C言語 ホームワーク

福山平成大学経営情報学科

福井正康

序

近年，Ｃの地位は不動のものとなり，パソコンレベルにおいてもプログラム開発にかなり利用されてきています。また情報処理試験でも採用され，利用者の数はこれからますます増えることが予想されます。さらに，Ｃ関係の書籍も書店にあふれ，参考書にこと欠かない状態です。このような状況においてもプログラム言語を教育する側の人間には困ったことが１つあります。それは参考書に比べて，演習書が以外と少ないということで，学生の課題用の簡単な題材選びに苦労します。

この本は特にプログラムに詳しくない人が，参考書を利用しつつ実際にプログラムを作成・実行してみるための演習書です。著者らは一部の例外を除けば，プログラムは「習うより慣れろ」であると信じています。この本では徹底的に基礎的な問題から出発して，だんだんと体を慣らしていきます。特に初心者の方は初めから自力で問題を解いて下さい。必ずや複雑なプログラムにも耐え得る基礎体力が付いてくることでしょう。なるべく解答を見ないようにするために，この本では問題と解答を分離しています。解答はスマートさよりも分かり易さを優先させていますが，コンパクトに出来る部分は注意書きのところに書いているつもりです。また覚えにくい事項は簡易資料として載せています。プログラムは LSIC86 を想定しています。問題は基本的な問題と少し考える問題に分け，それぞれ問題番号の後ろに a と b を付けています。この演習でプログラミングの考え方を少しでも理解していただけたら幸いです。

福山平成大学経営学部経営情報学科
福井正康

問　題　編

1章 入出力

1-1) 文字列の表示

問題 1-1-1a

Hello world!!と表示する。

問題 1-1-2a

We study と every day.を各々printf 文で書き，全体を 1 行に表示する。

問題 1-1-3a

BASIC と PASCAL を各々printf 文で書き，2 行に表示する。

問題 1-1-4a

1 つの printf 文で LISP の後に 2 行改行して PROLOG と表示する。

問題 1-1-5b

printf("%d\n",a); と表示する。

問題 1-1-6b

エスケープシーケンスを利用して全（テキスト）画面を消去する。

1-2) 整数型と倍長整数型

問題 1-2-1a

数 12345 を表示する。

問題 1-2-2a

数 56321 を表示する。

問題 1-2-3b

数 1234567 を表示する。

問題 1-2-4a

変数を使わずに 234×56 を求める。

問題 1-2-5b

変数を使わずに 2222×2222 を求める。

問題 1-2-6a

a を 10，b を 2 として和，差，積，商を求めて表示する。

問題 1-2-7a

3567÷24 の商と余りを求める。

問題 1-2-8a

a を 25，b を 4 として(a+b)/(a-b)を求める。

問題 1-2-9a

a と b を 30，c を 90 として(c+a*b)/(c-(a+b))を求める。

問題 1-2-10b

a を 139，b を 222，c を 165 として，その積を求める。

1-3）8，10，16 進数

問題 1-3-1a

164 を 8 進数，10 進数，16 進数で表示する。

問題 1-3-2a

164 を 16 進数で A4H のように大文字で最後に H を付けて表示する。

問題 1-3-3a

a を 49，b を 19 として a*b を 10 進数と 16 進数で表示する。

問題 1-3-4a

a を $2a_H$，b を $3b_H$ として a，b を用いて 2aH+3bH=65H と表示する。

問題 1-3-5a

a を $ff22_H$ として 10 進数で表示する。(-222)

1-4）実数型と倍精度実数型

問題 1-4-1a

a を 5.34，b を 2.51 として和，差，積，商を求める。

問題 1-4-2a

a を 543.210987，b を 12.3456789 として和，差，積，商を求める。

問題 1-4-3a

円周率 3.14159 を変数 pi に代入し，半径 25.31 を r に代入して，円周の長さと円の面積を求める。

問題 1-4-4a

x を 1.3524986，y を 4.1259354 として x^2+y^2 を求めて表示する。

問題 1-4-5a

a を 12.76，b を 2561.2368914 として，各々を指数(%e)表示する。

問題 1-4-6a

a を 3.452，b を 5.786 として c に a*b を代入し，a と b を%f 指定，c を%e 指定で表示する。

問題 1-4-7a

a を 1.245×10^{23}，b を 2.567×10^{-123} として，各々を表示する。

問題 1-4-8a

a を 1.245×10^{23}，b を 2.469×10^{29} として，a*b を変数 c に代入し表示する。

1-5）キャスト演算子（強制的な型変換）

問題 1-5-1a

a を int 型の 20，b を float 型の 15.3114 として，和を実数として求め，表示する。

問題 1-5-2a

b を 35.425，c を 52.954 として b+c を整数にして a1 に代入，b と c を各々整数にして和を求め，a2 に代入して a1 と a2 を表示する。

