

以下の変数宣言と代入文と printf が上から順番に実行されるとき、printf が実行された時点で表示される内容を書け。

代入文	x の値	表示される内容
unsigned char x;		―――
x = 'a';	97	―――
printf("%c", x);		
printf("%c", 'b');		
printf("%c", 98);		
x = x + 2;		―――
printf("%d", 'b');		
printf("%d", x + 2);		
printf("%c", x + 2);		
printf("%c", x - 'A' + 'B');		
x = '0';	48	―――
printf("%c", '0');		
printf("%d", '0');		
printf("%d", '0' - 48);		
printf("%d", '1' - 48);		
printf("%d", x - 48);		
x = x + 1;		―――
printf("%d", x - 48);		
printf("%c", x);		

問 70

変数 a と d の値を表示したい。以下のプログラムの間違いを指摘せよ。

```
char a, d;
printf("%d, a");
printf("d");
```

問 71

printf 関数の第一引数に%c などを含む文字列を書き、第二引数に表示する式を書くと、%c の部分は、式の評価結果に置き換えられて文字として表示され、それ以外の部分は文字列がそのまま表示される。%d の部分は、式の評価結果に置き換えられて数値として表示さる。

例：

```
printf("How are %d you?\n", 10);
printf("%c am fine, thank you.", 'I');
```

を実行すると、

```
How are 10 you?
I am fine, thank you.
```

と表示される。

以下の printf が上から順番に実行されるとき、printf が実行された時点で表示される内容を書け。

	表示される内容
printf("a%ccd\n", 'b');	
printf("123%d456\n", 87);	
printf("a is %d.\n", 'a');	
printf("a is %c.\n", 'a');	
printf("0 is %d.\n", 48);	
printf("0 is %d.\n", '0');	
printf("0 is %c.\n", '0');	

問 72

char 型変数 a の値を文字で表示するための printf を書け。

問 73

何が表示されるか。

```
char c1, c2, c3;
c1 = 'a';
c2 = 'c';
c3 = c2 - c1;
printf("c3 is %c\n", 'g' + c3);
```

問 74

printf 関数の第一引数に%c などを複数個含む文字列を書くと、複数の式を表示することができる。個数の分だけ第二引数、第三引数…に、式を書く。
第一引数内の%c などの順番と、第二引数以降の変数の順番は、対応している。

例：

```
printf("%c b c d %c f g %d h\n", 'A', 'E', 88);
```

を実行すると、

```
A b c d E f g 88 h
```

と表示される。つまり、最初の%c は'A' に対応し、二番目の%c は'E' に対応し、%d は 88 に対応する。

以下の printf が上から順番に実行されるとき、printf が実行された時点で表示される内容を書け。

	表示される内容
`printf("%c%c%c\n", 'a', 'b', 'c');`	
`printf("a is %d and %c.\n", 'a', 'a');`	

問 75

次のプログラムを実行すると何が表示されるか。

```
char c = 'a';
printf("%c %d\n", c, c - 'a');
```

問 76

int 型の整数や実数型を表示するためには、printf の第一引数の文字列に書く内容を変えなければならない。

- signed int 型の整数を表示したい場合には、"%d"と書く。
- unsigned int 型の整数を表示したい場合には、"%u"と書く。

- float 型または double 型の実数を表示したい場合には、"%f"と書き、（小数点第 7 桁を四捨五入して）小数点第 6 桁までが表示される。また、例えば"%.5f"と書くと（小数点第 6 桁を四捨五入して）小数点第 5 桁までが表示される。

  例:
  `printf("%f\n", 123.45);` の表示結果は `123.450000`
  `printf("%.1f\n", 123.45);` の表示結果は `123.5`

第一引数の指定と評価結果の型が異なる場合には、出力内容は不定である。

以下の `printf` が上から順番に実行されるとき、`printf` が実行された時点で表示される内容を書け。

	表示される内容
`printf("%f cm\n", 12.34);`	
`printf("%.1f cm\n", 12.34);`	
`printf("%.0f cm\n", 12.34);`	
`printf("%d\n", 52 - 55);`	
`printf("%d\n", 55 - 52);`	
`printf("%u\n", 55 - 52);`	
`printf("%u\n", 52 - 55);`	
`printf("%f\n", 55 - 52);`	
`printf("%f\n", 55 - 52.);`	
`printf("%d\n", 55 - 52.);`	
`printf("%.0f\n", 55 - 52.);`	
`printf("%d %f %d\n", 1, 2., 3);`	
`printf("%d %f %d\n", 1, 2, 3);`	

問 77

以下の代入文と `printf` が上から順番に実行されるとき、`printf` が実行された時点で表示される内容を書け。

```
float w1, w2, h1, h2;
int y1, y2;
w1 = 52.2;
w2 = 64.2;
h1 = 172.5;
h2 = 173.5;
y1 = 1999;
y2 = 2003;
printf("year : weight, height\n");
printf("%d : %.1fkg, %.1fcm\n", y1, w1, h1);
printf("%d : %.1fkg, %.1fcm\n", y2, w2, h2);
printf("grows : %.1fkg, %.1fcm total\n", w2-w1, h2-h1);
printf("grows : %.2fkg, ", (w2-w1)/(y2-y1));
printf("%.2fcm per year\n", (h2-h1)/(y2-y1));
```

_____
_____
_____
_____
_____
_____
_____

問 78

何が表示されるか。

```
int a, b;
a = 2;
b = 4;
a = b = 5;
printf("%d %d\n", a, b);
```

問 79

何が表示されるか。

```
int a, b, c;
a = 2;
b = 3;
c = 4;
printf("%d %d %d\n", a, b, c);
b = c;
printf("%d %d %d\n", a, b, c);
c = a;
printf("%d %d %d\n", a, b, c);
```

問 80

次のプログラム

```
int a, b, c;
a = 8;
b = 12;
c = 97;
printf(_________________, a, b, c);
```

を実行すると

```
8 12 a
```

と表示された。下線部を埋めよ。

問 81

次のプログラム

```
int a, b;
a = 2;
b = 3;
printf(’a+b= %d\n’, a+b);
```

には間違いがある。訂正せよ。

問 82

次のプログラム

```
int a, b;
a = 2;
b = 3;
prinf("a+b= %d\n", a+b);
```

には間違いがある。訂正せよ。

問 83

次のプログラム終了後には何が表示されるか。

```
int a, b;
a = 3;
b = 4;
printf("%d\n", a = b);
printf("%d\n", a);
```

11. 配列

たくさんの変数が必要な場合、それらの変数を全て宣言するのは大変である。たとえば、数列 $\{a_n\}$ の 100 項目までを表す C 言語の変数を宣言する場合、「`int a_1,a_2,a_3,a_4,`（途中省略）`,a_100`」のように、必要な分だけ書かなければならず、面倒である。

このような場合、ある名前の変数に番号（添字 index）を付けて、一度に宣言することができる。これを**配列 (array)** と呼ぶ。たとえば、`a` という名前の 100 個の変数からなる配列を宣言するには `int a[100];` とする。こうすると、`a[0]` から `a[99]` までの 100 個の変数を宣言したことになる。

変数名の直後の括弧「`[`」と「`]`」の間には、添字（番号を表す整数、または評価結果の型が整数になる式）を書く。

問 84

配列を構成する `a[0],a[1],a[2]` などの一つ一つの変数を、**要素**と呼ぶ。要素の数は宣言するときに `[]` 内に書く。

> 注意：宣言のときの `[ ]` には要素の総数を書き、式の中で用いる `[ ]` には要素の番号を書く。場所によって `[ ]` の意味が違うことに注意。

通常の変数と同様、初期化していない要素の値は不定である。
各変数の値を書け。

	`a[0]` の値	`a[1]` の値	`a[2]` の値
`int a[3];`			
`a[0] = 3;`			
`a[1] = -5;`			
`a[2] = 10;`			
`a[3-2] = 10;`			
`a[8/4] = 10;`			

問 85

配列の**添字の範囲は、0 から「要素の数-1」まで**である。この範囲内に添字がない場合には、評価結果は不定である。

> 例：`int a[100];` と宣言してある場合、`a[200]` や `a[-2]` や `a[100]` の評価結果は不定である。

表示される内容を書け。値が不定の場合は――を書け。

	表示される内容

`int a[3];`
`float b[10];`
`a[0] = 1;`
`a[2] = 2;`
	――
	――
	――
	――

`printf("%d %d\n", a[0], a[2]);`	
`printf("%d %d\n", a[0], a[1]);`	
`b[1] = a[0] + 1.5;` `b[2] = b[1] * 2;`	―― ――
`printf("%f %f\n", b[1], b[2] - 1);`	
`b[1] = 10 - b[2];` `b[0] = 10 - b[3];`	―― ――
`printf("%f %f\n", b[0], b[1]);`	

問 86

1年生から4年生までの各学年の学生の人数を格納するための配列 `studentNumber` を定義せよ。

また、毎日の1時間毎の平均気温を格納するための配列 `temperature` を定義せよ。

問 87

式中の配列の添字には、整数だけでなく、評価結果が整数型である式を書くことができる。その結果、添字が0から「要素の数-1」にない場合、評価結果は不定である。

各変数の値を書け。

	`a[0]` の値	`a[1]` の値	`a[2]` の値
`int a[3], i;`			
`i = 0;`			
`a[i] = 3;`			
`i = i + 1;`			
`a[i] = -5;`			
`i = i + 1;`			
`a[i] = 10;`			
`i = 0;`			
`a[i+1] = a[i];`			
`i = i + 2;`			
`a[i] = a[i-1];`			
`a[i] = a[i+1];`			

問 88

何が表示されるか。

```
int a[3];
a[0] = 1;
a[a[0]] = 2;
a[a[1]] = 3;
a[a[a[0]]] = 4;
printf("%d %d %d\n", a[0], a[1], a[2]);
```

問 89

char 型（文字型）の配列は、「文字列」と呼ばれる。

> 例：`char name[11];` と宣言した場合、11 文字文の文字型の配列が確保される。つまり、11 文字分の文字列を表現できる。

`printf` で文字列を表示する場合には、第一引数内に`%s` を書き、第二引数以降に配列名だけを書く（`[]` は必要ない）。この`%s` と第二引数以降の対応は、`%d` や`%f` などと同じである。

ただし、文字列（文字型の配列）の最後の要素の値は「0」でなくてはならない（文字`'0'` ではなく数値の 0 である）。最後の要素（つまり数値の「0」）は `printf` では表示されない。`printf` は、数値の「0」の直前までの内容を表示する。

もし最後の要素の値が `0` でない場合には、最後の要素のひとつ前の要素までは表示されるが、それ以降の出力は不定である。

> 例：`char name[11];` と宣言した場合、11 文字文の文字型の配列が確保されるが、`printf` で表示する場合には 10 文字分の文字列を表現する（最後の要素は 0 でなければならないため）。

以下のプログラムを実行した場合、何が表示されるか。

```
char b[4];
b[0] = 'a';
b[1] = 'b';
b[2] = 'c';
b[3] = 0;
printf("b is %s.\n", b);
printf("%c %c\n", b[2], b[0]);
b[0] = b[0] + 1;
b[1] = b[1] + 1;
b[2] = b[2] + 1;
printf("b is %s.\n", b);
```

問 90

何が表示されるか。

```
int a[3];
char b[3];
b[0] = ’0’;
b[1] = ’8’;
b[2] = ’2’;
a[0] = b[0] - ’0’;
a[1] = b[1] - ’0’;
a[2] = b[2] - ’0’;
printf("%d %d %d\n", a[0], a[1], a[2]);
```

問 91

何が表示されるか。

```
int a[3];
char b[3];
a[0] = 2;
a[1] = 3;
a[2] = 1;
b[0] = ’a’;
b[1] = ’b’;
b[2] = ’c’;
printf("%c %c %c\n", b[0], b[1], b[2]);
printf("%c %c %c\n", b[a[0]-1], b[a[1]-1], b[a[2]-1]);
printf("%c\n", b[a[a[a[2]]-1]-1]);
```

12. 宣言時の初期化

配列の要素が増えると、配列の各要素を初期化するために沢山の代入文が必要となる。これは面倒なので、宣言と同時に初期化することができる。宣言するときに、配列の要素の数だけ、値を中括弧{ }で括くる。

例：
```
int a[2];
a[0] = 1;
a[1] = 4;
```
のかわりに
```
int a[2] = {1, 4};
```
と書くことができる。

宣言以外の場所でこのような書き方はできない。つまり普通の代入文ではこのような書き方は許されない。

中括弧{ }の中の値の数は、配列の要素の数よりも多く書くことはできない。また逆に少ない場合には、残りの要素は初期化されない（つまり値は不定になる）。

問 92

各変数の値を書け。

	a[0]	a[1]	a[2]
int a[3] = {3, -5, 10};			
a[0] = a[1] + 10;			
a[7-6] = 10;			
a[10/5] = 8;			

問 93

以下の変数宣言と代入文を、宣言時の初期化に書き直せ。

```
float value[2];
value[1] = 4.5;
value[0] = 1.2;
```

問 94

何が表示されるか。

```
double d[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
printf("%f\n", d[1]);
```

問 95

通常の変数も、宣言時に初期化することができる。

何が表示されるか。

```
int a[5] = {1, 2, 3, 4, 5}, i = 5;
printf("%d\n", a[i-1]);
```

問 96

何が表示されるか。

```
int a[3] = {2, 3, 1}, x = 3, b[3] = {5, 4, 8};
printf("%d %d %d\n", b[0], b[1], b[2]);
printf("%d %d %d\n", b[a[0]-1], b[a[1]-1], b[a[2]-1]);
printf("%d %d %d\n", b[x - 3], b[x - 2], b[x - 1]);
```

問 97

何が表示されるか。

```
int a = 2, b = 3, c = 4;
printf("%d %d %d\n", a, b, c);
a = b;
printf("%d %d %d\n", a, b, c);
b = c;
printf("%d %d %d\n", a, b, c);
c = a;
printf("%d %d %d\n", a, b, c);
```

問 98

各変数の値を書け。

	a[0]	a[1]	b[0]	b[1]
int a[2]={4, 2};				
int b[2]={1, 0};				
a[0] = 1;				
b[0] = a[a[0]];				
b[1] = a[b[1]]+1;				
b[1] = a[b[0]-2];				

問 99

文字型配列の変数の初期化も同様に初期化できる。
何が表示されるか。

```
int x = 1;
char a[3] = {'a', 'b', 'c'};
printf("%c %c %c\n", a[0], a[1], a[2]);
printf("%c %c %c\n", a[x - 1], a[x], a[x + 1]);
```

問 100

文字型配列の変数は、特殊な初期化の方法がある。ダブルクォーテーション""で複数の文字（文字列）を括る。ただし、宣言する配列の要素の数は、文字列の文字数よりも 1 だけ多くなければならない（最後の要素には、自動的に数値の「0」が代入されるため）。

> 例：char a[4] = "abc"; と変数宣言した場合、
> a[0] には'a' が、a[1] には'b' が、
> a[2] には'c' が、a[3] には数値の「0」が、それぞれ代入される。

何が表示されるか。

```
int x = 1;
char a[4] = "abc";
printf("%c %c %c\n", a[0], a[1], a[2]);
printf("%d %d %d %d\n", a[0], a[1], a[2], a[3]);
printf("%c %c %c\n", a[x - 1], a[x], a[x + 1]);
printf("%s\n", a);
```

問 101

何が表示されるか。

```
int i = 0, b[3] = {0, 1, 2};
char a[4] = "abc";
i = 0;
printf("a[%d] = %c\n", i, a[i]);
i = i + 1;
printf("a[%d] = %c\n", i, a[i]);
i = i + 1;
printf("a[%d] = %c\n", i, a[i]);
i = 0;
printf("a[%d] = %c\n", b[i], a[b[i]]);
b[i] = b[i+1];
printf("a[%d] = %c\n", b[i], a[b[i]]);
b[i] = b[i+2];
printf("a[%d] = %c\n", b[i], a[b[i]]);
```

問 102

何が表示されるか。

```
int a[9] = {1,2,3,4,5,6,7,8,9};
char b[9] = "abcdefgh";
printf("%d", a[0]);
printf("%d", a[a[0]]);
printf("%d", a[a[a[0]]]);
printf("%d\n", a[a[a[a[a[a[a[0]]]]]]]);
printf("%c", b[a[0]]);
printf("%c", b[a[a[0]]]);
printf("%c", b[a[a[a[0]]]]);
printf("%c\n", b[a[a[a[a[0]]]]]);
```

問 103

何が表示されるか。

```
int a[9] = {1,2,3,4,5,6,7,8,9}, i;
char b[9] = "abcdefgh";
i = a[0];
printf("%d", i);
i = a[i];
printf("%d\n", i);
i = a[i];
printf("%d", i);
i = a[0];
printf("%c\n", b[i]);
i = a[i];
printf("%c", b[i]);
i = a[i];
printf("%c\n", b[i]);
```

問 104

次のプログラムがある。

```
char name[11] = "suzuki rie";
printf("%c%c\n", name[_______], name[_______]);
```

これを実行したところ、次のように表示された。

```
kr
```

プログラム中の下線部を埋めよ。

問 105

次のプログラムがある。

```
char name[10] = "Takeshi K";
printf("%c%c\n", name[_______], name[_______]);
```

これを実行したところ、次のように表示された。

```
KT
```

プログラム中の下線部を埋めよ。

問 106

次のプログラムがある。

```
char c[4] = "_________________";
printf("%c%c%c\n", c[1], c[2], c[0]);
```

これを実行したところ、次のように表示された。

```
abc
```

プログラム中の下線部を埋めよ。

13. 繰り返し：whileループ

変数宣言や代入文やprintf関数などを、まとめて文と呼ぶ。プログラムとは、文をセミコロン「;」で区切って並べたものである。
プログラムの実行（プログラムを実行する）とは、セミコロン「;」で区切られた文を順に評価することである。

プログラムの一部を、全く同じ書き方で繰り返して書ける場合には、その部分をwhile文でまとめることができる。while文は次の様に書く。

```
while(条件となる式){
  繰り返す文の並び
}
```

「条件となる式」を条件式と呼ぶ。条件式は小括弧「( )」で括り、繰り返す文の並びは中括弧「{}」で括る。
条件式を評価し、その評価結果が真（0以外、問12参照）であるとき、「繰り返す文の並び」を実行する。そして再び条件式の評価を行う。
条件式の評価結果が偽（0）ならば、while文を終了する。

例：次のような繰り返し

```
int i = 0;
printf("%d\n", i);
i = i + 1;
printf("%d\n", i);
i = i + 1;
printf("%d\n", i);
i = i + 1;
printf("end\n");
```

は、次の書き方

```
int i = 0;
while(i < 3){
  printf("%d\n", i);
  i = i + 1;
}
printf("end\n");
```

と等価である。

whileによるプログラムの繰り返しをwhileループと呼ぶ。

問 107

$5 \times 1 = 5$ を計算するには、＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿を＿＿＿＿＿＿＿＿回繰り返せばよい。

変数宣言は `int i = 0;`、条件式は `i < 4`、繰り返す文は `i = i + 1;` である `while` ループを書け。

問 108

`while` 文では、「条件式の評価」と「繰り返す文の並びの実行」を順に行う。文を実行している間に条件式を満たしているかどうかは関係がない。

次のプログラムでは、条件式を評価し、三つの文を実行し、条件式を評価し、ということを繰り返している。

```
int a = 0, b = 0;
while(a < 5){
  a = a + 5;
  b = b + 3;
  a = a - 2;
}
```

この `while` ループが終了したときの `a,b` の値を答えよ。また、一つ一つの繰り返しのときの `a,b` の値の変化と、条件式「`a < 5`」の評価結果を答えよ。

問 109

何が表示されるか。

```
int i = 0;
while(i < 10){
  printf("%d ", i);
  i = i + 1;
}
```

何が表示されるか。

```
int i = 0;
while(i < 10){
  i = i + 1;
  printf("%d ", i);
}
```

それぞれの繰り返しのときの `i` の値の変化に注目して、上の二つのプログラムはどのように違うのかを述べよ。

問 110

何が表示されるか。また、これは何の計算をしているのかを述べよ。

```
int i = 0;
while(i < 66){
  printf("%d ", i);
  i = i + 7;
}
```

問 111

次のプログラムがある。これを、while 文を使わないプログラムに書き直せ。

```
int i = 0, n = 0;
while(i < 3){
  n = n + i;
  i = i + 1;
  printf("%d %d\n", n, i);
}
```

問 112

次のプログラムがある。これを、while 文を使ったプログラムに書き直せ。

```
int i = 0, a[3];
a[i] = i * 2;
printf("%d %d\n", i, a[i]);
i = i + 1;
a[i] = i * 2;
printf("%d %d\n", i, a[i]);
i = i + 1;
a[i] = i * 2;
printf("%d %d\n", i, a[i]);
i = i + 1;
```

問 113

次のプログラムを実行すると何が表示されるか。

```
char c = 'b';
while(c != 'f'){
  printf("%c", c);
  c = c + 1;
}
```

問 114

何が表示されるか。

```
int i = 0;
while(i < 5){
  printf("%d ", i);
  i = i + 1;
}
printf("%d ", i);
while(i >= 0){
  printf("%d ", i);
  i = i - 1;
}
```

問 115

何が表示されるか。

```
int i = 0, a[5] = {2, 4, 6, 8, 10};
while(i < 5){
  printf("%d ", a[i]);
  i = i + 1;
}
while(i > 0){
  i = i - 1;
  printf("%d ", a[i] - 1);
}
```

問 116

何が表示されるか。

```
int i = 0, b[3] = {0, 1, 2};
char a[4] = "abc";
while(i < 3){
  printf("a[%d] = %c\n", i, a[i]);
```

```
  i = i + 1;
}
i = 0;
while(i < 3){
  printf("a[%d] = %c\n", b[i], a[b[i]]);
  i = i + 1;
}
```

問 117

何が表示されるか。また、これは何の計算をしているのかを述べよ。

```
int i = 0, a[6] = {2,4,6,8,10,7};
while(a[i] < 10){
  printf("%d ", a[i]);
  i = i + 1;
}
```

問 118

何が表示されるか。また、これは何の計算をしているのかを述べよ。

```
int i = 12;
char b[14] = "!gninrom doog";
while(b[i] != '!'){
  printf("%c", b[i]);
  i = i - 1;
}
printf(".\n");
```

問 119

次のプログラムがある。

```
int i = 0, a[7] = {1,3,5,7,9,11,8};
while(a[i] < 10){
  printf("%d ", a[i]);
  ____________________
}
```

これを実行したときに、次のように表示したい。

```
1 3 5 7 9
```

プログラムの下線部を埋めよ。

問 120

次のプログラムがある。

```
int i = 1;
while(i <= 100){
  printf("%d ", i);
  _______________
}
```

これを実行すると、100 以下の全ての奇数を表示するようにしたい。プログラムの下線部を埋めよ。

問 121

次のプログラムがある。

```
int i = 0, a[5] = {3, 5, 2, 6, 9};
while( _________________ ){
  printf("%d ", a[i]);
  i = i + 1;
}
```

これを実行すると、配列 a の全ての要素を表示するようにしたい。プログラムの下線部を埋めよ。

問 122

次のプログラムがある。

```
int i;
char c[5] = "abcd";
________
while( _________________ ){
  printf(_________________, _________________);
  _______________
}
```

これを実行すると、配列 c の全ての要素を連続して表示するようにしたい。プログラムの下線部を埋めよ。

問 123

次の配列 f がある。

```
float f[4] = {1.23, 4.56, 78.9, 10.11};
```

この配列 f の全ての要素を表示するプログラムを while 文を用いて書け。

問 124

次のプログラムがある。

```
int a[5] = {4, 5, 6, 7, 8}, b[5], i = 0;
while(i < 5){
  b[i] = a[i] + 4;
  i = i + 1;
}
```

_____
_____
_____
_____
_____

このあとに配列 b の全ての要素を表示するようにプログラムの続きを書け。また、その結果何が表示されるか。

問 125

何が表示されるか。また、これは何の計算をしているのかを、それぞれの繰り返しのときの i と sum の値の変化に注目して、述べよ。

```
int i = 0, sum = 0;
while(i < 10){
  i = i + 1;
  sum = sum + i;
}
printf("%d\n", sum);
```

問 126

何が表示されるか。また、これは何の計算をしているのかを、それぞれの繰り返しのときの i と factorial の値の変化に注目して、述べよ。

```
int i = 1, factorial = 1;
while(i < 4){
  factorial = factorial * i;
  i = i + 1;
}
printf("%d\n", factorial);
```

問 127

while 文を使って、100 以下の 7 の倍数を全て表示するプログラムを書け。

_____
_____
_____
_____
_____

問 128

次のプログラムがある。これは、ある数列の 1000 項目までを表示するものである。数列の第 n 項を（数学の数式で）書け。

```
int n = 1;
while(n <= 1000){
  printf("%f\n", (2.0 * n - 3.0) / (n * n + 1));
  n = n + 1;
}
```

問 129

次のプログラムがある。

```
int n = 1
float sum = 0;
while(n <= 1000){
  sum = ____________________________________
  printf("%f\n", sum);
  n = n + 1;
}
```

これは、次の級数の 1000 項目までの和を表示するものである。プログラム中の下線部を埋めよ。

$$\sum_{n=1}^{1000} \left(\frac{1}{n^2} + 2n \right)$$

問 130

次のプログラムがある。

```
int i= [     ] , n=0;
while(i < 150){
   n = [                ] ;
   i = i + 1;
}
```

これが表す数式は次のようなものである。
空欄を埋めよ。

$$n = \sum_{i=-3}^{\square} (2i+4)$$

問 131

次のプログラムがある。

```
int i=-3, n=0;
while(i != 128){
  n = n + 1;
  i = i + 1;
}
```

これが表す数式は次のようなものである。
空欄を埋めよ。

$$n = \sum_{i=\square}^{\square} \square$$

問 132

次の三つのプログラムがある。

```
int i = 0, a[6] = {0, 0, 0, 1, 2, 3};
while(a[i] == 0){
  printf("%d ", a[i]);
  i = i + 1;
}
-------------------------------------
int i = 0, a[6] = {0, 0, 0, 1, 2, 3};
while(a[i] = 0){
  printf("%d ", a[i]);
  i = i + 1;
}
-------------------------------------
int i = 0, a[6] = {1, 2, 3, 0, 0, 0};
while(a[i]){
  printf("%d ", a[i]);
  i = i + 1;
}
```

それぞれ実行したときに、何が表示されるか。また上の三つのプログラムはどのように違うのか、なぜ表示結果が違うのかの理由を述べよ。

問 133

何が表示されるか。

```
int i = 0, a[6] = {11, 56, 34, 77, 39, 0};
while(a[i] != 0){
  printf("%d ", a[i]);
  i = i + 1;
}
```

問 134

何が表示されるか。

```
int a[5] = {4, 0, 1, 3, 2}, i = 0, j = 0;
char b[6] = "abcde";
while(j < 10){
  printf("%c ", b[i]);
  i = a[i];
  j = j + 1;
}
```

問 135

何が表示されるか。

```
int numbers[6] = {15, 13, 64, 37, 74, 62}, i = 1;
while(numbers[i] < 30 && i < 6){
  printf("%d\n", numbers[i]);
  i = i + 2;
}
```

問 136

配列 a の平均 (mean) $E[a]$ を表示したい。下線部を埋めよ。

```
int a[10] = {3, 4, 1, 6, 6, 2, 7, 2, 9, 9}, i = 0;
float mean = 0;
while(i < 10){
  mean = ____________________
  i = i + 1;
}
mean = ____________________
printf("mean = %f\n", mean);
```

問 137

配列 a の平均と分散 (variance) $V[a]$ を表示したい。下線部を埋めよ。

```
int a[10] = {3, 4, 1, 6, 6, 2, 7, 2, 9, 9}, i = 0;
float mean = 0, variance = 0;
while(i < 10){
  mean     = ____________________________
  variance = variance + a[i] * a[i];
  i = i + 1;
}
mean     = ____________________________
variance = ____________________________
printf("mean     = %f\n", mean);
printf("variance = %f\n", variance);
```

ここで、a の分散 $V[a]$ は $V[a] = E[a^2] - E[a]^2$ である。

14. 二重ループ

while ループの中に while を書いてもよい。これを二重ループと呼ぶ。外側の while 一回に付き、内側の while は毎回繰り返される。

例：

12:30 のように「時:分」という形式で時刻を表示する。

外側のループは、時を 0 から 23 まで変化させる。時を表す変数を hour とすると、そのループは次のようになる。

```
hour = 0;
while(hour < 24){
  ここで「hour:00」から「hour:59」まで表示する
  hour = hour + 1;
}
```

内側のループは、hour の値を固定している間に、分を 0 から 59 まで変化させる。分を表す変数を minute とすると、このループは次のようになる。

```
minute = 0;
while(minute < 60){
  ここで「hour:minute」を表示する
  minute = minute + 1;
}
```

二つのループを組み合わせると、次のようになる。

```
   int hour = 0, minute;
┌─ while(hour < 24){
│    minute = 0;
│ ┌  while(minute < 60){
│ │    printf("%d:%d\n", hour, minute);
│ │    minute = minute + 1;
│ └  }
│    hour = hour + 1;
└─ }
```

ここで左端の罫線は、それぞれ外側のループと内側のループの範囲の対応が分かるように描いてある。また、プログラム中でもループの範囲が明確になるように、左側に空白をあけているが、内側のループには外側のループよりも多く空白をあけている（これを**インデント**と呼ぶ）。

問 138

次のプログラムの左端に、それぞれ外側のループと内側のループの範囲の対応が分かるように罫線を描け。また、一つ一つの繰り返しのときの i,j の値の変化を考えて、何が表示されるかを答えよ。

```
int i, j;
i = 0;
while(i < 3){
  printf("i = %d\n", i);
  j = 0;
  while(j < 2){
    printf("i = %d, j = %d\n", i, j);
    j = j + 1;
  }
  i = i + 1;
}
```