問題 1-5-3a

b を 21.234，c を 33.125 として b*c を倍精度実数にして表示し，b と c を各々倍精度実数にして積を求め表示する。

問題 1-5-4a

a を 3，b を 2，c を 4 として a+b/c の中で 1 箇所(float)を入れて，答 3.5 を

表示する。a+(b/c)ではどうか。

問題 1-5-5a

a を 3，b を 2，c を 4 として a/b+b/c*a の中に 2 箇所(float)を入れて，答 3 を表示する。

1-6）文字とアスキーコード

問題 1-6-1a

変数 a に'A'，b に'B'，c に'C'を代入して，1 桁ずつ空けて 1 行に A B C を表示する。

問題 1-6-2a

Fukuyama の 1 文字ずつを変数に代入して Fukuyama と表示する。

問題 1-6-3a

変数 a,b,c に'A','a','0'を代入してそのアスキーコードを 10 進数と 16 進数で各々表示する。

問題 1-6-4a

変数 a，b，c に 65，98，53 を与えて，そのアスキーコードの文字を表示する。

問題 1-6-5a

変数 a，b，c，d，e に B，A，S，I，C のアスキーコードを代入して，%c 指定で BASIC と表示する。

問題 1-6-6b

変数 a に'a'-'A'，b に'x'-'X'を代入して a，b を 16 進数で表示する。

問題 1-6-7b

変数 c に'X'を代入して小文字 x を c を用いて表示する。

問題 1-6-8b

basic の 1 文字ずつを大文字 BASIC に変換して表示する。

1-7）書式指定

問題 1-7-1a

1，10，100，1000 を 5 桁右詰めと左詰めで表示する。

問題 1-7-2a

19.2345，1245.87654 を整数部 6 桁，小数点以下 3 桁で表示する。

問題 1-7-3a

12345.3445 を小数点以下 5 桁で表示する。

問題 1-7-4a

987654321 を小数点以下 4 桁の指数表現で表示する。

問題 1-7-5a

987654321 を全体 15 桁小数点以下 4 桁の指数表現で表示する。

問題 1-7-6a

変数 a に 38，変数 b に 5 を代入して，a，b を用いて以下のように表示する。

```
38/5=7...3
```

問題 1-7-7a

変数 a に'A'を代入し，a を用いて以下のように表示する。

```
A の ASCII コードは 41H です
```

1-8）入力

問題 1-8-1a

整数を１つ入力して表示する。

問題 1-8-2a

16 進数を１つ入力して 10 進数と 16 進数で表示する。

問題 1-8-3a

倍長整数を１つ入力して表示する。

問題 1-8-4a

実数を１つ入力して表示する。

問題 1-8-5a

倍精度実数を１つ入力して表示する。

問題 1-8-6a

文字を１つ入力して表示する。

問題 1-8-7a

３つの実数を３つの scanf 関数で入力して和を求めて表示する。

問題 1-8-8a

２つの整数を１つの scanf 関数で入力して商と余りを求めて表示する。

問題 1-8-9a

１つの整数と１つの実数を１つの scanf 関数で入力して積を求めて表示する。

問題 1-8-10a

５つの文字を１つの scanf 関数で入力して連続して表示する。

問題 1-8-11b

３つのカンマ(,)区切りの整数を１つの scanf 関数で入力して，和を求めて表示する。\ 区切りの場合はどうか。

問題 1-8-12b

9 桁の連続する数から 3 桁ずつ切り取って入力して表示する。先頭に-符号が付いた場合はどうなるか。

２章 条件判断

2-1) 条件判断

問題 2-1-1a

整数を１つ入力して正のときプラスと表示する。

問題 2-1-2a

整数を１つ入力して正のときプラス，負のときマイナス，0 のとき，ゼロと表示する。

問題 2-1-3a

整数を１つ入力してその絶対値を表示する。

問題 2-1-4a

整数を１つ入力して，偶数か奇数か表示する。

問題 2-1-5a

文字を１つ入力して，'Y'の時 YES，'N'の時 NO，その他の時は?と表示する。

問題 2-1-6a

文字を１つ入力して，'Y'または'y'の時 YES，その他の時は NO と表示する。

問題 2-1-7a

整数を１つ入力して 0 以上 100 未満なら領域内，それ以外は領域外と表示する。

問題 2-1-8a

実数を１つ変数 x に入力して-2≤x≤-1 または 1≤x≤2 のとき域内，それ以外のとき領域外と表示する。

問題 2-1-9a

1 から 7 までの数字を入力すると，数字に応じて 1 から東京，名古屋，京都，大阪，岡山，広島，博多と表示し，それ以外は？と表示する。(switch case 文)

問題 2-1-10b

文字を１つ入力して，英大文字，英小文字，数字，その他を判定する。

問題 2-1-11b

質問 Qn に 1 か 0 で答えると場合に応じて以下のように表示する。他の答えの場合は ERROR!と表示する。

```
Q1:1 → Q2:1 →          Answer is A.
       Q2:0 →          Answer is B.
Q1:0 → Q3:1 → Q4:1 → Answer is C.
              Q4:0 → Answer is D.
       Q3:0 →          Answer is E.
```

３章 繰返し

3-1) for 文

問題 3-1-1a

自分の名前を 1 桁空白を開けて，5 個横に表示する。

問題 3-1-2a

1 から 10 まで改行して縦に表示する。

問題 3-1-3a

1 から 10 まで間に 1 つずつ空白を入れて横に表示する。

問題 3-1-4a

1 から 10 まで 4 桁右詰めで横に表示する。

問題 3-1-5a

10 から 1 まで 4 桁右詰めで横に表示する。

問題 3-1-6a

7 から 77 までの 7 の倍数を 4 桁右詰めで横に表示する。

問題 3-1-7b

100,80,60,40,20 と表示する。（カンマも含む）

問題 3-1-8b

繰り返しを使って ABCD･･･Z を表示する。

問題 3-1-9a

1 から 100 までの和を求める。(5050)

問題 3-1-10a

1+2+3+ ･･･ +10 と $1^2+2^2+3^2+$ ･･･ $+10^2$ を求める。(55，385)

問題 3-1-11a

1 から 100 までの偶数と奇数の和を求める。(2500，2550)

問題 3-1-12a

1!，2!，3!，･･･，10!の値を表示する。(1，2，6，24，･･･)

問題 3-1-13a

1/2+2/3+3/4+ ･･･ +10/11 の結果を表示する。(7.980)

問題 3-1-14a

1×2×3+4×5×6+7×8×9+ ･･･ +28×29×30 の結果を表示する。(71610)

問題 3-1-15a

$a_1=1$，$a_n=5a_{n-1}-1$ のとき，a_{10} の値と a_{10} までの和 S_{10} を表示する。(1464844, 1831057)

問題 3-1-16a

自分の名字を漢字で横に 5 個，縦に 10 行，表のように出力する。

問題 3-1-17a

1 から 100 までの数で以下のように 10 行 10 列の表を出力する。

```
  1   2   3   4   5   6   7   8   9  10
 11  12  13  14  15  16  17  18  19  20
                ・・・・・・・・・・・・・・・・・・
 91  92  93  94  95  96  97  98  99 100
```

問題 3-1-18a

1×1+1×2+ ・・・ +9×8+9×9 のように九九を全部たしあわせた結果を表示する。(2025)

3-2) while 文

問題 3-2-1a

0 が入力されるまで入力した整数の絶対値を繰り返し表示する。

問題 3-2-2a

-999 が入力されるまで入力した整数について合計を計算する。

問題 3-2-3b

-999 が入力されるまで入力した数について平均と分散を計算する。但し，平均と分散は以下の式で与えられる。

平均：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ ，分散：$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} {x_i}^2 - \bar{x}^2$.

問題 3-2-4b

1+2+3+ ・・・ +n<500 となる最大の n とその時の和を求める。(31，496)

問題 3-2-5b

入力した整数のその値以外の最大の約数を求める。

４章 配列とポインタ

4-1）配列

問題 4-1-1a

整数型配列 a[10]を 0 から 9 で初期化し表示する。

問題 4-1-2a

整数型配列 a[10]に 0 から 9 までの値を代入し表示する。

問題 4-1-3a

以下の値で配列 a[5]，b[5]を初期化し，各成分の和を配列 c[5]に代入して表示する。

a[5] ： 1，2，3，4，5
b[5] ： 3，4，5，6，7

問題 4-1-4a

整数型配列を 3，9，1，4，3，7，8，5 で初期化して，その和を求める。

問題 4-1-5a

5，4，6，8，2，3，5，1，3，6，2，4，8，9，9，7 で整数型配列を初期化し，最大と最小を求める。

問題 4-1-6a

実数型配列 a[5]を 2.0，4.0，6.0，8.0，10.0 で初期化し，配列の成分 a[i]と*(a+i)の形を用いて 2 通りで表示する。

問題 4-1-7a

以下の値で整数型配列 a[4][3]を初期化し表示する。

$$\begin{pmatrix} 34 & 17 & 44 \\ 24 & 59 & 47 \\ 91 & 11 & 61 \\ 89 & 22 & 65 \end{pmatrix}$$