問 139

一つ一つの繰り返しのときの a,i の値の変化を考えて、何が表示されるかを答えよ。

```
int i;
char a = 'a';
while(a < 'j'){
  i = 0;
  while(i < 5){
    printf("%c", a);
    a = a + 1;
    i = i + 1;
  }
  printf("\n");
}
```

問 140

配列 a の値を*で（例えば 3 なら***と）表示する次のプログラムがある。

```
int a[10] = {3, 4, 1, 6, 6, 2, 7, 2, 9, 9}, i = 0, k;
while( _____________ ){
  k = 0;
  printf("a[%d]: ", i);
  while( _____________ ){
    printf("*");
    k = k + 1;
  }
  printf("\n");
  i = i + 1;
}
```

出力結果は以下の様になる。

```
a[0]: ***
a[1]: ****
a[2]: *
a[3]: ******
a[4]: ******
a[5]: **
a[6]: *******
a[7]: **
a[8]: *********
a[9]: *********
```

どのような処理が（二重ループの）外側のループと内側のループになるのか、変数 i や k は何のために使われているのかを考えて、プログラム中の下線部を埋めよ。

問 141

何が表示されるか。

```
float x = 0;
while(x < 5){
  printf("f(%.1f) = %.1f\n", x, 2*x*x+1);
  x = x + 0.5;
}
```

問 142

次の二つのプログラムがある。

```
 float x = 0, y = 0;
 while(y < 3){
   x = 0;
   while(x < 3){
     printf("f(%.1f,%.1f) = %.1f\n", x, y, 2*x+y+1);
     x = x + 1;
   }
   y = y + 1;
 }
-----------------------------------------------------
 float x = 0, y = 0;
 while(y < 3){
   x = 0;
   while(x > 3){};
   printf("f(%.1f,%.1f) = %.1f\n", x, y, 2*x+y+1);
```

```
    x = x + 1;
    y = y + 1;
 }
```

それぞれ実行したときに、何が表示されるか。また上の二つのプログラムはどのように違うのか、なぜ表示結果が違うのかの理由を述べよ。

問 143

この章の冒頭の例を参考に、「分 : 秒」という形式で表示するプログラムを書け。また適切にインデントを付けよ。

問 144

この章の冒頭の例と前問を参考に、「時 : 分 : 秒」という形式で表示するプログラムを書け（これは三重ループになることに注意）。また適切にインデントを付けよ。

15. 無限ループ

while の条件式が常に真であるとき、while ループから抜け出すことができなくなる。これを無限ループと呼ぶ。これはよくあるプログラミング時の間違いである。

例：次の while ループは、条件式に用いている変数 i の値が変わらないため、終了することがない。

```
int i = 0;
while(i == 0){
  printf("%d\n", i);
}
```

無限ループは、条件式の評価結果が繰り返しによって変わるべきところが変わらなかったり、永久に満たされない条件になっていたり、比較演算子（>,<,>=,<=）を使うべきところに否定演算子（!）を使ったりするような、プログラム上の間違いによって起こる。

問 145

配列 a の全ての要素を表示したいが、無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
int i = 0, int a[6] = {0, 0, 0, 1, 2, 3};
while(i < 6){
  printf("%d ", a[i]);
}
```

問 146

配列 a の全ての要素を表示したいが、無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
int i = 0, int a[6] = {0, 0, 0, 1, 2, 3};
while(a[i] != 4){
  printf("%d ", a[i]);
  i = i + 1;
}
```

問 147

a から二つおきに f までのアルファベットを表示したいが、無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
char c = 'a';
while(c != 'f'){
  printf("%c", c);
  c = c + 2;
}
```

問 148

無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
int i, j;
i = 0;
while(i < 3){
  printf("i = %d\n", i);
  j = 0;
  while(j < 2){
    printf("i = %d, j = %d\n", i, j);
    j = j + 1;
  }
  j = j + 1;
}
```

問 149

無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
float x = 0;
while(x != 10){
  printf("f(%.1f) = %f\n", x, 2*x*x+1);
  x = x + 0.3;
}
```

問 150

1 から 200 までの数字を表示したいが、無限ループになっている。それはなぜか。また、意図した通りに表示されるように訂正せよ。

```
int i = 1;
while(i = 200){
  printf("%d\n", i);
  i = i + 1;
}
```

16. 条件分岐：if

プログラムの一部を、ある条件によって実行するかしないのかを、if 文によって選択することができる。

if 文は次の様に書く。

```
if(条件となる式){
  選択的に実行する文の並び
}
```

「条件となる式」を条件式と呼ぶ。条件式は小括弧「( )」で括り、選択的に実行する文の並びは中括弧「{}」で括る。

条件式を評価し、その評価結果が真（0 以外、問 12 参照）であるとき、「選択的に実行する文の並び」を実行する。条件式の評価結果が偽（0）ならば（なにも実行せず）if 文を終了する。

例：

```
printf("x is ");
if(x < 10){
  printf("less ");
}
if(x > 10){
  printf("larger ");
}
printf("than 10.\n");
```

を実行すると、例えば x の値が 5 のときは

```
x is less than 10.
```

が表示され、x の値が 15 のときは

```
x is larger than 10.
```

が表示される。

このように選択的にプログラムを実行することを、条件分岐と呼ぶ。

問 151

何が表示されるか。

```
int x = 0;
if(x == 0){
  printf("x is 0\n");
}
if(x == 1){
```

```
  printf("x is 1\n");
}
if(x == 2){
  printf("x is 2\n");
}
```

問 152

何が表示されるか。

```
int x = 1;
if(x > 0){
  printf("x is larger than 0\n");
}
if(x != 1){
  printf("x is not 1\n");
}
if(x < 2){
  printf("x is smaller than 2\n");
}
```

問 153

何が表示されるか。

```
double x = 1.5;
if(0 < x && x < 2){
  printf("x is between 0 and 2\n");
}
if(x == 1){
  printf("x is 1\n");
}
if(x == 1.5 || x == 2.5){
  printf("x is 1.5 or 2.5\n");
}
```

問 154

配列 a の 40 より大きい要素だけを表示したいが、次の三つのプログラムがある。

```
int i = 0, a[6] = {56, 11, 34, 77, 39, 90};
while(i < 6){
  if(a[i] > 40){
    printf("%d ", a[i]);
  }
  i = i + 1;
}
```

```
int i = 0, a[6] = {56, 11, 34, 77, 39, 90};
while(a[i] > 40){
  printf("%d ", a[i]);
  i = i + 1;
}
```

```
int i = 0, a[6] = {56, 11, 34, 77, 39, 90};
while(i < 6){
  if(a[i] > 40){
    printf("%d ", a[i]);
    i = i + 1;
  }
}
```

それぞれ実行したときに、何が表示されるか。また上の三つのプログラムはどのように違うのか、それぞれの `while` ループにおける `if` の条件式の評価に注目して、なぜ表示結果が違うのかの理由を述べよ。

問 155

何が表示されるか。また、これは何をしているのかを述べよ。

```
int numbers[6] = {64, 37, 74, 15, 13, 62}, i = 0;
while(i < 5){
  if(numbers[i] > 40){
     printf("%d ", numbers[i]);
  }
  i = i + 1;
}
```

問 156

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0;
char c[16] = "abcdefghijklmno";
```

```
while(i < 15){
  printf("%c", c[i]);
  i = i + 1;
  if(i == 5 || i == 10){
    printf("\n");
  }
}
```

問 157

何が表示されるか。また、これは何をしているのかを述べよ。

```
int a[4] = {7, 1, 9, 5}, i = 0, t;
while(i < 4){
  t = a[i];
  if(t == 5){
    printf("test");
  }
  if(t == 1){
    printf("is ");
  }
  if(t == 7){
    printf("This ");
  }
  if(t == 9){
    printf("a ");
  }
  i = i + 1;
}
printf(".\n");
```

17. 条件分岐：if ～ else ～

プログラムを、ある条件によってどちらを実行するか、if 文と else 文によって決めることができる。

if 文と else 文は次の様に書く。

```
if(条件となる式){
  条件を満たすときに実行する文の並び
}else{
  条件を満たさないときに実行する文の並び
}
```

「条件となる式」を条件式と呼ぶ。条件式は小括弧「( )」で括り、if の後に書く。実行する文の並びは、それぞれ if と else の後に続けて中括弧「{ }」で括る。

条件式を評価し、その評価結果が真（0 以外、問 12 参照）であるとき、「条件を満たすときに実行する文の並び」を実行する。

条件式の評価結果が偽（0）ならば、「条件を満たさないときに実行する文の並び」を実行する。

例：

```
printf("x is ");
if(x < 10){
  printf("less ");
}else{
  printf("larger ");
}
printf("than 10.\n");
```

を実行すると、例えば x の値が 5 のときは

```
x is less than 10.
```

が表示され、x の値が 15 のときは

```
x is larger than 10.
```

が表示される。

前章の冒頭の例と比較せよ。

問 158

何が表示されるか。

```
int x = 0;
if(x == 0 || x == 1 || x == 2){
  printf("x is 0 or 1 or 2\n");
}else{
  printf("x is not 0/1/2\n");
}
```

問 159

何が表示されるか。

```
int x = 1;
if(x > 0){
  printf("x is larger than 0\n");
}else{
  printf("x is smaller than 0\n");
}
```

問 160

何が表示されるか（問 12 参照）。

```
int a = 2, b = 2;
if(a){
  printf("a is true.\n");
}else{
  printf("a is false.\n");
}
if(a - b){
  printf("a-b is true.\n");
}else{
   printf("a-b is false.\n");
}
```

問 161

何が表示されるか（問 12 参照）。

```
if(2){
  printf("2 is true.\n");
}else{
  printf("2 is false.\n");
}
if(-3){
  printf("-3 is true.\n");
}else{
  printf("-3 is false.\n");
}
if(0){
  printf("0 is true.\n");
```

```
}else{
  printf("0 is false.\n");
}
```

問 162

何が表示されるか。

```
int a = 3, b = 1;
if((a = b) < 2){
  printf("2より小\n");
}else{
  printf("2以上\n");
}
```

問 163

何が表示されるか。

```
int a = 3, b = 1;
if((a = b) == 3){
  printf("this is 3.\n");
}else{
  printf("this is not 3.\n");
}
```

問 164

何が表示されるか（問 19 参照）。

```
int a = 2, b = 2, c = 2;
if(a = b = c = 3){
  printf("true %d %d %d\n", a, b, c);
}else{
  printf("false %d %d %d\n", a, b, c);
}
```

問 165

何が表示されるか（問 19 参照）。

```
int a = 3, b = 5;
if(a = b){
  printf("a = %d, b = %d\n", a, b);
}else{
  printf("a = %d, b = %d\n", a, b);
}
```

問 166

何が表示されるか（問 20 参照）。

```
int a = 2, b = 2, c = 2;
if(a = b = c < 3){
  printf("true %d %d %d\n", a, b, c);
}else{
  printf("false %d %d %d\n", a, b, c);
}
```

問 167

何が表示されるか。

```
char c1 = 'd';
if('a' <= c1  && c1 <= 'z'){
  printf("%c is a small letter.\n", c1);
}else{
  printf("%c is not a small letter.\n", c1);
}
```

問 168

何が表示されるか。

```
printf("c-a  and  f-d  are ");
if( 'c' - 'a' == 'f' - 'd' ){
  printf("same.\n");
}else{
  printf("different.\n");
}
```

18. 条件分岐と繰り返しの組み合わせ

while ループの中で、if 文で処理を分岐することができる。このような繰り返しと条件分岐の組み合せは、さまざまな処理を行うことができ、非常に重要でよく使われる。

問 169

次のプログラムは何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0, count = 0;
char c[15] = "This Is A Pen.";
while(i < 14){
  if('A' <= c[i] && c[i] <= 'Z'){
    count = count + 1;
  }
  i = i + 1;
}
printf("count: %d\n", count);
```

問 170

文字列 fourWords 内の小文字の数を表示したい。下線部を埋めよ。

```
int i = 0, count = 0;
char fourWords[25] = "United States Of America";
while( __________ ){
  if( ____________________________________ ){
    ____________________
  }
  i = i + 1;
}
printf("count: %d\n", count);
```

問 171

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0;
char greeting[21] = "I'm fine, thank you.";
while(i < 20){
  if('a' <= greeting[i] && 'z' >= greeting[i]){
```

```
    printf("%c", greeting[i] - ('a' - 'A'));
  }else{
    printf("%c", greeting[i]);
  }
 i = i + 1;
}
printf("\n");
```

問 172

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0;
char cipher[13] = "Hnv aqd xnt?";
while(i < 12){
  if('b' <= cipher[i] && 'y' >= cipher[i]){
    printf("%c", cipher[i] + 1);
  }else{
    printf("%c", cipher[i]);
  }
 i = i + 1;
}
printf("\n");
```

問 173

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0;
char c[18] = "I'm 18 years old.";
while(i < 17){
  if('0' <= c[i] && '8' >= c[i]){
    printf("%c", c[i] + 1);
  }else{
    printf("%c", c[i]);
  }
  i = i + 1;
}
printf("\n");
```

問 174

文字列を全て小文字にして表示したい。下線部を埋めよ。

```
int i = 0;
char c[21] = "I’m Fine, Thank You.";
while( _______ ){
  if( ____________________________ ){
    printf("%c", c[i] ____________________);
  }else{
    printf("%c", c[i]);
  }
 i = i + 1;
}
printf("\n");
```

問 175

次の配列 f がある。

```
float f[6] = {1.23, 4.56, 78.9, 10.11, 30.23, 88.12};
```

この配列 f の 20 以上 80 未満の要素を表示するプログラムを書け。

問 176

if 文の中にさらに if 文を書いてもよい。
何が表示されるか。

```
int a = 0;
while(a < 150){
  a = a + 33;
  if(50 < a && a < 100){
    if(a < 75){
      printf("50 < %d < 75\n",  a);
    }else{
      printf("75 < %d < 100\n", a);
    }
  }else{
    if(a < 50){
```

```
      printf("%d < 50\n", a);
    }else{
      printf("100 < %d\n", a);
    }
  }
}
```

問 177

次のプログラムは、判別式が正のとき、二次方程式 $ax^2+bx+c=0$ の解 $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ を計算し表示するものである。double 型変数 a,b,c はすでに宣言されていて、何らかの値が代入されているとする。

問 34 等を参考に、以下のプログラムの下線部を埋めよ。

```
double x1, x2;
if( ____________________________ >= 0){
  x1 = ____________________________
  x2 = ____________________________
  printf("%f %f\n", x1, x2);
}else{
  printf("no solution.\n");
}
```

問 178

次のプログラムは、二次方程式 $ax^2+bx+c=0$ の解 $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ を計算し表示するものである。判別式 D が 0 のときは実数解（重根）を一つ、判別式 D が正のときは実数解を二つ、判別式 D が負のときは虚数解を二つ表示する。

double 型変数 a,b,c はすでに宣言されていて、何らかの値が代入されているとする。問 34 等を参考に、以下のプログラムの下線部を埋めよ。

```
double x1, x2, D;
D = ____________________________
if(D == 0){
  printf("%f\n", ____________________);
}else{
  if(D > 0){
    x1 = ____________________________
    x2 = ____________________________
    printf("%f\n%f\n", x1, x2);
  }else{
    x1 = ____________________________
    x2 = ____________________________
```

```
    printf("%f+%fi\n", x1, x2);
    printf("%f-%fi\n", x1, x2);
  }
}
```

問 179

次のプログラムは、$a+b+c=17$ を満たす 10 以下の自然数 a, b, c の組を求めるものである。プログラム中の下線部を埋めよ。

また、このプログラムはどのように解を求めているのか（アルゴリズム）を述べよ。さらに、このアルゴリズムは効率が悪い（実行した時に繰り返しの回数が多いということ）。それはなぜか、どうすれば良くなるのかを述べ、プログラムを修正せよ。

```
int a = 1, b, c;
while(a <= 10){
  b = 1;
    while(b <= 10){
      c = 1;
      while(c <= 10){
        if( ____________________ ){
          printf("%d+%d+%d=%d\n", a,b,c, ________);
      }
      c = c + 1;
    }
    b = b + 1;
  }
  a = a + 1;
}
```

問 180

次のプログラムは、$a+b+c=7$ を満たす 10 以下の自然数 a, b, c を一つだけ求めるものである。何が表示されるか。

また前問のプログラムとどう違うのか、変数 `flag` の役割は何かを述べよ。

```
int a = 1, b, c, flag = 1;
while(a <= 10 && flag){
  b = 1;
    while(b <= 10 && flag){
      c = 1;
      while(c <= 10 && flag){
        if(a+b+c == 7){
          printf("%d+%d+%d=%d\n", a,b,c,a+b+c);
```

```
            flag = 0;
        }
        c = c + 1;
      }
      b = b + 1;
    }
    a = a + 1;
}
```

19. 剰余：余りをもとめる

9/5 の評価結果は（整数同士の除算なので商）1 である。余りは 4 であるが、これを求めるには剰余演算子「%」を使う。

> 例：9 % 5 の評価結果は 4 である。

この演算子「%」は、他の+,-,*,/などの演算子と同様に式中に書くことができるが、演算子の優先度に注意すること。

> 例：18 % 4 / 2 は (18 % 4) / 2 と等価であるので、評価結果は 1 である。

問 181

何が表示されるか。

```
printf("%d\n", 4 % 2);
printf("%d\n", 1 % 2);
printf("%d\n", 9 % 2);
printf("%d %d\n", 9 / 5, 9 % 5);
```

問 182

何が表示されるか。また、これは何をしているのかを述べよ。

```
int n = 1;
while(n <= 100){
 if(n % 17 == 0){
   printf("%d ", n);
 }
 n = n + 1;
}
```

問 183

何が表示されるか。また、これは何をしているのかを述べよ。

```
int n = 1;
while(n <= 100){
 if(n % 3 == 0 && n % 7 == 0){
   printf("%d ", n);
 }
 n = n + 1;
}
```

問 184

ある値が入っている int 型変数 x の 1 の位を y に代入したい。下線部を埋めよ。

```
int y;
y = ____________________
```

問 185

ある値が入っている int 型変数 x の 10 の位を y に代入したい。下線部を埋めよ。

```
int y;
y = (x / 10) _________________
```

問 186

ある値が入っている int 型変数 x の 100 の位を y に代入したい。下線部を埋めよ。

```
int y;
y = ____________________________
```

問 187

何が表示されるか。

```
int i = 0;
char c[15] = "Today is fine.";
while(i < 24){
 printf("%c", c[i % 14]);
 i = i + 1;
}
```

問 188

何が表示されるか。

```
int i = 0;
char c[15] = "Today is fine.";
while(i < 14){
 printf("%c", c[(i+3) % 14]);
 i = i + 1;
}
```

問 189

100 以下の 5 と 7 の公倍数を求めたい。下線部を埋めよ。

```
int n = 1;
while(__________){
 if( ______________________ ){
   printf("%d ", n);
 }
 n = n + 1;
}
```

問 190

以下は 98 の約数を求めるプログラムである。下線部を埋めよ。

```
int n = 98;
while(__________){
 if( 98 % n == 0 ){
   printf("%d ", n);
 }
 n = _______________
}
```

問 191

以下は 45 と 630 の公約数を求める効率の悪いプログラムである。下線部を埋めよ。また効率のよい公約数を求めるアルゴリズムにはどのようなものがあるのか、調べよ。

```
int n = __________;
while(__________){
 if( ______________________ ){
   printf("%d ", n);
 }
 n = _______________
}
```

20. 算術代入演算子

C 言語では、代入と+,-,*,/の演算子を一度に行うことができる。
+=,-=,*=,/=演算子である（=とその前の記号は離してはいけない。二文字で一つの演算子である）。

例：i = i + 2; の代わりに、i += 2; と書くことができる。

問 192

それぞれの代入文を、算術代入演算子を用いて書き直せ。また、その逆を書け。

代入文	+=,-=,*=,/=を使う場合
i = i + 2;	i += 2;
x = x - 1;	
x = x + y;	
y = y * z * 2;	
	z /= x + 4;
	y += 2 * x + 3 * z;
	x *= z + 2;

問 193

何が表示されるか。

```
int i = 0;
while(i < 10){
  printf("%d ", i);
  i += 2;
}
```

問 194

何が表示されるか。

```
int n = 100;
while(n > 1){
  if(n % 17 == 0){
    printf("%d ", n);
  }
  n -= 1;
}
```

問 195

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 2, f = 1;
while(i < 7){
  printf("%d ", f);
  f *= i;
  i += 1;
}
```

問 196

頻繁に使用する `i += 1;` と `i -= 1;` だけは、C 言語には特別な書き方がある。
変数の直後に記号を二つならべて、`i++;` と `i--;` と書くことができる（問 13 参照）。

例：`i = i + 1;` は、`i += 1;` とも `i++;` とも書くことができる。

それぞれの代入文を、`++,--`演算子を用いて書き直せ。また、その逆を書け。

代入文	`++,--`を使う場合
`i = i + 1;`	
`x = x - 1;`	
`x += 1;`	
`y -= 1;`	
	`z++;`
	`z--;`

問 197

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0;
while(i < 10){
  printf("%d ", i);
  i++;
}
```

問 198

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 0, a[7] = {2,4,6,8,10,9,12};
while(a[i] < 10){
  printf("%d ", a[i]);
  i++;
}
```

問 199

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 6, a[6] = {2,4,6,8,10,12};
while(i > 0){
  i--;
  printf("%d ", a[i]);
}
```

問 200

配列 a の値を（*で）表示したい（問 140 参照）。
下線部を埋めよ。

```
int a[10] = {3, 4, 1, 6, 6, 2, 7, 2, 9, 9}, i = 0, k;
while( _____________ ){
  k = 0;
  printf("%d: ", i);
  while( _____________ ){
    printf("*");
    k++;
  }
  printf("\n");
  i++;
}
```

問 201

何が表示されるか。また、これは何をしているのかを述べよ。

```
int i = 6, j = 0, a[6] = {2,4,6,8,10,12};
while(i > 0){
  i--;
  printf("%d %d\n", a[i], a[j]);
  j++;
}
```

21. 繰り返し：forループ

プログラムの一部を、全く同じ書き方で繰り返して書ける場合には、その部分を for 文でまとめることができる。

for 文は次の様に書く。

```
for(式1; 式2; 式3){
  繰り返す文の並び
}
```

式 1、式 2、式 3 の間はセミコロン「;」で区切り、繰り返す文の並びは中括弧「{ }」で括る。

1 まず最初に**式 1 を評価**する。通常は、ループを制御する変数を初期化する代入文である（i = 0 など）。2 次に、**式 2 を評価**する（通常は i < 10 などの条件式である）。式 2 の評価結果が真（0 以外、問 12 参照）であるとき、「繰り返す文の並び」を実行する。3 そして**式 3 を評価**する。通常は変数の値を変化させる算術代入文である（i++ や i += 2 など）。2 そして**再び式 2 の評価**に戻る。

式 2 の評価結果が偽（0）ならば、for 文を終了する。

式 1 変数の初期化を行う代入文（i=0 など）、

式 2 繰り返しを行うかどうかを判定する条件式（i<3 など）、

式 3 繰り返しの最後に変数の値を増やす（減らす）ための代入文（i++など）

例：次の while による繰り返し

```
int i = 0;
while(i < 3){
  printf("%d\n", i);
  i = i + 1;
}
```

は、次の for による繰り返し

```
int i;
for(i = 0; i < 3; i++){
  printf("%d\n", i);
}
```

と等価である。

for によるプログラムの繰り返しを for ループと呼ぶ。for ループは while ループとまったく同じことができるが、for ループの方が簡潔に書ける場合が多い。

問 202

何が表示されるか。

```
int i;
for(i = 0; i < 10; i++){
  printf("%d ", i);
}
```

問 203

何が表示されるか。

```
int i;
for(i = 0; i < 66; i += 7){
  printf("%d ", i);
}
```

問 204

次のプログラムがある。

```
int i = 0, a[6] = {2,4,6,8,10,12};
while(a[i] < 10){
  printf("%d ", a[i]);
  i++;
}
```

これを for を使って書き直す。下線部を埋めよ。

```
int i, a[6] = {2,4,6,8,10,12};
for(i = 0 ; ___________ ; i++){
  printf("%d ", a[i]);
}
```

問 205

次のプログラムがある。

```
int i = 6, a[6] = {2,4,6,8,10,12};
while(i > 0){
  i--;
  printf("%d ", a[i]);
}
```

これを for を使って書き直したい。しかし、変数 i の値を減らすための代入文が繰り返しの最後にない。まず、次のように書き換える。

```
int i = 5, a[6] = {2,4,6,8,10,12};
while( __________ ){
  printf("%d ", a[i]);
  i--;
}
```

これを for を使って書き直す。

```
int i, a[6] = {2,4,6,8,10,12};
for(_____________________________ ){
  printf("%d ", a[i]);
}
```

下線部を埋めよ。

問 206

次のプログラムを実行すると何が表示されるか。

```
char c;
for(c = 'b'; c != 'f'; c++){
  printf("%c", c);
}
```

問 207

何が表示されるか。

```
int i;
for(i = 0; i < 5; i++){
  printf("%d ", i);
}
printf("%d ", i);
for(i = 5; i >= 0; i--){
  printf("%d ", i);
}
```

問 208

何が表示されるか。

```
int i, a[5] = {2, 4, 6, 8, 10};
for(i = 0; i < 5; i++)
  printf("%d ", a[i]);
}
```

問 209

次のプログラムがある。

```
int a[6] = {1, 2, 3, 4, 5, 6}, i;
for(i = 0; i < 6;  ＿＿＿＿＿＿＿＿  ){
  printf("%d ", a[i]);
}
```

これを実行したときに、次のように表示したい。

```
1 3 5
```

プログラム中の下線部を埋めよ。

問 210

次のプログラムがある。

```
int a[6] = {1, 2, 3, 4, 5, 6}, i;
for(i = 5; i >= 0;  ＿＿＿＿＿  ){
  printf("%d ", a[i]);
}
```

これを実行したときに、次のように表示したい。

```
6 4 2
```

プログラム中の下線部を埋めよ。

問 211

次のプログラムがある。

```
int i, a[7] = {1,3,5,7,9,11,13};
for(i = 0 ; ＿＿＿＿＿＿＿＿＿＿＿＿ ){
  printf("%d ", a[i]);
}
```

これを実行したときに、次のように表示したい。

```
1 3 5 7 9
```

プログラムの下線部を埋めよ。

問 212

次のプログラムがある。

```
int i;
for(_____________________________ ){
  printf("%d ", i);
}
```

これを実行すると、100 以下の全ての奇数を表示するようにしたい。プログラムの下線部を埋めよ。

問 213

次の配列 f がある。

```
float f[4] = {1.23, 4.56, 78.9, 10.11};
```

この配列 f の全ての要素を表示するプログラムを for 文を用いて書け（問 175 参照）。

問 214

何が表示されるか。また、何を計算して表示しているか（問 125 参照）。

```
int i, sum = 0;
for(i = 1; i <= 10; i++){
  sum = sum + i;
}
printf("%d\n", sum);
```

問 215

このプログラムは while を使って、3 の階乗を計算するものである。

```
int i = 1, f = 1;
while(i < 4){
  f *= i;
  i++;
}
printf("%d\n", f);
```

for を使ったプログラムに書き直せ。

問 216

このプログラムは、西暦 1000 年から 2003 年までの閏年を表示するプログラムである。閏年（leap year）は、4 で割り切れて 100 で割り切れない年か、400 で割り切れる年である。
下線部を埋めよ。

```
int year;
for(year = 1000; year <= 2003; year++){
  if( ____________________________________ ){
    printf("Year %d is a leap year.\n", year);
  }
}
```

問 217

何が表示されるか.

```
int i, n=0;
for( i=-2; i<=9; i++ ){
   n += 1;
}
printf("n is %d\n", n);
```

問 218

for 文を使って，整数型の配列 a[128] のすべての要素の和を計算し表示したい．下線部を埋めよ。

```
int i, sum = 0;
for( ____________________________ ){
   sum += a[i];
}
printf("sum is %d\n", sum);
```

問 219

次のプログラムがある。

```
int i, n = 0;
for( i = -2; i < 100; i++ ){
   n += i;
}
```

これが表す数式は次のようなものである。空欄を埋めよ。

$$n = \sum_{i=\square}^{\square} \quad \square$$

問 220

次のプログラムがある。

```
int i, n = 0;
for( i = ________ ; i < 200 ; i++ ){
  n += ____________________________;
}
```

これが表す数式は次のようなものである。下線部と空欄を埋めよ。

$$n = \sum_{i=8}^{\square} (5i+1)$$

問 221

次の数学の漸化式の第 10 項までを C 言語で計算したい。

$$\begin{aligned} a_1 &= 1 \\ a_n &= 2a_{n-1}+1, \quad n \geq 2 \end{aligned}$$

プログラム中の空欄を埋めよ。

```
int n, a = 1;
printf("%d\n", a);
for(n = 2; n <= 10; n++){
  ____________________________________
  printf("%d\n", a);
}
```

問 222

何が表示されるか。また、変数 x1, x2 の値の変化に注目して、何を計算しているのか、変数 t の役割は何かを述べよ。

```
int i, x1 = 1, x2 = 2, t;
for(i = 0; i < 6; i++){
  printf("%d %d\n", x1, x2);
  t = x1;
  x1 = x2;
  x2 = t;
}
```

問 223

何が表示されるか。

```
int i, x1 = 1, x2 = 2, x3 = 3;
for(i = 0; i < 6; i++){
  printf("%d %d %d\n", x1, x2, x3);
  x1 = x2;
  x2 = x3;
  x3 = i * 2;
}
```

問 224

次の数学の漸化式の第 10 項までを C 言語で計算したい。

$$\begin{aligned} a_1 &= a_2 = 1 \\ a_n &= a_{n-1} - 2a_{n-2} + 1, \quad n \geq 3 \end{aligned}$$

問 221, 問 222, 問 223 を参考にして、次のプログラム中の空欄を埋めよ。

```
int n, an1 = 1, an2 = 1, an;
for(n = 3; n <= 10; n++){
  an = an1 - 2 * an2 + 1;
  printf("%d\n", an);
  ____________________________________
  ____________________________________
}
```