問題 4-1-8a

問題 4-1-7 の表を整数型配列 a[4][3]に代入して，行と列を入れ換えて表示する。

問題 4-1-9a

以下の表のデータで配列 a[3][2]，b[3][2]を初期化して，それぞれの項の和の配列を作り表示する。

$$\begin{pmatrix} 32 & 55 \\ 48 & 73 \\ 63 & 35 \end{pmatrix} \qquad \begin{pmatrix} 24 & 23 \\ 64 & 79 \\ 98 & 18 \end{pmatrix}$$

問題 4-1-10a

行列 $\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 2 & 8 \end{pmatrix}$ を配列 a[2][2]に格納して，行列式の値を求める。

ただし，$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の行列式は $a \times d - b \times c$ である。

問題 4-1-11a

問題 4-1-10 の行列の逆行列を求める。

ただし，$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の逆行列は $\dfrac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ で与えられる。

問題 4-1-12a

文字 PROGRAM を各々文字型配列 a[0]～a[6]に代入して%c で続けて表示する。

問題 4-1-13a

文字型配列を PROLOG の各１文字ずつで初期化して，%c で続けて表示する。

問題 4-1-14a

文字型配列を PROLOG の各１文字ずつで初期化して，%s で続けて表示する。

問題 4-1-15a

文字列"BASIC"で配列 a[6]を初期化し，全体を%s 指定で表示した後１文字ずつ取り出して１桁置きに表示する。

問題 4-1-16b

配列 a[30]に文字列を入力し，その文字列の字数を表示する。

問題 4-1-17a

以下のデータを１文字ずつ２次元文字型配列に代入して%s で表示する。

Tokyo，Hiroshima，Okayama

問題 4-1-18a

以下のデータで２次元文字型配列を初期化して%s で表示する。

Tokyo，Nagoya，Osaka

問題 4-1-19a

BASIC，PASCAL，C，LISP の 4 文字列の先頭ポインタをポインタ配列に代入して表示する。

問題 4-1-20a

BASIC，PASCAL，C，LISP の 4 文字列の先頭ポインタでポインタ配列を初期化し表示する。

4-2）ポインタ

問題 4-2-1a

a を整数型変数とし，そのポインタの値を%p で表示する。

問題 4-2-2a

a を整数型変数とし，p を整数型ポインタとする。a に 7 を代入し p に a のポインタを代入して p を用いて 7 を表示する。

問題 4-2-3a

a を整数型変数とし，p を a を指すポインタとする。*p の値として 8 を代入し，a の値を表示する。

問題 4-2-4a

文字型ポインタ変数 p，q に"北海道"，"沖縄"の先頭ポインタを代入して文字列を表示し，それぞれのポインタを交換して再度表示する。

問題 4-2-5a

整数型配列 a[5]を 6，3，4，2，1 で初期化し，各数値を[記号を用いずに表示する。

問題 4-2-6a

文字型配列 str[6]を文字 Tokyo で初期化し，各文字を[記号を使う場合と使わない場合両方で表示する。

問題 4-2-7a

文字型配列 str[6]を文字 Tokyo で初期化し，各文字を表すポインタを[]記号を使わない場合と使う場合両方で表示する。

問題 4-2-8b

c[10]，a[10]，x[10]をそれぞれ文字型配列，整数型配列，実数型配列とするとき，それぞれの配列の先頭ポインタと 2 番目の要素を指すポインタを求め，各型のバイト数を確認する。

問題 4-2-9b

a[13]を"東京にいます"として文字列全体を%s で表示し，その後"います"の部分だけを%s で表示する。

問題 4-2-10b

文字型配列 c[20]の中に"Tokyo"と"Kyoto"の２つの文字列を同時に代入し，それぞれの文字列を%s を利用して表示する。

問題 4-2-11a

2 次元整数型配列 a[2][3]の各要素を指すポインタを全て表示する。

問題 4-2-12b

2 次元整数配列 a[2][3]を以下のデータで初期化し，各数字を[を用いずに表示する。

{{1，2，3}，{7，8，9}}

問題 4-2-13a

文字型ポインタ配列に"Osaka"と"Hiroshima"の先頭ポインタを代入し，各文字を%c を用いて表示する。

5 章 関数

5-1) 関数１

以下の仕様の関数を作り，動作を確認せよ。

問題 5-1-1a

関数　void cls(void)
機能　画面をクリアする。

戻値　なし

問題 5-1-2a

関数　int spc(int n)
機能　n 個のスペースを出力する。
戻値　n>0 のとき:1, n≤0 のとき:0

問題 5-1-3a

関数　double dabs(double x)
機能　引数の絶対値を返す。
戻値　引数の絶対値

問題 5-1-4a

関数　double ipow(double x, int n)
機能　x^n を返す。
戻値　x^n

問題 5-1-5a

関数　int dispvec(int *c, int n)
機能　配列 c[0]～c[n-1]の値を１つずつ改行して出力する。
戻値　n>0 のとき:1, n≤0 のとき:0

問題 5-1-6a

関数　int set0(int b[], int n)
機能　配列 b[0]～b[n-1]を 0 にセットする。
戻値　n>1 のとき:1, n≤0 のとき:0

問題 5-1-7a

関数　void setw(int *b, int m, int n)
機能　配列 b[m][n]をすべて 0 で初期化する。
戻値　なし

問題 5-1-8a

関数　int strlength(char *str)
機能　文字列配列 str[]の文字数を数える。
戻値　文字数（最後の 0x0 は含めない）

問題 5-1-9a

関数　void dabs2(double x, double *ret)
機能　x の絶対値をポインタ ret に返す。
戻値　なし

問題 5-1-10a

関数　int strcopy(char *str, char *ret)
機能　文字列配列 str[]の内容を配列 ret[]にコピーする。
戻値　コピーした文字数（最後の 0x0 は含めない）

問題 5-1-11a

関数　int strsl(char *str, char *ret)
機能　文字列配列 str[]中の小文字を大文字にして配列 ret[]に返す。
戻値　小文字から大文字に換えた文字数

5-2) 関数２

以下の関数の仕様を考えて関数を作成し、動作を確認せよ。

問題 5-2-1a

座標(x,y)の原点からの距離が指定した値以上か未満か判定する。

問題 5-2-2a

扇型の面積を求める。

問題 5-2-3a

n!の値を求める。

問題 5-2-4a

文字列ポインタ配列の指定した数の文字列を改行しながら表示する。

問題 5-2-5b

整数型配列中のデータの最大値と最小値を求める。

問題 5-2-6b

２つの文字列を１つにつないで新しい文字列を作る。

問題 5-2-7b

文字列配列中から指定した文字を探す。

5-3) main 関数の引数

問題 5-3-1a

main 関数の引数の数とその文字列を表示する。

問題 5-3-2a

main 関数の引数として２つの文字列をとり，それらを改行して表示する。ただし，２つ以外は引数個数のエラーと出力する。

問題 5-3-3a

main 関数の引数として１つの文字列をとり，その文字列の先頭が'-'または'/'なら YES と表示し，それ以外なら NO と表示する。ただし，２つ以上の引数の場合は ERROR!で，引数のない場合は ARGUMENT!と表示する。

問題 5-3-4a

main 関数の引数として適当な数の文字列をとり，それらの中に"/a"か"/A"が入っていたら YES，入ってなかったら NO と表示する。

６章 標準ライブラリ関数

6-1) 標準ライブラリ関数

問題 6-1-1a

文字列を入力し，strupr()関数を用いて小文字を大文字に変えて表示する。

問題 6-1-2a

getch()関数でキーボードから１文字ずつ読み込み，isalnum()関数を使って英数字かどうか判定し，英数字なら putchar()関数を用いて表示する。また，ESC なら終了する。

問題 6-1-3a

getcwd()関数を用いてカレントドライブ･カレントディレクトリを取得して表示する。

問題 6-1-4a

atoi()関数を用いて数字のならんだ文字列を整数型数値に変換し，itoa()関数を用いて 10 進数表示で文字列に戻す。

問題 6-1-5b

文字列または数字をキーボードから読み込む際最初の１文字を getchar()関数を用いて読み，ungetc()関数を使って，１文字分標準入力(stdin)に返し，その文字が数字か英字かで整数または文字列として読み込み表示する。

例 ： 123 は"数値=123"，abc は"文字=abc"と分けて表示する。

問題 6-1-6a

log()関数と log10()関数を用いて,0.1 から 2.0 まで 0.1 きざみで自然対数値と常用対数値を表示する表を作成する。

問題 6-1-7a

time()関数を用いて 1970/1/1 00:00:00 から現在までの通算秒数を表示し、さらに ctime()関数を用いてそれを時刻データの文字列に変換して表示する。

問題 6-1-8a

srand()関数を用いて乱数を初期化し、rand()関数によって、0 以上 1 未満の乱数及び、100 以上 200 未満の整数乱数をそれぞれ 100 個出力する。但し、乱数の初期化の seed には 1 を用いる。

問題 6-1-9b

rand()関数を用いて半径 1 の円の面積をモンテカルロシミュレーションで求める。精度は練習であるので 1000 回程度のループでよい。但し、乱数の初期化の seed には 1 を用いる。

ヒント ：半径 1 の円の第 1 象限部分で(0,0)-(1,1)の正方形考える。この正方形の面積は 1 である。この中へ乱数で点を打ち，このうち円の内側へ何割入ったかで第 1 象限部分の円の面積を推測出来る。円全体ではその 4 倍であるから，円の面積=4×1×割合。

問題 6-1-10a

以下の文字列をアスキーコードで比較し，最小と最大の文字列を求める。
ただし，比較には strcmp()関数を用いる。

OKAYAMA, OSAKA, KYOTO, NAGOYA, TOKYO

問題 6-1-11a

２つの文字型配列に strcpy()関数を用いてそれぞれ"いちご"と"大福"を代入し，その後 strcat()関数を用いて１つの文字列に連結して表示する。

問題 6-1-12b

BOOK という単語をいくつか含む文字列を gets()関数により読み込み，文字列中の BOOK を LETTER に変えて，新しい文字列として表示する。
ただし，文字列の検索には strstr()関数，文字列の連結には strcat()関数，文字列の文字数を調べるには strlen()関数を用いる。

7 章 演算子

7-1) 論理演算子

問題 7-1-1a

a を 0，b を 1，c を 2 としてそれぞれの論理演算の否定を求める。

問題 7-1-2a

a を 0，b を 1，c を 2 として 3 つの変数の各組合せに対して論理積，論理和，またこれらを使って作った排他的論理和の演算を行った結果を求める。

7-2) 代入演算子

問題 7-2-1a

a[i++]の形式を用いて配列 a[10]に 0～9 まで代入する。

問題 7-2-2a

a[++i]の形式を用いて配列 a[10]に 0～9 まで代入する。

問題 7-2-3a

+=演算子を用いて 1 から 100 までの合計を計算する。

問題 7-2-4a

*=演算子を用いて 10!を計算する。

問題 7-2-5a

最初に a=2 として a==4 と a=4 の式の値を表示する。

問題 7-2-6b

a が適当な文字列配列を指すポインタであるとして，次のプログラム

```
i=0; c[i]=*a;
while(*a!=0){i=i+1; a=a+1; c[i]=*a};
```

が　i=0; while(この中を考える);　の中に収まるように簡単化する。

7-3) シフト演算子

問題 7-3-1a

45 を 1bit 右へシフトした値と 3bit 左へシフトした値を 10 進数で求める。同じ演算を四則演算子で行うにはどうするか。

問題 7-3-2a

int 型及び，unsigned int 型の 0x2222 を 4bit ずつ左右へシフトした値を 16 進数で求める。0xaaaa の場合はどうか。

問題 7-3-3a

0x1234 の上位 8bit の値 0x12 をシフト演算子を用いて 16 進数で表わす。

問題 7-3-4a

a を 0x120 として 0x12 と 0x1200 をシフト演算子を用いて求める。

問題 7-3-5b

a を 0xff12 として 0xff をシフト演算子を用いて表す。

問題 7-3-6a

a を 200 として a/8 をシフト演算子を用いて表す。

問題 7-3-7a

a を 10 として a*64 をシフト演算子を用いて表す。

7-4) ビット演算子

問題 7-4-1a

0xaa55 のビットごとの NOT を求め 16 進数で表示する。

問題 7-4-2a

5 と 12 のビットごとの AND，OR，XOR(^演算子)を求める。

問題 7-4-3a

a が 0x1213 のとき，上位 8bit 分及び下位 12bit 分を 0 でマスクしてそれぞれを表示する。

問題 7-4-4a

a に整数を読み込んで，下位から 3bit 及び 8bit 目が 1 か 0 かを判定する。

問題 7-4-5a

a に 16 進整数を読み込んで，下位から 4bit 及び 8bit 目を強制的に 1 にして表示する。

問題 7-4-6a

a に 16 進整数を読み込んで，下位から 6bit 及び 12bit ビット目を強制的に 0 にして値を表示する。

問題 7-4-7b

a が 0x1234 のとき，上位 4bit を最後に持ってきて 0x2341 とするのをシフト演算子とビット演算子を用いて表す。

問題 7-4-8b

整数型データを引数としてデータを 2 進数で,最後に B を付けて表示する void 型関数を作成する。

8 章 構造体と共用体

8-1) 構造体

問題 8-1-1a

文字型のポインタ name，整数型の age，income をメンバとし，タグ名を shain とする構造体を定義し，次に shain 型の構造体変数 a を宣言して，name に文字列"佐々木康宏"の先頭のポインタ，age に 22，income に 380 を代入してメンバを表示する。