問 225

次の数学の漸化式の第 10 項までを C 言語で計算したい。

$$\begin{aligned} & a_1 = 1, \quad b_1 = -2 \\ & a_n = -2a_{n-1} + 6b_{n-1}, \quad b_n = a_{n-1} - b_{n-1}, \quad n \geq 2 \end{aligned}$$

問 221～224 を参考にして、次のプログラム中の空欄を埋めよ。

```
int n, an, bn, an1 = 1, bn1 = -2;
for(n = 2; n <= 10; n++){
  an = -2 * an1 + 6 * bn1;
  bn = an1 - bn1;
  printf("%d %d\n", an, bn);
  _____________________________
  _____________________________
}
```

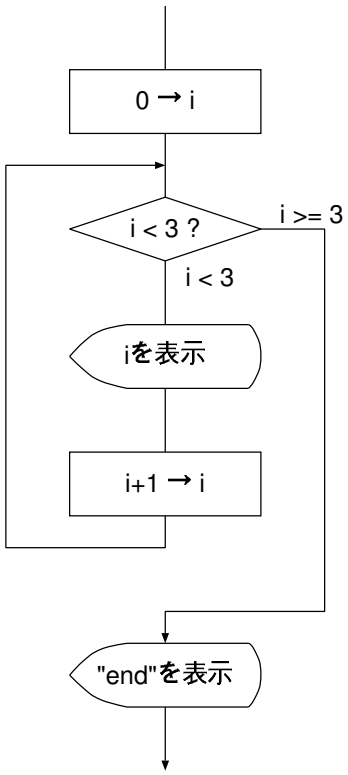
22. プログラムの図式化：フローチャート

プログラムがどのように実行されていくか（処理の流れ）を図式化する方法の一つに、フローチャート（flowchart、流れ図）がある。

例：次の while による繰り返し

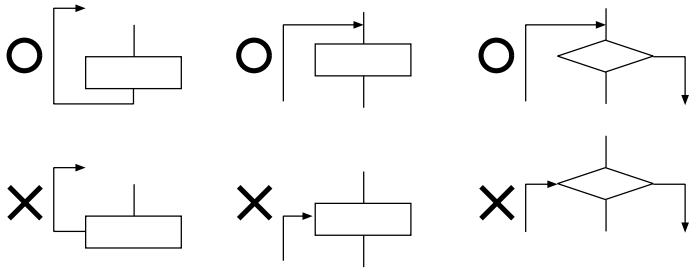
```
int i = 0;
while(i < 3){
 printf("%d\n", i);
 i++;
}
printf("end\n");
```

のフローチャートは次のように描く。



矩形（四角形）は一般の処理を、菱形は判断（判定）を、釣鐘型は表示を表す。処理のつながりを上から下へと線で結ぶ。判断の箇所では線は分岐し、繰り返しを表す場合には線を上へ戻す。

上に戻ったり分岐したりした線は、矩形や菱形には必ず上から入り、線には横から入る。矩形からでる線は下へ、菱形から出る線は左右下へと描く（下図参照）。



代入は「『代入する値（式）』→『代入先の変数』」と書く（C 言語の代入文の書き方とは異なる）。C 言語特有の演算子（++, --, +=, -=, *=, /=など）は使用しない。

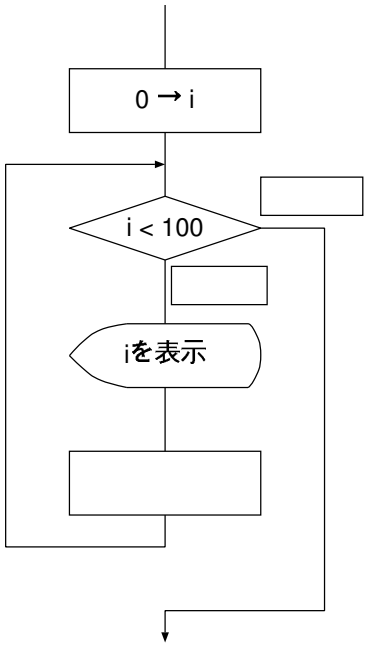
変数の宣言は描かない。

問 226

以下のプログラムは 7 の倍数を表示する。

```
int i = 0;
while(i < 100){
 printf("%d ", i);
 i += 7;
}
```

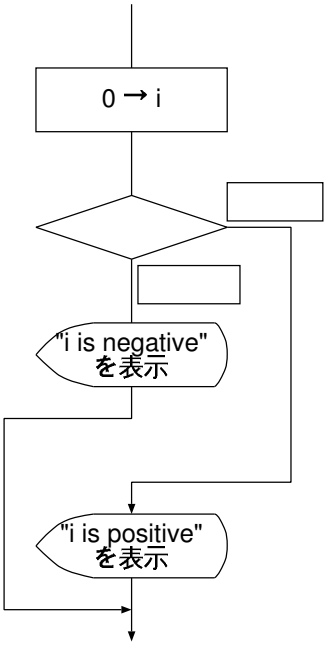
このフローチャートは次の様になる。空欄を埋めよ。



問 227

以下のプログラムに対応するフローチャートは次の様になる。空欄を埋めよ。

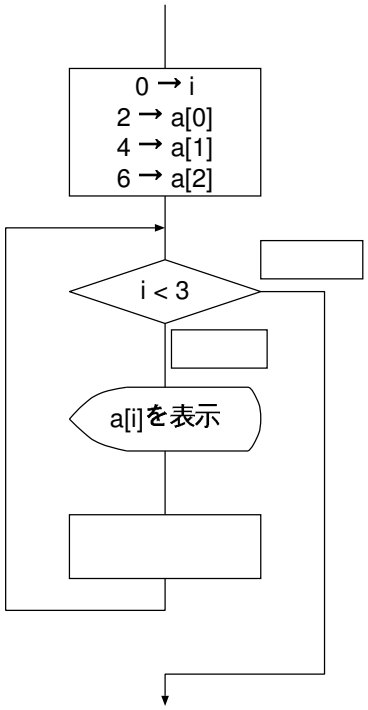
```
int i = 0;
if(i > 0){
  printf("i is positive.\n");
}else{
  printf("i is negative.\n");
}
```



問 228

次のプログラムに対応するフローチャートは次の様になる。空欄を埋めよ。

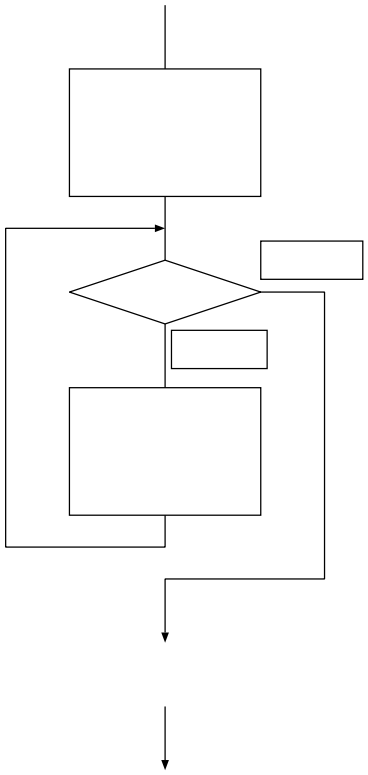
```
int i = 0, a[3] = {2,4,6};
while(i < 3){
  printf("%d ", a[i]);
  i++;
}
```



問 229

次のプログラムに対応する次のフローチャートを完成させよ。

```
int i = 0, sum = 0;
while(i < 10){
  i = i + 1;
  sum = sum + i;
}
printf("%d\n", sum);
```



問 230

次のプログラムに対応する次のフローチャートを完成させよ。

```
int i = 1, f = 1;
while(i < 4){
  f *= i;
  i++;
}
printf("%d\n", f);
```

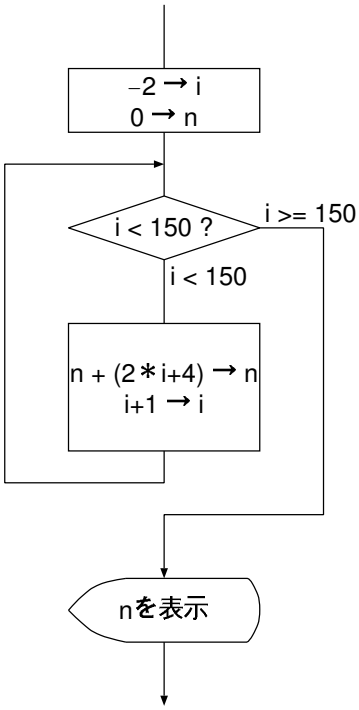
問 231

次のプログラムに対応するフローチャートを描け。

```
int i=-3, n=0;
while(i != 128){
   n++;
   i++;
}
printf("%d\n", n);
```

問 232

次のフローチャートに対応するC言語プログラムを、`while`文と`for`文を使ったものをそれぞれ書け。



問 233

繰り返しを使って、100 以下の 9 の倍数を全て表示するプログラムのフローチャートを描け。

問 234

次の配列 f がある。

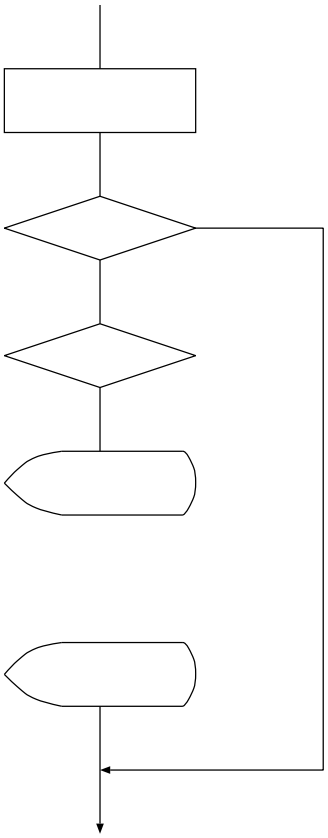
```
float f[4] = {1.23, 4.56, 78.9, 10.11};
```

この配列 f の全ての要素を表示するプログラムのフローチャートを描け。

問 235

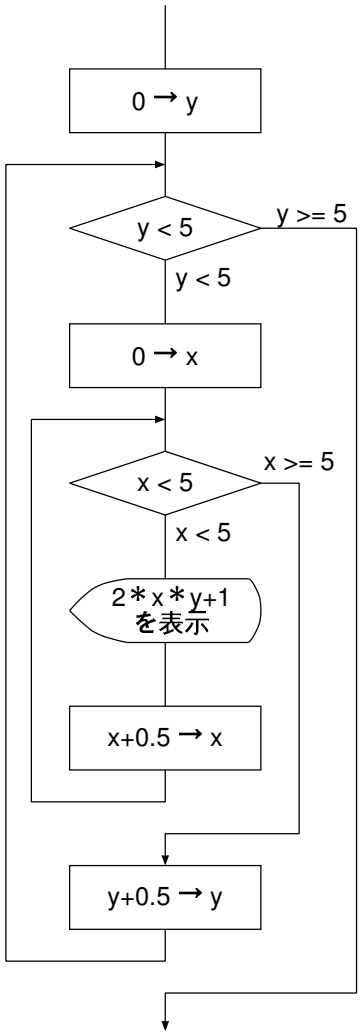
次のプログラムに対応するフローチャートを完成させよ。

```
int a = 76;
if(50 <= a && a < 100){
 if(a < 75){
   printf("50 < a < 75\n");
 }else{
   printf("75 < a < 100\n");
 }
}
```



問 236

次のフローチャートは、$2xy+1$ の値を表示するプログラムを二重ループによって表現している。これに対応する C 言語プログラムを、`while` と `for` を使ってそれぞれ書け。



問 237

次のプログラムに対応するフローチャートを描け。また、何が表示されるか。

```
int i;
char a = ’a’;
while(a < ’j’){
  i = 0;
  while(i < 5){
    printf("%c", a);
    a = a + 1;
    i = i + 1;
  }
  printf("\n");
}
```

23. ポインタ

C 言語では、他の変数そのものを値に持つ（その変数の値ではなく）型がある。これを**ポインタ**と呼ぶ。int 型のポインタ変数 p が、int 型の変数 x を値に持つとき、**ポインタ p は x を指している**と言う。

ポインタ変数を宣言するときは、変数名の前（型名の後）に*をつける。

例：**int 型のポインタ p を宣言するには**

```
int *p;
```

とする。二つ以上のポインタを同時に宣言するには、

```
int *p1, *p2, *p3;
```

のように変数名の前に*をつける。

int 型の変数を同時に宣言するには、

```
int x, *p1, y, *p2, *p3;
```

のように、ポインタ変数の前にだけ*をつける（この場合 x と y はポインタではなく通常の int 型の変数である）。

ポインタの型はそれが指す変数の型によって異なる。それはポインタの宣言時に決まり、異なる型の変数を指すことはできない。

例：int 型のポインタは float 型の変数を指すことはできない。

問 238

ある**ポインタ p が**別の**変数 z を指すためには**、その変数 z のアドレスを p へ代入する。これにはアドレス演算子 & を用いる（注意：AND 演算子とは何の関係もない）。

例：変数 z のアドレスをポインタ p へ代入するには、

```
p = &z;
```

とする。

ポインタに普通の数値を代入することはできない（してはならない）。必ず変数のアドレスを代入すること。

悪い例： `p = 6;` というような代入はできない。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。不定であれば不定と書け。ポインタの値は、それが指す変数名を書け。

代入文	x の値	y の値	p の値
`int x, *p, y;`			
`p = &x;`			
`x = 3;`			
`p = &y;`			
`y = 4;`			

問 239

同じ型同士のポインタは代入することができる。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	p の値	x の値	y の値	z の値
`int *x, *y;`				
`int p, z;`				
`p = 1;`				
`z = 2;`				
`x = &p;`				
`y = x;`				
`x = &z;`				

問 240

ポインタが別の変数を指しているとき、その指している変数の値を評価するには、間接（参照）演算子 * を用いる（注意：乗算の演算子とは何の関係もない）。

例：int 型ポインタ p が int 型変数 x を指している（つまり p=&x; とされている）とき、x と*p は同じものである。つまり、

```
x = 3;
y = *p;
*p = 4;
```

とすれば、y には（x と等価である）*p の評価結果 3 が代入され、（*p と等価である）x には 4 が代入されることになる。

ポインタ変数を宣言するときの*と、式中の*はまったく意味が異なることに注意。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x の値	y の値	p の値	*p の値
`int x,y,*p;`				
`x = 3;`				
`p = &x;`				
`x = 4;`				
`y = 5;`				
`p = &y;`				
`*p = 6;`				

問 241

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	x	y	p1	*p1	p2	*p2
int x,y,*p1,*p2;						
x = 3;						
y = 5;						
p2 = &x;						
p1 = p2;						
*p2 = 8;						
p1 = &y;						
y = 7;						
x = 1;						