問題 8-1-2a

問題 8-1-1 と同じ構造体を同じデータで初期化し，メンバを表示する。

問題 8-1-3a

問題 8-1-1 のデータの入った構造体へのポインタを p として, この p を用いてメンバを表示する。

問題 8-1-4a

#typedef を用いて以下のデータを格納する構造体を SALES 型で定義し，それ

を用いて以下のデータを格納し表示する。

商品名	単価	売上
PR1100	148,000	8

問題 8-1-5a

以下のデータを構造体配列変数に代入して，表示する。

名前	年齢	年収	勤務歴
高尾政之	38	562	16
石丸敬二	30	430	8
鎌刈智恵	24	385	2

問題 8-1-6a

以下のデータで構造体配列変数を初期化して，表示する。

商品名	単価	売上
PR1100	148,000	8
PR2100	218,000	15
PR3100	284,000	4

問題 8-1-7a

getdate()関数と gettime()関数を用いて現在の（システムの）日付と時間を求めて 1991-01-10/15:30:10 の形で表示する。

8-2) 共用体

問題 8-2-1a

unsigned int 型のデータと long 型のデータで共用体を宣言し，読み込んだデータが 0x10000 より小さい場合は unsigned int 型へ，他の場合は long 型へ代入して表示する。

問題 8-2-2a

以下のデータを共用体で宣言し，データを各々代入しながら表示してゆく。

名前	年齢	年収
安部健	22	450

問題 8-2-3b

float 型のデータをキーボードから読込み、それを 2 進数で表示する。

問題 8-2-4b

int 型 1 つと、unsigned char 型 2 つよりなる構造体によって構成される共用体を用いて、int 型 1234_H のデータのメモリへの格納値を 1 バイトずつ抜き出して表示する。

問題 8-2-5b

以下の 2 つの構造体 wregs と bregs で共用体 myregs をグローバルに定義し ah, al, bx, ch, cl, dx にそれぞれ 10, 0, 1, 0, 1, 5 の値を代入し，表示する。

```
struct wregs {unsigned int ax,bx,cx,dx};
struct bregs {unsigned char ah,al,bh,bl,ch,cl,dh,dl};
```

問題 8-2-6b

問題 8-2-5 のデータの入った共用体へのポインタを p として, この p を用いてデータの入ったメンバを表示する。

9章 プリプロセッサ

9-1) プリプロセッサ

問題 9-1-1a

3.14159 を PI，printf を PRINT で置換し，半径 10 の円の面積を表示する。

問題 9-1-2a

scanf を INPUT，printf を PRINT で置換し，自分の名前を読み込んで表示する。

問題 9-1-3a

エスケープシーケンスを用いて文字色を緑にする命令を GREEN で, 標準色に戻す命令を NORMAL で置換し，動作を確認する。

問題 9-1-4a

エスケープシーケンスを用いてカーソルを指定した位置(x,y)に移動させるマクロ cjump(x,y)をプリプロセッサで作り，動作を確認する。但し，0≤x≤79, 0≤y≤24 とする。

問題 9-1-5b

テキストの色を変える命令を COLOR1～COLOR7，カーソルのジャンプを cjump(x,y)，カーソルの表示・非表示を CREP，CVAN として，#define と#if

defined を用いてＰＣ９８でもＦＭＲ（ＤＯＳ／Ｖ）でも１箇所の変更で使えるプロプロセッサを作り，動作を確認する。

問題 9-1-6c

以下のメニュー項目名を持つメニュー選択用の関数を作る。
ただし，矢印キーの上下によって項目名が反転表示され，RET キーによって反転された項目が決定されるようにする。最後に決定項目名を表示して終了する。
キー入力は getch()を用いる。項目名は文字列のポインタ配列のデータ形式を取る。

ファイル入力
ファイル出力
終　　了

１０章 ファイル処理

10-1) 高水準入出力

問題 10-1-1a

ファイル natu.dat に文字列"沖縄", "北海道"と改行して書く。

問題 10-1-2a

問題 10-1-1 で作成されたファイル natu.dat に"海！山！"と追加で書く。

問題 10-1-3a

以下のデータをファイル file3.dat に間に 1 桁ずつ空白を入れて書き, その後読み出して表示する。

1, 2, 3, 4, 5, 0, 9, 8, 7, 6

問題 10-1-4a

以下のデータをファイル file4.dat に 3 桁右詰めで書き, その後読み出して表示する。

115, 33, 257, 118, 5

問題 10-1-5a

以下のデータファイルを file5.dat に%10.3f で書き，その後読み出して表示する。

123.4, 44.6, -556.34, 0, 38.45

問題 10-1-6a

1～100 までの 100 個の整数乱数の偶数をファイル even.dat に奇数をファイル odd.dat に 1 桁ずつ空けて出力する。

問題 10-1-7a

データを-999 が入力されるまで打ち込みつつ，ファイル file7.dat に出力する。

問題 10-1-8a

テキストファイルを読み込んで表示する。

問題 10-1-9b

以下のカンマ区切りのデータをファイル file9.dat に出力する。その後最初のデータ数を参考にファイルからデータを読み込み，平均，分散，標準偏差を計算する。(5.40，4.04，2.01)

10　(データ数)

4,6,8,3,4,6,7,4,9,3

但し，平均：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$，分散：$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^{\,2} - \bar{x}^2$，標準偏差：$\sigma$．

問題 10-1-10b

以下のカンマ区切りのデータをファイル file10.dat に出力する。その後ファイルからデータを読み込み，平均，分散，標準偏差を計算する。これは問題 10-1-9 でデータ数を記入しない場合である。(5.40，4.04，2.01)

4,6,8,3,4,6,7,4,9,3

問題 10-1-11b

1～100 の整数を 4 桁でファイル file11.dat に出力する。その後 fseek()関数を利用して 90，59，32，87 番目の数を読み出して出力する。

問題 10-1-12b

1～100 の整数を 100 個 5 桁でファイル file12.dat に出力し，50，60 番目の数をマイナスに書き直す。

問題 10-1-13b

1～20 の整数を 4 桁で各々改行してファイル file13.dat に出力し，10 番目の数字を呼び出して出力する。

問題 10-1-14b

1～100 の整数を配列に保管し, ファイル file14.dat に fwrite()関数を用いて保存する。その後同ファイルからデータを freed()関数を用いて一括して読み出して出力する。

問題 10-1-15b

氏名, 年齢, 身長, 体重のデータを 3 組キーボードから構造体配列に入力して, ファイル file15.dat に fwrite()関数を用いて保存する。その後同ファイルからデータを fread()関数を用いて読み出して出力する。^Z はコントロール Z と読む。