問 242

ポインタが配列を指すためには、アドレス演算子 & は**必要ない**。

例：ポインタ p が配列変数 a[100] を指すには、単に
```
p = a;
```
とすればよい。

配列を指しているポインタに整数を足すと、配列の要素を指すことができる。さらに間接（参照）演算子 * を用いて、その要素の値を評価することができる。

例：ポインタ p が配列変数 a[100] を指しているとき、p+19 は a[19] を指す。したがって、a[19] と *(p+19) は等価であるので、
```
*(p+19) = 7;
```
は
```
a[19] = 7;
```
と同じことである。
また、p は p+0 と同じである（したがって *p は *(p+0) と等価）。

配列を指していないポインタに整数を足すことはできない（もし足すと不定になる）。

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	a[0]	a[1]	p1	*p1	p2	*p2
int a[2],*p1,*p2;						
p1 = a;						
p2 = p1 + 1;						
*p2 = 3;						
*p1 = 5;						
a[1] = a[0] + 3;						
a[0] = *p2 + 3;						
p2 = p1;						
a[0] = *(p1 + 1) - 8;						

問 243

配列または配列の要素を指しているポインタは、整数を足すと、その配列の別の要素を指すことができる。

> 例:ポインタ p が配列変数 a[100] の要素のうち a[1] を指しているとする。このとき、
> p = p + 1; または p++;
> とすると、p は a[2] を指すようになる。

また、配列の有効な要素（添字が 0 から要素数-1 の範囲、問 85 参照）を指しているかぎり、ポインタに足したり引いたりすることができる。

> 例：ポインタ p1 が配列変数 a[100] の要素のうち a[22] を指しているとする。このとき、
> p2 = p1 - 15;
> とすると、p2 は a[7] を指すようになり、
> p1 += 11;
> とすると、p1 は a[33] を指すようになる。

配列を指していないポインタに整数を足したり引いたりすることはできない（もし足すと不定になる）。また、足したり引いたりした結果、配列の有効な要素ではなくなった場合には、ポインタは不定になる。

何が表示されるか。

```
float f[4] = {1.23, 4.56, 78.9, 10.11}, *fp1, *fp2;
int i;
fp1 = f;
fp2 = fp1 + 1;
for(i = 0; i < 4; i+=2){
  printf("%.2f, %.2f\n", *fp2, *fp1);
  fp1++;
  fp2++;
}
```

問 244

配列名に整数を足すと、ポインタのように、配列の要素を指すことができる。

> 例:int a[100]; という配列がある場合、a + 1 は a[1] を指す。その値を評価する場合には、*(a + 1) とすればよい。
> また a は a+0 と等価なので、a[0] を指す。

また、ポインタと違い、配列名には値を代入できない。

例:int型の配列 a[100]の先頭を指すint型ポインタ *p; がある場合、p++; や p = a + 1; のように p に代入することは可能だが、a++; や a = a + 1; という a への代入はできない。

何が表示されるか。

```
float f[4] = {1.23, 4.56, 78.9, 10.11}, *fp;
int i;
fp = f;
for(i = 0; i < 4; i++){
  printf("%.2f %.2f\n", *(f + i), *fp);
  fp++;
}
```

問 245

ポインタ同士を ==, != で比較することができる。
何が表示されるか。

```
char a[10] = "123456789", *cp1, *cp2;
cp1 = a;
cp2 = a+6;
while(cp1 != cp2){
  printf("%c,",  *cp1);
  cp1++;
}
printf("\n");
cp1 = a+2;
while(cp1 != cp2){
  printf("%c,",  *cp2);
  cp2--;
}
```

問 246

ポインタを指すポインタを使うことができる。このためには、ポインタに対してアドレス演算子&や間接参照演算子*を適用すればよい。

例： int 型のポインタを指すポインタを宣言するには、
    int **p2;
とする（これは int *(*p2); という書き方の省略形である）。

例： int 型のポインタ p1 が int 型の変数 x を指す場合に
    p1 = &x;
としたのと同様に、p2 が p1 を指すに場合には
    p2 = &p1;
とする。

例： p1 が指す x の値を評価する場合に間接（参照）演算子を使って
    *p1
としたのと同様に、p2 が指す p1 が指す x の値を評価する場合には、
    **p2
とする。

例：上記の例で p2 = &p1; とした場合、
    *p2 は p1 と等価であり、**p2 は*p1 と等価である。

「ポインタを指すポインタ」を二重ポインタと呼ぶこともある。
以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	z	x	*x	w	*w	**w
int z,*x,**w;						
w = &x;						
x = &z;						
z = 3;						

問 247

以下の代入文が上から順番に実行されるとき、それぞれの代入文が実行された時点での各変数の値を書け。

代入文	z	x	*x	w	*w	**w
int z,*x,**w;						
x = &z;						
w = &x;						
*x = 7;						
**w = 2;						

問 248

ポインタの配列を作ることができる。
何が表示されるか。

```
char a[15] = "I’m a student.",
     b[11] = "Yes, I am.",
     c[13] = "No, I’m not.";
char *p[3];
int i;

p[0] = a;
p[1] = b;
p[2] = c;

for(i = 0; i < 3; i++){
  printf("%s\n", p[i]);
}
```

問 249

ポインタの配列へのポインタは、二重ポインタになる。
何が表示されるか。また、p1, p2 はどんな役割をしているのかを述べよ。

```
char a[15] = "I’m a student.",
     b[11] = "Yes, I am.",
     c[13] = "No, I’m not.";
char *p[3];
char **p1, **p2;

p[0] = a;
p[1] = b;
p[2] = c;
p1 = p;
p2 = p + 2;

for(p1 = p; p1 != p2; p1++){
  printf("%s\n", *p1);
}
```

問 250

前問において、*p1 は文字列（文字型配列）を指すポインタである。したがって、*p1+i は p1 が指す配列のある要素を指し、*(*p1+i) はその要素の値を評価することになる。
何が表示されるか。

```
char a[15] = "I’m a student.",
     b[11] = "Yes, I am.",
```

```
     c[13] = "No, I’m not.", d;
char *p[3];
char **p1, **p2;
int i;

p[0] = a;
p[1] = b;
p[2] = c;
p[3] = &d;
p1 = p;
p2 = p + 3;

for(p1 = p; p1 != p2; p1++){
  for(i = 0; i < 5; i++){
    printf("%c", *(*p1 + i));
  }
}
```

解答

問 1 :
3, 1, 6, 2, 1, 2.5, 72.6, -6, 11.1, 15.3, 1000, -12.5
問 2 :
1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0
問 3 :
-1.5, 17, 6, 5, 1, 25, 1, 1, 0, 1, -1.5, 17, 6, 5, 1, 1, 0, 1, 1, 0
問 4 :
10, 0, 32, 15, 1, 0, 1, 1, 0, -5, 6, 8, 15, 1, 0, 1, 1, 0
問 5 :
-2, -2, -8, 1, -10, 12, 10, 0, 1, 2, 1, 1, 3, 1, 1, 0, 1, 0
問 6 :
$\frac{10}{3}+1$, $\frac{2\times5}{2}\times5$, $\frac{\frac{9}{3}}{3}-1$, $\frac{9}{\frac{3}{3}}-1$, $1+\frac{3\times4}{2}-10$, $(1+3)\times\frac{4}{2}-10$, $\frac{1+3}{4\times2}-10$, `1 / 2 * 4`, `1 - 2 / 2 * 4 + 3`, `(1-2) / 2 * (4+3)`, `1 + (1-3)/(2/4 + 1) - 2`, `1 + (1 + (1 + (2 - 3)))`, `-(1/4 - 4/2)/(1 - 3/2) + 6/2`
問 7 :
○○○×○×○×××○○○○○○○×
問 8 :
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 4, 5
問 9 :
1, 0, 1, 0, 1, 0, 0, 0
問 10 :
`!(a<=2||a>=4)`
問 11 :
`3>4 || -2<6`, $5>4>3$, `3>=2 && 2>=1`
問 12 :
1, 1, 1, 1, 1, 0, 4, 0, 0, 1, 1
問 13 :
-2, 2, 2, -2, 2, -2, -3, 2, 2, 1, 5, -3, -2, -2, -4
問 14 :
<>という演算子はない。>と=の間が空いている。=>,!==という演算子はない。=と=の間が空いている。=<という演算子はない。/の間違い。--が連続している。
問 15 :
`x = -3 , x = 1 - 2 , x = 1 + 2 + 3 + 4 * 5 , w = -2 * (4 - 2) , x = 0 < 1 , y = 3 > (1 && 5) , x = 1 - 2 == 4`
問 16 :
17, 1, 1, 15, 12, 83
問 17 :
3, 2, 5, 11, 1, 0, 1, 1, 5, 7, 12, 18, 0, 1, 0
問 18 :
`x: 3 2 2 1 1 1 1 3 3 5 5 8 8 13 13 7 7`
`y: 4 4 6 6 0 0 2 2 4 4 6 6 10 10 17 17 8`
問 19 :
3, 7, 5, 5, 5, 3, 6, 7
問 20 :

x	y	z
1	不定	不定
1	2	不定
1	2	3
8	2	3
7	7	7
8	8	8
14	14	8
14	14	1
14	14	14
0	14	0
5	14	0
5	5	0
3	5	0

問 21 :
`3 + 1` に代入することはできない。`(x = y)` に代入することはできない。4 に代入することはできない。
問 22 :

代入文	x の値	y の値	z の値
`y = 2;`	不定	2	不定
`x = 3;`	3	2	不定
`y = x;`	3	3	不定
`x = y + 2 * z;`	不定	3	不定
`y = x + 6;`	不定	不定	不定

問 23 :

代入文	x の値	y の値	z の値
`z = -4;`	不定	不定	-4
`x = z * 2 + y;`	不定	不定	-4
`z = x / 6;`	不定	不定	不定
`x = 2 ;`	2	不定	不定
`z = y + x * 6;`	2	不定	不定

問 24 :

代入文	x の値	y の値	z の値
`y = 2;`	不定	2	不定
`x = 3;`	3	2	不定
`x = y;`	2	2	不定
`z = x + 2;`	2	2	4
`y = x;`	2	2	4

代入文	x の値	y の値	z の値
`y = 2;`	不定	2	不定
`x = 3;`	3	2	不定
`y = x;`	3	3	不定
`z = x + 2;`	3	3	5
`x = y;`	3	3	5

代入文 `y=x` と `x=y` の順序が異なる。

問 25 :

代入文	x の値	y の値	z の値
y = 2;	不定	2	不定
x = 3;	3	2	不定
z = x;	3	2	3
x = y;	2	2	3
y = z;	2	3	3

入れ換える前の
x の値：3,　y の値：2
入れ換えた後の
x の値：2,　y の値：3

問 26 :

代入文	x の値	y の値	z の値
z = 1;	不定	不定	1
y = 3;	不定	3	1
x = z;	1	3	1
z = y;	1	3	3
y = x;	1	1	3

問 27 :

代入文	x の値	y の値	z の値
w = x;	2	15	87
x = y;	15	15	87
y = z;	15	87	87
z = w;	15	87	2

入れ換える前の
x の値：2,　y の値：15,　z の値：87
入れ換えた後の
x の値：15,　y の値：87,　z の値：2

問 28 :

x:　不定 3 5 5 5 5 5 5 1 3 5 7 9
2 4 6 8 32 32 32 32 32 32 32 32
32 16 8 4 2
y:　2 2 2 2 3 4 5 6 6 6 6 6 6
6 6 6 6 6 1 2 6 24 1000 100 10 1
1 1 1 1

問 29 :

$\frac{25y}{z}$, $\frac{x^2y}{3} + y$, $\frac{9}{xy} - 1$, $\frac{9}{z^2} - 1$, $x + \frac{yz}{w} - u$, $\frac{(1+v)u}{2} - 10$, $7abxyz$, `2*x + 1, a*x*x + b*x + c, v * t * t / 2, h - g * t * t / 2, 1 + (2*x - m) / ((3*a)/2 + b) - 2*c, 1 + (x + (y + (2 - z))), -(4*x - 2*u) / (1 - 3*w) + 6/2`

問 30 :

$x + \log_{10} 20$, $\frac{2e^{3x}}{2a}$, $2a|x| + b$, $\sqrt{b^2 - 4ac}$, $\sqrt{x^2 + y^2}$, $2\sqrt[3]{2c}$, $\frac{\sin^2 x}{2} - 4$, $\frac{y}{2}\cos\frac{x}{2}$, $(x + 2y)^3$, $\lfloor z \rfloor + \lceil y \rceil$,

```
2 * fabs(x) + sin(x), 1 + pow(y + 3 * x, 15) + 4 * y, 4 * sqrt(k/m - 1), asin(x) - 1, log(x*x*x), 1 - exp(-t/(2*r)), (tan(a)+tan(b)) / (1-tan(a)), atan(2*a / b)
```

問 31 :

（例）`2 * g * x_0 * cos(theta / 2), 4 * Omega * Omega / 3 + k_0 * q * Q / Lambda, mu * (v_20 - v21), sqrt(G * m * fabs(M) / (R * l_1 * eps_0))`

問 32 :

（例）`-b + sqrt(b*b - 4*a*c)` と `-b - sqrt(b*b - 4*a*c), y - x_dash * sin(x), v_ddash + 2 * u_dot - 3 * z_hat, V_bar * V_bar * G * m / 2 + fabs(V_bar)`

問 33 :

```
x = (y*y - 8) / 4;
x = 2 * b - 4 * sin(theta/2);
x1 = -b + sqrt(b*b - 2*c); x2 = -b - sqrt(b*b - 2*c);
```

問 34 :

```
x1:  3 3
x2:  不定 -2
```

問 35 :

```
D: 5 5 5
x1:  不定 3 3
x2:  不定 不定 -2
```

問 36 :

```
b:  不定 -1 -1 -1 -1
c:  -6 -6 -6 -6 -6
D: 不定 不定 5 5 5
x1:  不定 不定 不定 3 3
x2:  不定 不定 不定 不定 -2
```

問 37 :

```
b:  -1 -1 -1 -1 -1 -1 -1 -1
c:  不定 -6 -6 -6 -6 -6 -6 -6
D: 不定 不定 25 5 5 5 5 5
x1:  不定 不定 不定 不定 6 3 3 3
x2:  不定 不定 不定 不定 不定 不定 -4 -2
```

問 38 :

```
x:  不定 3 3 3
y:  2 2 2 2
z:  不定 不定 3 -1
```

問 39 :

```
x:  不定 3.2 3.2 3.2
y:  2.0 2.0 2.0 2.0
z:  不定 不定 3.2 -1.9
```

問 40 :

```
x:  不定 1 0 0 1 1 0 0 0 0
y:  2 2 2 2 2 2 2 2 2 2
z:  不定 不定 不定 1.9 1.9 0.9 0.9 0.9 -0.9 0.0
```

問 41 :

以下の数字で、小数点の付いているものは実数型、付いていないものは整数型である。
-2, 2, -1.5, -1.0, 2, 2.5, 2.5, 2.5, 3.0, 1.0, 0.0, 1.0, 0, 4, 0

問 42 :
x: 不定 不定 12 12 12 12 12 12 12 12 12 12
y: 0 1 1 1 1 1 1 1 38 7 0 7
z: 不定 不定 不定13.0 13.0 13.1 13.2 13.3 13.3 13.3 13.3 13.3