問題 10-1-16a

文字列"嵐山の紅葉"と^Z(0x1a)をファイル aki.dat に書き, その後同じファイルに"保津川下り"と追加で書く。最後は^Z(0x1a)にする。

問題 10-1-17a

プリンタへきんかん，みかん，夏みかんとそれぞれ改行して出力する。

問題 10-1-18a

プリンタへすいかと横倍角文字で出力する。

10-2）低水準入出力

問題 10-2-1a

1～100 の整数を 100 個バイナリ形式でファイル bfile1.dat に低水準出力する。

問題 10-2-2a

問題 10-2-1 で作成したファイル bfile1.dat から 100 個のデータを低水準入力し表示する。

問題 10-2-3a

0.01～1.00 の 0.01 きざみの小数を 100 個バイナリ形式でファイル bfile3.dat に低水準出力する。

問題 10-2-4a

問題 10-2-3 で作成したファイル bfile3.dat から 100 個のデータを低水準入力し表示する。

問題 10-2-5b

問題 10-2-1 で作成したファイル bfile1.dat から 40，50，60 番目のデータを呼び出して表示する。

問題 10-2-6b

問題 10-2-1 で作成したファイル bfile1.dat の 30 番目のデータを 0 に換える。

問題 10-2-7b

低水準入出力を利用してファイルを別のファイル名で 1 文字ずつコピーする。

１１章 再帰処理

11-1) 再帰処理

問題 11-1-1a

n!の値を再帰呼び出しを用いて求める関数を作成する。

問題 11-1-2a

次の漸化式を再帰呼び出しを用いて解く。n の値を入力して，a_n を求める。

$a_1=2$，$a_n=3a_{n-1}+2$

問題 11-1-3a

次の漸化式を再帰呼び出しを用いて解く。n の値を入力して，a_n を求める。

$a_1=1$，$a_2=2$，$a_n=2a_{n-1}+a_{n-2}+1$

問題 11-1-4b

84=2×2×3×7 のようにある数を素数に分解することを素因数分解と言うが，再帰呼び出しを用いてこれを実行する。

問題 11-1-5a

3 本の柱の内の 1 本に n 枚の大きさの異なる円盤がはまっている。但し，円盤は必ず大きな円盤の上に小さな円盤を乗せるようにする。
このルールを守って 1 本の柱から他の 1 本の柱へ 1 枚ずつ円盤を移す手順を考える。この問題をハノイの塔の問題というが，これを再帰呼び出しを用いて解く。（殆どの教科書で解説されている）

問題 11-1-6b

ドライブ名を指定するとそのドライブのサブディレクトリ名を検索してすべ

て表示する。ディレクトリの検索には_dos_findfirst(), _dos_findnext()関数を使用する。

１２章 メモリ管理とデータ構造

12-1) メモリ管理

問題 12-1-1a

char 型のデータ 10 個分のメモリを動的に確保し，その中に"Turbo C"という文字列を代入し，表示する。

問題 12-1-2a

int 型のデータ 100 個分のメモリを動的に確保し, 1～100 までの数を代入した後表示する。

問題 12-1-3a

m と n をキーボードから読み込んで m 行 n 列分の int 型データのメモリを動的に確保し，1～m×n の整数をその中に代入し，行列の形で m 行 n 列で表示する。

問題 12-1-4a

double 型のデータ 100 個分のメモリを動的に確保し, 0.01～1.00 の 0.01 きざみの小数を入力した後，それらを表示する。

問題 12-1-5a

10 バイトのデータ 5 個分のメモリを malloc 関数で動的に確保し，"京都"という文字列を 5 個代入した後，表示する。

問題 12-1-6a

5 個分のポインタ配列に対してそれぞれ 10 バイト分のメモリを動的に確保し，それぞれに"大阪"という文字列を代入した後，表示する。

問題 12-1-7a

以下の構造体 5 個分のメモリを動的に確保し, 名前と年齢をキーボードから入力する。その後，3 番目の要素のデータを表示する。

```
struct name{
    char name[20];
    int age;
```

```
};
```

12-2) データ構造

問題 12-2-1b

以下の構造のデータを 100 個リスト的につないで，順番に 0 から 99 までの数を data の部分に入力た後，その値をリストの先頭から順番に表示する。最後に確保したメモリを解放して終了する。但し，sturct numlist 型は typedef を用いて DATA 型とする。

```
struct numlist{
    struct numlist *prev;
    int data;
    struct numlist *next;
}
```

問題 12-2-2b

問題 12-2-1 で作られたリストの１つの要素を消去する関数
DATA *dlist(DATA *)を作る。但し，引数は消去する要素へのポインタであり，戻値は最後の要素を除いて消去した要素の次のポインタである。

問題 12-2-3b

問題 12-2-1 で作られたリストの指定したポインタの前に１つの要素を挿入する関数 DATA *ilist(DATA *)を作る。但し，引数は指定ポインタとし，戻値は挿入された要素のポインタとする。

問題 12-2-4b

問題 12-2-1 で作られたリストの最後に１つ要素を加える関数
DATA *alist(DATA *)を作る。但し，引数はリスト内の任意の要素のポインタとする。

問題 12-2-5b

問題 12-2-1 のリストで先頭から m 番目の要素から続けて n 個の要素をリストから取り去る関数 int dellist(DATA *, int, int)を作る。戻り値はリストの先頭ポインタとする。

問題 12-2-6b

問題 12-2-1 のリストの前から m 番目の後に１つ要素を追加する関数

int addlist(DATA *, int)を作る。戻り値はリストの先頭ポインタとする。

問題 12-2-7b

以下の形の 100 個の構造体に 0 以上 1000 未満の乱数を 100 個代入し木構造のデータを作る。その後, 確保したメモリを解放して終了する。但し, left->data に小さい数を right->data に大きい数を設定する。また sturct tree 型は typedef を用いて DTREE 型とする。

```
struct tree {
    struct tree *left;
    int data;
    struct tree *right;
}
```

問題 12-2-8b

問題 12-2-7 のデータから最大と最小の値を求める関数 int maxtree(DTREE *) と int mintree(DTREE *)を作る。

問題 12-2-9b

問題 12-2-7 のデータで小さい順に値を表示する関数 void disptree(DTREE *) を作る。

問題 12-2-10b

問題 12-2-7 のデータから指定数 n 以上の数で n に最も近い値を求める関数 int searchtree(DTREE *, int)を作る。

１３章 割込み

13-1) ファンクションコール

問題 13-1-1b

ファンクションコール 08_H を用いて ESC が入力されるまで標準入力から文字を入力して, ファンクションコール 02_H を用いて入力した文字を標準出力に表示する。

問題 13-1-2b

ファンクションコール 19_H を用いてカレントドライブを取得して A:, B:の形で表示する。

問題 13-1-3b

ファンクションコール 47_H を用いてカレントドライブのカレントディレクトリを取得して表示する。

問題 13-1-4b

ファンクションコール $2A_H$ と $2C_H$ を用いて日付と時刻を取得し，1991-01-10/15:30:10 の形で表示する。

問題 13-1-5b

ファンクションコール $0B_H$ と $0C_H$ を用いてキーボードバッファにデータがたまっているか確認し，たまっていればクリアする関数 void keyclear(void)を作る。

解　答　編

１章 入出力

1-1) 文字列の表示

解答 1-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("Hello world!!\n");
5.  }
```

注意 1-1-1

1. stdio.h は printf()関数使用時のインクルードファイル

解答 1-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("We study ");
5.      printf("every day.\n");
6.  }
```

解答 1-1-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("BASIC\n");
5.      printf("PASCAL\n");
6.  }
```

解答 1-1-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("LISP\n\nPROLOG\n");
5.  }
```

解答 1-1-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("printf(\"%%d\\n\",a);");
5.  }
```

注意 1-1-5

4. \ 記号: \\, " 記号: \", % 記号: %%

解答 1-1-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("\x1b[2J");
5.  }
```