問 43 :
x: 不定 不定 不定 1 0 0 3 3 5
y: 未定義 不定 2.0 2.0 2.0 2.0 2.0 2.0 2.0
z: 未定義 不定 不定 不定 不定 1.8 1.8 0.9 0.9

問 44 :
x: 未定義 未定義 不定 不定 1 0 0 2 2 4
y: 未定義 不定 不定 2.0 2.0 2.0 2.0 2.0 2.0 2.0
z: 不定 不定 不定 不定 不定 不定 1 1 0 0

問 45 :
int c; の変数宣言位置が代入文の後になっている。

問 46 :
int, int, double, float, double, double

問 47 :
double, double, double, double, double

問 48 :
double, double, float, double, float, int, double, float, float, float, float, double, float, double, int

問 49 :
例：(x - 2) * a, x - 2 * a, (float)(x - 2 * a), x - 2 * a, (float)(x - 2 * a), 5 / 2, 5 / 2. , 5 / (float)exp(x), 5 / (float)exp(x), (float)(1 / 2.), (float)(1 / 2), 1. / 2. , x / (float)y, x / (double)y, (int)floor(z) + (int)ceil(a)

問 50 :
x: 不定 不定 不定 2 2 2 2 2 2 2 2 4
y: 未定義 不定 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0
z: 未定義 不定 不定 不定 2.5 2.0 2.0 2.1 2.2 2.3 2.3 2.3

問 51 :
float: ○○○×○××○××
double:○○○○○○○○×○

問 52 :
float: ○○○×○××○××
double:○○○○○○○○×○

問 53 :
x: 不定 不定 不定 不定 2.23e20 2.23e20 不定 不定
y: 未定義 不定 1.23e20 1.23e20 1.23e20 4.46e20 4.46e20 不定
z: 未定義 不定 不定 1.0e20 1.0e20 1.0e20 1.0e20 1.0e20

問 54 :
x: 不定 不定 1.2e20 1.2e20 1.2e20 1.2e20 1.2e20 1.2e20
y: 不定 不定 不定 2.0e20 2.0e20 2.0e20 2.0e20 2.0e20
z: 未定義 不定 不定 不定 2.4e40 1.2e200 不定 不定

問 55 :
x: 不定 不定 不定 不定 不定 不定 -1.23e+10 -1.23e+10 -1.23e+10
y: 未定義 不定 不定 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9
z: 未定義 未定義 不定 不定 1000000000 不定 不定 不定 不定

問 56 :
x: 不定 不定 不定 不定 不定 不定 10 10 10
y: 未定義 不定 不定 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9 1.0000000002e9
z: 未定義 未定義 不定 不定 3000000000 不定 不定 不定 不定

問 57 :
x: 不定 不定 不定 100 100 不定 100 100 不定 100 100
y: 未定義 不定 不定 不定 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
z: 未定義 未定義 不定 不定 不定 不定 不定 1000000 1000000 1000000 不定

問 58 :
signed short int, unsigned short int, signed int, unsigned int, signed long int, unsigned long int, short, long, unsigned short

問 59 :

（例）signed int, unsigned short, float, float, float, signed short, float, float, float, unsigned int, float, double, unsigned int, unsigned int

問 60：
○○×（二文字）×（B が全角文字）○×（クォートではない）×（クォートではない）○○×（クォートが全角文字）×（全角文字）

問 61：
98, 102, 48, 57, 65, 71

問 62：
x: 不定 97 98 97 98 1 50

問 63：
x: 不定 不定 97 97 97 97 98 不定 不定 8
y: 未定義 不定 不定 97 97 97 97 97 97 97
z: 未定義 不定 不定 不定 不定 1 1 1 1 1

問 64：
x の値: 不定 不定 97 98 99 100 100 4 65 68 4 97 99 99 65 65 67 99
x の値が表す文字: -- -- a b c d d -- A D -- a c c A A C c

問 65：
1, 0, 0, 1, 1, 0

問 66：
0, 1, 0, 0, 1, 0

問 67：
This isa test.
Next, you say
hello world

問 68：
```
printf("I am a student.\n ");
printf("This is a pen.\n ");
```

問 69：
x の値: 不定 97 97 97 97 99 99 99 99 99 48 48 48 48 48 48 49 49 49
表示される内容：-- -- a b b -- 98 101 e d -- 0 48 0 1 0 -- 1 1

問 70：
printf("%d, a"); ではなく
printf("%d", a);
printf("d"); ではなく
printf("%d", d); か printf("%c", d);

問 71：
abcd
12387456
a is 97.
a is a.
0 is 48.
0 is 48.
0 is 0.

問 72：
```
printf("%c\n", a);
```

問 73：
c3 is i

問 74：
abc
a is 97 and a.

問 75：
a 0

問 76：
12.340000 cm
12.3 cm
12 cm
-3
3
3
不定
不定
3.000000
不定
3
1 2.000000 3
不定

問 77：
year : weight, height
1999 : 52.2kg, 172.5cm
2003 : 64.2kg, 173.5cm
grows : 12.0kg, 1.0cm total
grows : 3.00kg, 0.25cm per year

問 78：
5 5

問 79：
2 3 4
2 4 4
2 4 2

問 80：
"%d %d %c\n"

問 81：
「'」ではなく「"」が正しい。つまり
```
printf("a+b= %d\n", a+b);
```

問 82：
prinf ではなく printf（これはプログラミング中に非常によくある間違い）

問 83：
4
4
代入文の評価を忘れていたら問 19 へ。ただし、実際にこのようなまぎわらしい書き方をすることは好ましくない。

問 84 :
```
a[0]:  不定 3 3 3 3
a[1]:  不定 不定 -5 -5 10 10
a[2]:  不定 不定 不定 10 10 10
```
問 85 :
```
1 2
----
2.500000 4.000000
----
```
問 86 :

要素数が 4 の整数型の配列を定義すればよい。（例）`unsigned int studentNumber[4];`

24 時間分の気温（実数）を格納すればよいので、要素数が 24 の実数型の配列を定義すればよい。（例）`float temprature[24];`

問 87 :

i の値も一緒に考えるとよい。

a[0] の値	a[1] の値	a[2] の値	i の値
不定	不定	不定	不定
不定	不定	不定	0
3	不定	不定	0
3	不定	不定	1
3	-5	不定	1
3	-5	不定	2
3	-5	10	2
3	-5	10	0
3	3	10	0
3	3	10	2
3	3	3	2
3	3	不定	2

問 88 :
```
1 2 4
```
問 89 :
```
b is abc.
c a
b is bcd.
```
問 90 :

この問のプログラムは、文字列を数値の配列に変換している。
```
0 8 2
```
問 91 :
```
a b c
b c a
a
```
問 92 :

a[0]	a[1]	a[2]
3	-5	10
5	-5	10
5	10	10
5	10	8

問 93 :
```
float value[2] = {1.2, 4.5};
```
問 94 :
```
2.200000
```
問 95 :
```
5
```
問 96 :
```
5 4 8
4 8 5
5 4 8
```
問 97 :
```
2 3 4
3 3 4
3 4 4
3 4 3
```
問 98 :

a[0]	a[1]	b[0]	b[1]
4	2	未定義	未定義
4	2	1	0
1	2	1	0
1	2	2	0
1	2	2	2
1	2	2	1

問 99 :
```
a b c
a b c
```
問 100 :
```
a b c
97 98 99 0
a b c
abc
```
問 101 :
```
a[0] = a
a[1] = b
a[2] = c
a[0] = a
a[1] = b
a[2] = c
```
問 102 :
```
1237
bcdf
```
問 103 :
```
12
3b
cd
```
問 104 :
```
4, 7
```
問 105 :
```
8, 0
```
問 106 :
```
cab
```
問 107 :

1 を足す計算, 5 回
```
int i = 0;
while(i < 4){
  i = i + 1;
}
```

問 108 :
まず、a と b の値は 0 である。
一回目の繰り返し：条件式は真（0 < 5）である。a は 5 になり、b は 3 になり、さらに a は 3 になる。一回目の繰り返しが終わった時点で、a は 3 である。
二回目の繰り返し：条件式は真（3 < 5）である。a は 8 になり、b は 6 になり、さらに a は 6 になる。
三回目の繰り返し：条件式は偽（6 < 5）である。したがって繰り返しは終了する。
つまり、繰り返しが終了した時点で、a の値は 6、b の値は 6 である。
問 109 :
```
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```
問 110 :
```
0 7 14 21 28 35 42 49 56 63
```
66 未満の 7 の倍数（0 を含む）を計算している。
問 111 :
```
int i = 0, n = 0;
n = n + i;
i = i + 1;
printf("%d %d\n", n, i);
n = n + i;
i = i + 1;
printf("%d %d\n", n, i);
n = n + i;
i = i + 1;
printf("%d %d\n", n, i);
```
問 112 :
```
int i = 0, a[3];
while(i < 3){
  a[i] = i * 2;
  printf("%d %d\n", i, a[i]);
  i = i + 1;
}
```
問 113 :
```
bcde
```
問 114 :
```
0 1 2 3 4 5 5 4 3 2 1 0
```
問 115 :
```
2 4 6 8 10 9 7 5 3 1
```
問 116 :
```
a[0] = a
a[1] = b
a[2] = c
a[0] = a
a[1] = b
a[2] = c
```
問 117 :
```
2 4 6 8
```
10 未満の要素を順に表示し、最初に 10 以上の要素が見付かった時点で繰り返しを終了する。
問 118 :
```
good morning.
```
文字列を逆順に、つまり文字型配列の最後の要素から順に表示している。また、最後の ! は表示されない。
問 119 :
```
i = i + 1;
```
問 120 :
```
i = i + 2;
```
問 121 :
```
i < 5
```
問 122 :
```
int i;
char c[5] = "abcd";
i = 0;
while(i < 4){
  printf("%c", c[i]);
  i = i + 1;
}
```
問 123 :
```
int i;
float f[4] = {1.23, 4.56, 78.9, 10.11};
i = 0;
while(i < 4){
  printf("%f\n", f[i]);
  i = i + 1;
}
```
問 124 :
```
i = 0;
while(i < 5){
  printf("%d\n", b[i]);
  i = i + 1;
}
```
表示される結果
```
8
9
10
11
12
```
問 125 :
55, （略）0（または 1）から 10 までの和を計算している
問 126 :
6, （略）3 の階乗を計算している
問 127 :
問 110 を参照。
```
int i = 7;
while(i <= 100){
printf("%d ", i);
i = i + 7;
```

```
}
```
0 は 7 の倍数としない。

問 128 :

$$\frac{2n-3}{n^2+1}$$

問 129 :

```
sum + (1.0 / (n * n) + 2 * n);
```

問 130 :

```
-3
n + (2 * i + 4)
```
149

問 131 :

$$n = \sum_{i=-3}^{127} 1$$

問 132 :

一番目のプログラムの表示：000

二番目のプログラムの表示：なにも表示されない

三番目のプログラムの表示：123

（略）二番目の条件式は代入文になっているため（これは実際にとてもよくある間違い）。詳しくは問 19 を参照。

問 133 :

```
11 56 34 77 39
```

問 134 :

```
a e c b a e c b a e
```

問 135 :

```
13
```

問 136 :

```
mean + a[i];
mean / 10;
```

問 137 :

```
mean + a[i];
mean / 10;
variance / 10 - mean * mean;
```

なぜこのプログラムが平均と分散を計算できているのかをよく考えること。

問 138 :

```
i = 0
i = 0, j = 0
i = 0, j = 1
i = 1
i = 1, j = 0
i = 1, j = 1
i = 2
i = 2, j = 0
i = 2, j = 1
```

問 139 :

```
abcde
fghij
```

問 140 :

```
i < 10
k < a[i]
```

問 141 :

```
f(0.0) = 1.0
f(0.5) = 1.5
f(1.0) = 3.0
f(1.5) = 5.5
f(2.0) = 9.0
f(2.5) = 13.5
f(3.0) = 19.0
f(3.5) = 25.5
f(4.0) = 33.0
f(4.5) = 41.5
```

問 142 :

一つ目のプログラムの実行結果：

```
f(0.0,0.0) = 1.0
f(1.0,0.0) = 3.0
f(2.0,0.0) = 5.0
f(0.0,1.0) = 2.0
f(1.0,1.0) = 4.0
f(2.0,1.0) = 6.0
f(0.0,2.0) = 3.0
f(1.0,2.0) = 5.0
f(2.0,2.0) = 7.0
```

二つ目のプログラムの実行結果：

```
f(0.0,0.0) = 1.0
f(0.0,1.0) = 2.0
f(0.0,2.0) = 3.0
```

（略）内側のループの範囲が異なる。

問 143 :

```
int minute = 0, second;
while(minute < 60){
  second = 0;
  while(second < 60){
    printf("%d:%d\n",
           minute, second);
    second = second + 1;
  }
  minute = minute + 1;
}
```

問 144 :

```
int hour = 0, minute, second;
while(hour < 24){
  minute = 0;
  while(minute < 60){
    second = 0;
    while(second < 60){
      printf("%d:%d:%d\n",
             hour, minute,
second);
      second = second + 1;
    }
    minute = minute + 1;
  }
  hour = hour + 1;
}
```

問 145 :

i の値が変化していないため。printf の次の行に i = i + 1; を追加する。

問 146 :

間違った条件式 a[i] != 4 が常に真となってしまうため。

条件式は、a[i] の値で判別するのではなく、i < 6 する。

問 147 :

条件式 c != 'f' は変数 c の値が'f' にならなければ、ループが終了しないが、変数 c は'a' から二つおきに変化しているため、絶対に'f' になることがない。

条件式には != (非等号) は使わずに、不等号を使って c < 'f' とする。 問 149 参照。

問 148 :

外側のループの条件式は i < 3 なのに、i の値が変化していない。つまり、最後から二行目の「j = j + 1;」は間違いで、「i = i + 1;」が正しい。

これは実際のプログラミングで非常によくある間違いである。

問 149 :

条件式は x != 10 であるが、x の値は 0 から 0.3 刻みで変化しているため、絶対に 10 になることはない。

条件式には != (非等号) は使わずに、不等号を使って x < 10 とする。問 147 参照。

これは実際のプログラミングで非常によくある間違いである。

問 150 :

条件式 i = 200 は代入文であり、その評価結果は代入された値 (この場合は 200) である (問 19 参照)。したがって必ず真となり (問 12 参照)、ループは終了しない。

条件式には = (代入) と間違えやすい == (等号) は使わずに、不等号を使って i <= 200 とする。

これは実際のプログラミングで非常によくある間違いである。

問 151 :

```
x is 0
```

問 152 :

```
x is larger than 0
x is smaller than 2
```

問 153 :

```
x is between 0 and 2
x is 1.5 or 2.5
```

問 154 :

1 番目のプログラムの表示結果：56 77 90

2 番目のプログラムの表示結果：56

3 番目のプログラムの表示結果：無限ループ

(略)

問 155 :

```
64 74
```

(略) 40 より大きい要素を全て表示している。

問 156 :

```
abcde
fghij
klmno
```

(略) 5 文字目と 10 文字目の後に改行を表示している。

問 157 :

```
This is a test.
```

(略)

問 158 :

```
x is 0 or 1 or 2
```

問 159 :

```
x is larger than 0
```

問 160 :