注意 1-1-6

4. \x1b: 16 進コード 1B (ESC) のエスケープ文字
   <ESC>[2J は画面消去・カーソルホームポジションのエスケープシーケンス

1-2) 整数型と倍長整数型

解答 1-2-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.  int a=12345;
5.      printf("%d\n",a);
6.  }
```

注意 1-2-1

5. %d: int 型 10 進数表示

解答 1-2-2

```
1.  #include <stdio.h>
2.  main()
3.  {
```

```
4.      long a=56321L;
5.      printf("%ld\n",a);
6.  }
```

注意 1-2-2

4. long 型の定数には後ろに L を付ける
   unsigned int 型でも可能
5. %ld: long 型 10 進数表示
   unsigned int 型の場合は%u

解答 1-2-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long a=1234567L;
5.      printf("%ld\n",a);
6.  }
```

注意 1-2-3

4. long 型の定数には後ろに L を付ける
5. %ld: long 型 10 進数表示

解答 1-2-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("%d\n",234*56);
5.  }
```

解答 1-2-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      printf("%ld\n",(long)2222*2222);
5.  }
```

注意 1-2-5

4. 値が int 型の範囲-32768～32767 を越えるので long 型にキャスト
   2222L*2222L として予め値を long 型に指定してもよい

解答 1-2-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=10,b=2;
5.      printf("和 = %d\n",a+b);
6.      printf("差 = %d\n",a-b);
7.      printf("積 = %d\n",a*b);
8.      printf("商 = %d\n",a/b);
9.  }
```

解答 1-2-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=3567,b=24;
5.      printf("商   = %d\n",a/b);
6.      printf("剰余 = %d\n",a%b);
7.  }
```

解答 1-2-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=25,b=4;
5.      printf("%d\n",(a+b)/(a-b));
6.  }
```

解答 1-2-9

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=30,b=30,c=90;
5.      printf("%d\n",(c+a*b)/(c-(a+b)));
6.  }
```

注意 1-2-9

5. 演算は *,/,%(余り) が +,- より優先
   同じ優先順位の演算子の場合は左から実行

解答 1-2-10

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=139,b=222,c=165;
5.      printf("%ld\n",(long)a*b*c);
6.  }
```

注意 1-2-10

5. 値が int 型の範囲を越える
   long 型と int 型の計算は long 型で実行される

1-3) 8，10，16 進数

解答 1-3-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=164;
5.      printf("%o\n",a);
6.      printf("%d\n",a);
7.      printf("%x\n",a);
8.  }
```

注意 1-3-1

5. %o: 8 進数表示
6. %d: 10 進数表示
7. %x: 16 進数表示（小文字）

解答 1-3-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=164;
5.      printf("%XH\n",a);
6.  }
```

注意 1-3-2

5. %X は大文字 16 進数表示(A4)

%x は小文字 16 進数表示(a4)

解答 1-3-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=49,b=19;
5.      printf("%d\n",a*b);
6.      printf("%x\n",a*b);
7.  }
```

解答 1-3-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a=0x2a,b=0x3b;
5.      printf("%xH+%xH=%xH\n",a,b,a+b);
6.  }
```

注意 1-3-4

4. プログラム中で 16 進数は先頭に 0x を付ける

解答 1-3-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0xff22;
5.      printf("%d\n",a);
6.  }
```

注意 1-3-5

4. int 型だと負の値となる
   先頭ビットの 0/1 よって符号の正負が決まる

1-4) 実数型と倍精度実数型

解答 1-4-1

```
1.  #include <stdio.h>
2.  main()
```

```
3.  {
4.      float a=5.34,b=2.51;
5.      printf("和 = %f\n",a+b);
6.      printf("差 = %f\n",a-b);
7.      printf("積 = %f\n",a*b);
8.      printf("商 = %f\n",a/b);
9.  }
```

注意 1-4-1

5-8. %f:浮動小数点表示

解答 1-4-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=2543.210987,b=312.345678;
5.      printf("和 = %f\n",a+b);
6.      printf("差 = %f\n",a-b);
7.      printf("積 = %f\n",a*b);
8.      printf("商 = %f\n",a/b);
9.  }
```

注意 1-4-2

4. float 型では桁数不足
   double 型は有効数字約 15 桁

5-8. 表示桁数は有効桁数より小さい

解答 1-4-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float pi=3.14159,r=25.31;
5.      printf("円周 = %f\n",2.0*r*pi);
6.      printf("面積 = %f\n",r*r*pi);
7.  }
```

解答 1-4-4

```
1.  #include <stdio.h>
2.  main()
```

```
3.  {
4.      double x=1.3524986,y=4.1259354,z;
5.      printf("x*x+y*y = %f\n",z);
6.  }
```

注意 1-4-4

4. float 型では桁数不足

解答 1-4-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=12.76,b=2561.2368914;
5.      printf("a = %e\n",a);
6.      printf("b = %e\n",b);
7.  }
```

注意 1-4-5

5,6. %e の出力書式は±x.xxxxxxe±xx

解答 1-4-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float a=3.452,b=5.786,c;
5.      printf("a = %f\n",a);
6.      printf("b = %f\n",b);
7.      printf("c = %e\n",c);
8.  }
```

解答 1-4-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=1.245e+23,b=2.567e-123;
5.      printf("a = %e\n",a);
6.      printf("b = %e\n",b);
7.  }
```

注意 1-4-7

4. b は float 型の範囲外

解答 1-4-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=1.245e+23,b=2.469e+29,c;
5.      printf("%f\n",c);
6.  }
```

注意 1-4-8

4. a*b は float 型の範囲外

1-5) キャスト演算子

解答 1-5-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=20;
5.      float b=15.3114;
6.      printf("%f\n",(float)a+b);
7.  }
```

注意 1-5-1

6. (float):キャスト演算子
   float 型 × int 型は float 型で処理されるので，なくても構わない
   一般にバイト数の多いもので計算される

解答 1-5-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a1,a2;
5.      float a=35.425,b=52.954;
6.      a1=(int)(a+b);
7.      a2=(int)a+(int)b;
8.      printf("a1 = %d\n",a1);
```

```
9.      printf("a2 = %d\n",a2);
10. }
```

注意 1-5-2

6. 計算後に int 型変換を行う
7. 計算前にそれぞれ int 型変換を行う

解答 1-5-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float b=21.234,c=33.125;
5.      printf("%f\n",(double)(b*c));
6.      printf("%f\n",(double)b*(double)c);
7.  }
```

注意 1-5-3

5. 計算後に double 型変換を行う
6. 計算前にそれぞれ double 型変換を行う
   有効桁数の違いに注意すること

解答 1-5-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=3,b=2,c=4;
5.      printf("%f\n",a+(float)b/c);
6.      printf("%f\n",a+((float)b/c));
7.  }
```