```
a is true.
a-b is false.
```

問 161 :

```
2 is true.
-3 is true.
0 is false.
```

問 162 :

2 より小

問 163 :

```
this is not 3.
```

問 164 :

```
true 3 3 3
```

問 165 :

```
a = 5, b = 5
```

問 166 :

```
true 1 1 2
```

問 167 :

```
d is a small letter.
```

問 168 :

```
c-a and f-d are same.
```

問 169 :

```
count:  4
```

(略) 文字列 c の中の大文字の数を数えて表示している。

問 170 :

```
i < 24
'a' <= fourWords[i]
     && fourWords[i] <= 'z'
count = count + 1;
```

前問を参照。

問 171 :

```
I'M FINE, THANK YOU.
```

(略) 全ての小文字を大文字に変換して表示している。問 64 参照。

問 172 :

```
How are you?
```
（略）暗号化された文字列 cipher を復号化している。つまり、アルファベット小文字の b から y を表示するときには、一つずらして次の文字を表示している。

問 173：
```
I’m 29 years old.
```
（略）0 から 8 の数字を表示するときは、一つずらして次の数字を表示している。

問 174：
```
i < 20
’A’ <= c[i] && c[i] <= ’Z’
+ (’a’ - ’A’)
```
問 169、問 171 参照。

問 175：
```
float f[6] = {1.23, 4.56, 78.9,
10.11, 30.23, 88.12};
int i = 0;
while(i < 6){
   if(20 <= f[i]
        && f[i] < 80){
     printf("%.2f ", f[i]);
   }
   i = i + 1;
}
```
問 176：
```
33 < 50
50 < 66 < 75
75 < 99 < 100
100 < 132
100 < 165
```
問 177：
```
b*b - 4*a*c >= 0
x1 =
(-b+sqrt(b*b-4*a*c))/(2*a);
x2 =
(-b-sqrt(b*b-4*a*c))/(2*a);
```
問 178：
```
D = b*b - 4*a*c
-b / (2*a)
x1 =
(-b+sqrt(b*b-4*a*c))/(2*a);
x2 =
(-b-sqrt(b*b-4*a*c))/(2*a);
x1 = -b / (2*a);
x2 = sqrt(-(b*b - 4*a*c)) /
(2*a);
```
問 179：
```
a+b+c == 17
a+b+c
```
（略）a, b, c にそれぞれ 1 から 10 の数字をすべて代入し、式を満たすかどうかを判定している。そのため、if 文の条件式は $10 \times 10 \times 10$ 回評価されることになる。しかし、1+8+8 も 8+1+8 も 8+8+1 も同じ解の組み合わせであるのに、別個に計算されている。

これを改善するには、`b = 1;` と `c = 1;` の部分をそれぞれ `b = a;` と `c = b;` に書き換える。なぜこれで同じ組合せを計算しなくなるのか、またなぜ効率が良くなるのか（つまり if 文の条件式が評価される回数は何回になるのか）は各自考えること。

問 180：
```
1+1+5=7
```
（略）flag が 0 になったら処理がすべて終了する。

問 181：
```
0 1 1 1 4
```
問 182：
```
17 34 51 68 85
```
（略）100 以下の 17 の倍数を求めている。

問 183：
```
21 42 63 84
```
（略）100 以下の 3 と 7 の公倍数を求めている。

問 184：
```
x % 10;
```
問 185：
```
% 10;
```
問 186：
```
(x / 100) % 10;
```
問 187：
```
Today is fine.Today is f
```
問 188：
```
ay is fine.Tod
```
問 189：
```
n <= 100
n % 5 == 0 && n % 7 == 0
```
問 190：
```
n > 1
n - 1;
```
問 191：
```
45
n > 1
630 % n == 0 && 45 % n == 0
n - 1;
```
（略）ユークリッドの互除法。

問 192：
```
x -= 1;
x += y;
y *= z * 2;
z = z / (x + 4);
y = y + 2 * x + 3 * z;
x = x * (z + 2);
```
問 193：
```
0 2 4 6 8
```
問 194：
```
85 68 51 34 17
```

問 195 :
```
1 2 6 24 120
```
（略）1 から 5 までの階乗 (1!, 2!, 3!, 4!, 5!) を計算している。
問 196 :
```
i++;
x--;
x++;
y--;
z = z + 1;
z = z - 1;
```
問 197 :
```
0 1 2 3 4 5 6 7 8 9
```
（略）0 から 10 未満の整数を表示している。
問 198 :
```
2 4 6 8
```
（略）配列の要素で先頭から表示し、10 以上があったら終了する。
問 199 :
```
12 10 8 6 4 2
```
（略）配列の内容を逆順に表示している。
問 200 :
```
i < 10
k < a[i]
```
問 201 :
```
12 2
10 4
8 6
6 8
4 10
2 12
```
（略）配列の内容を先頭からと末尾からそれぞれならべて表示している。
問 202 :
```
0 1 2 3 4 5 6 7 8 9
```
問 203 :
```
0 7 14 21 28 35 42 49 56 63
```
問 204 :
```
a[i] < 10
```
問 205 :
```
i >= 0
i = 5 ; i >= 0 ; i--
```
問 206 :
```
bcde
```
問 207 :
```
0 1 2 3 4 5 5 4 3 2 1 0
```
問 208 :
```
2 4 6 8 10
```
問 209 :
```
i += 2
```
問 210 :
```
i -= 2
```
問 211 :
```
i < 5 ; i++
```
または
```
a[i] < 10 ; i++
```
問 212 :
```
i = 1 ; i <= 100 ; i += 2
```
または
```
i = 99 ; i > 0 ; i -= 2
```
問 213 :
```
float f[4] = {1.23, 4.56, 78.9, 10.11};
int i;
for(i = 0; i < 4; i++){
   printf("%.2f ", f[i]);
}
```
問 214 :
```
55
```
（略）1 から 10 までの和を計算している
問 215 :
```
int i, f = 1;
  for(i = 1; i < 4; i++){
     f *= i;
  }
printf("%d\n", f);
```
問 216 :
```
(year %4 == 0 && year % 100 != 0) || (year % 400 == 0)
```
問 217 :
```
n is 12
```
問 218 :
```
i = 0; i < 128; i++
```
または
```
i = 0; i <= 127; i++
```
問 219 :

$$\sum_{i=-2}^{99} i$$

問 220 :
```
8
5 * i + 1
```
199
問 221 :
```
a = 2 * a + 1;
```
問 222 :
```
1 2
2 1
1 2
2 1
1 2
2 1
```
（略）t を媒介して x1 と x2 の値を交互に入れ換えている。
問 223 :
```
1 2 3
2 3 0
3 0 2
```

```
0 2 4
2 4 6
4 6 8
```

（略）x1,x2,x3 の値を順次 2 倍して入れ換えている。

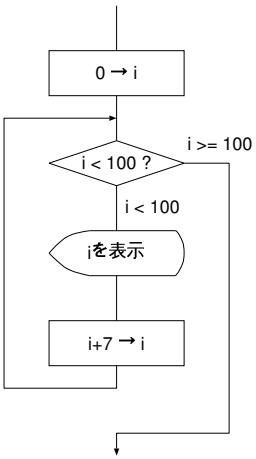
問 224 :

```
an2 = an1;
an1 = an;
```

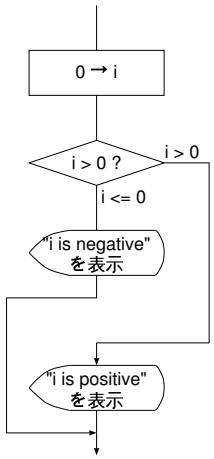
問 225 :

```
an1 = an;
bn1 = bn;
```

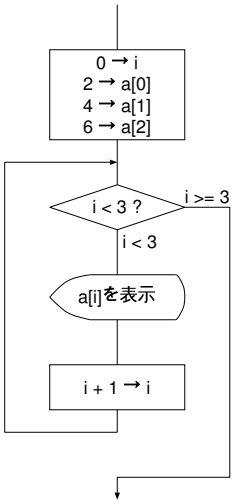
問 226 :



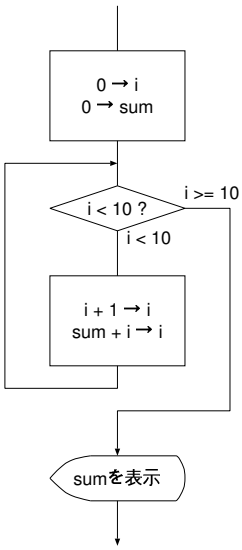
問 227 :



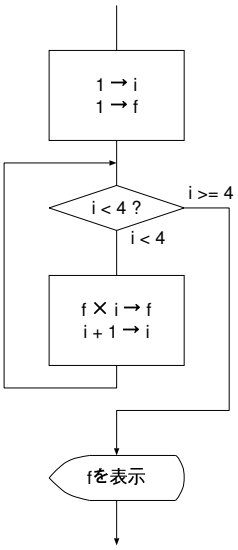
問 228 :



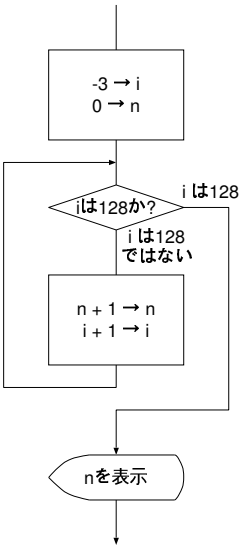
問 229 :



問 230 :



問 231 :

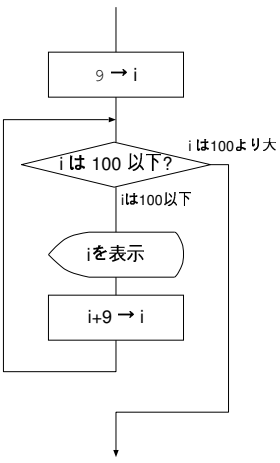


問 232 :

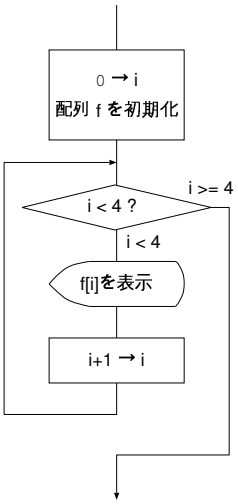
```
int i = -2, n = 0;
while(i < 150){
  n += 2 * i + 4;
  i++;
}
printf("%d\n", n);
```

```
int i, n = 0;
for(i = -2; i < 150; i++){
  n += 2 * i + 4;
}
printf("%d\n", n);
```

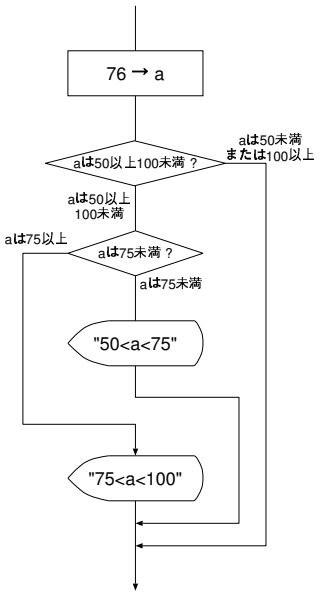
問 233 :



問 234 :



問 235 :



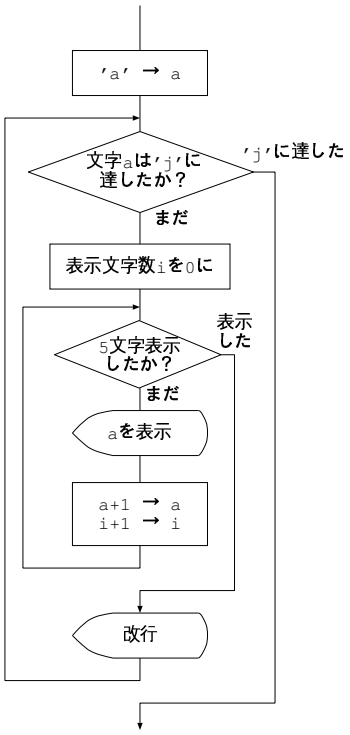
問 236 :

```
float y = 0, x;
while(y < 5){
  x = 0;
  while(x < 5){
    printf("%f\n", 2*x*y + 1);
    x += 0.5;
  }
  y += 0.5;
}
```

```
float y, x;
for(y = 0; y < 5; y += 0.5){
  for(x = 0; x < 5; x += 0.5){
    printf("%f\n", 2*x*y + 1);
  }
}
```

問 237 :

```
abcde
fghij
```



問 238 :

x の値	y の値	p の値
不定	不定	不定
不定	不定	x
3	不定	x
3	不定	y
3	4	y

問 239 :

p の値	x の値	y の値	z の値
未定義	不定	不定	未定義
不定	不定	不定	不定
1	不定	不定	不定
1	不定	不定	2
1	p	不定	2
1	p	p	2
1	z	p	2

問 240 :

x の値	y の値	p の値	*p の値
不定	不定	不定	不定
3	不定	不定	不定
3	不定	x	3
4	不定	x	4
4	5	x	4
4	5	y	5
4	6	y	6

問 241 :

x	y	p1	*p1	p2	*p2
不定	不定	不定	不定	不定	不定
3	不定	不定	不定	不定	不定
3	5	不定	不定	不定	不定
3	5	不定	不定	x	3
3	5	x	3	x	3
8	5	x	8	x	8
8	5	y	5	x	8
8	7	y	7	x	8
1	7	y	7	x	1

問 242 :

a[0]	a[1]	p1	*p1	p2	*p2
不定	不定	不定	不定	不定	不定
不定	不定	a[0]	不定	不定	不定
不定	不定	a[0]	不定	a[1]	不定
不定	3	a[0]	不定	a[1]	3
5	3	a[0]	5	a[1]	3
5	8	a[0]	5	a[1]	8
11	8	a[0]	11	a[1]	8
11	8	a[0]	11	a[0]	11
0	8	a[0]	0	a[0]	0

問 243 :

```
4.56, 1.23
78.90, 4.56
```

問 244 :

```
1.23 1.23
4.56 4.56
78.90 78.90
10.11 10.11
```

問 245 :

```
1,2,3,4,5,6,
7,6,5,4,
```

問 246 :

z	x	*x	w	*w	**w
不定	不定	不定	不定	不定	不定
不定	不定	不定	x	不定	不定
不定	z	不定	x	z	不定
3	z	3	x	z	3

問 247 :

z	x	*x	w	*w	**w
不定	不定	不定	不定	不定	不定
不定	z	不定	不定	不定	不定
不定	z	不定	x	z	不定
7	z	7	x	z	7
2	z	2	x	z	2

問 248 :

```
I’m a student.
Yes, I am.
No, I’m not.
```

問 249 :

```
I’m a student.
Yes, I am.
```

（略）p1 を始点、p2 を終点とする範囲を表示する。

問 250 :

```
I’m aYes, No, I
```

ここでの p は、後に出てくる main 関数の第二引数 *argv[] と同じ。

プログラミング基礎実習 問題集

平成 17 年 6 月 14 日

新潟大学 工学部 情報工学科

玉木徹 tamaki@ie.niigata-u.ac.jp