注意 1-5-4

5,6. 一般に int 型と float 型の演算は float 型で行われる

解答 1-5-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=3,b=2,c=4;
5.      printf("%f\n",(float)a/b+(float)b/c*a);
6.  }
```

注意 1-5-5

5. *,/,%(余り)は +,- より優先することを利用する

1-6) 文字とアスキーコード

解答 1-6-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a='A',b='B',c='C';
5.      printf("%c %c %c\n",a,b,c);
6.  }
```

注意 1-6-1

5. 'A'は文字 A のアスキーコードを表す
   "A"は文字列となり意味が違ってくる
   %c:アスキーコードに応じた 1 文字を出力する

解答 1-6-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a='F',b='u',c='k',d='u',e='y',f='a',g='m',h='a';
5.      printf("%c%c%c%c%c%c%c%c\n",a,b,c,d,e,f,g,h);
6.  }
```

解答 1-6-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a='A',b='a',c='0';
5.      printf("%c = %d %xH\n",a,a,a);
6.      printf("%c = %d %xH\n",b,b,b);
7.      printf("%c = %d %xH\n",c,c,c);
8.  }
```

注意 1-6-3

4. 'A'は文字 A のアスキーコードを表す

解答 1-6-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a=65,b=98,c=53;
5.      printf("%c %c %c\n",a,b,c);
6.  }
```

注意 1-6-4

4. アスキーコードの 10 進数表示

解答 1-6-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a=0x42,b=0x41,c=0x53,d=0x49,e=0x43;
5.      printf("%c%c%c%c%c\n",a,b,c,d,e);
6.  }
```

注意 1-6-5

4. アスキーコードの 16 進数表示

解答 1-6-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a,b;
5.      a='a'-'A';
6.      b='x'-'X';
7.      printf("a = %x\n",a);
8.      printf("b = %x\n",b);
9.  }
```

注意 1-6-6

5-6. 小文字と大文字のアスキーコードの差は 20_{H}

解答 1-6-7

```
1.  #include <stdio.h>
2.  main()
```

```
3.  {
4.      char c='X';
5.      printf("%c\n",c+0x20);
6.  }
```

注意 1-6-7

5．小文字と大文字のアスキーコードの差が 20_{H} であることを利用する

解答 1-6-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a='b',b='a',c='s',d='i',e='c';
5.      printf("%c%c%c%c%c\n",
6.           a-0x20,b-0x20,c-0x20,d-0x20,e-0x20);
7.  }
```

1-7）書式指定

解答 1-7-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=1,b=10,c=100,d=1000;
5.      printf("%5d\n",a);
6.      printf("%5d\n",b);
7.      printf("%5d\n",c);
8.      printf("%5d\n",d);
9.      printf("%-5d\n",a);
10.     printf("%-5d\n",b);
11.     printf("%-5d\n",c);
12.     printf("%-5d\n",d);
13. }
```

注意 1-7-1

5-8．5 桁右詰めの 10 進整数表示
9-12．5 桁左詰めの 10 進整数表示

解答 1-7-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=19.2345,b=1245.87654;
5.      printf("%10.3f\n",a);
6.      printf("%10.3f\n",b);
7.  }
```

注意 1-7-2

5,6. 全体 10 桁，小数点以下 3 桁の浮動小数点表示

解答 1-7-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=12345.3445;
5.      printf("%.5f\n",a);
6.  }
```

注意 1-7-3

4. float 型では桁数不足
5. 小数点以下 5 桁の浮動小数点表示

解答 1-7-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a=987654321;
5.      printf("%.4e\n",a);
6.  }
```

注意 1-7-4

4. 指数表示をするので double 型にしている
5. 小数点以下 4 桁の指数表示

解答 1-7-5

```
1.  #include <stdio.h>
2.  main()
```

```
3.  {
4.      double a=987654321;
5.      printf("%15.4e\n",a);
6.  }
```

注意 1-7-5

5. 全体 15 桁，小数点以下 4 桁右詰めの指数表示

解答 1-7-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=38,b=5;
5.      printf("%d/%d=%d...%d\n",a,b,a/b,a%b);
6.  }
```

解答 1-7-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a='A';
5.      printf("A の ASCII コードは %xH です\n",a);
6.  }
```

1-8）入力

解答 1-8-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      printf("%d\n",a);
8.  }
```

注意 1-8-1

5. 入力内容がはっきり分かるようにメッセージを表示する
6. 10 進整数を読み込む

scanf()関数の場合読み込み変数はポインタである
a は値であるので & を付けてポインタにする
ポインタを値に変えるには * を付ける
ポインタ: メモリ上で変数の位置するアドレスを表すもの

解答 1-8-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("16 進数 > ");
6.      scanf("%x",&a);
7.      printf("%d %xH\n",a,a);
8.  }
```

注意 1-8-2

6. 16 進整数を読み込む

解答 1-8-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long a;
5.      printf("倍長整数 > ");
6.      scanf("%ld",&a);
7.      printf("%ld\n",a);
8.  }
```

注意 1-8-3

6. long 型 10 進整数を読み込む
   long 型の場合は l を付ける

解答 1-8-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float a;
5.      printf("実数 > ");
6.      scanf("%f",&a);
```

```
7.      printf("%f\n",a);
8.  }
```

注意 1-8-4

6. float 型浮動小数点数の読み込み

解答 1-8-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      double a;
5.      printf("倍精度実数 > ");
6.      scanf("%lf",&a);
7.      printf("%f\n",a);
8.  }
```

注意 1-8-5

6. double 型浮動小数点数の読み込み
   double 型の場合は l を付ける
   ただし，printf()関数の場合 l は付けない

解答 1-8-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a;
5.      printf("文字 > ");
6.      scanf("%c",&a);
7.      printf("%c\n",a);
8.  }
```

注意 1-8-6

6,7: 1 文字を読み込んで表示する
     ただし，リターンキーが押されるまでは表示されない

解答 1-8-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float a,b,c;
```

```
5.      printf("実数 > ");
6.      scanf("%f",&a);
7.      printf("実数 > ");
8.      scanf("%f",&b);
9.      printf("実数 > ");
10.     scanf("%f",&c);
11.     printf("合計 = %f\n",a+b+c);
12. }
```

解答 1-8-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,b;
5.      printf("２つの整数(例:33 6) > ");
6.      scanf("%d%d",&a,&b);
7.      printf("商    = %d\n",a/b);
8.      printf("剰余 = %d\n",a%b);
9.  }
```

注意 1-8-8

6. 空白，タブ，改行を区切りにして入力する

解答 1-8-9

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      float b;
6.      printf("１つの整数と１つの実数(例:3 2.6) > ");
7.      scanf("%d%f",&a,&b);
8.      printf("%f\n",(float)a*b);
9.  }
```

注意 1-8-9

7. 空白，タブ，改行を区切りにして入力する

解答 1-8-10

```
1.  #include <stdio.h>
```

```
2.  main()
3.  {
4.      char a,b,c,d,e;
5.      printf("５つの文字(例:abcde) > ");
6.      scanf("%c%c%c%c%c",&a,&b,&c,&d,&e);
7.      printf("%c%c%c%c%c\n",a,b,c,d,e);
8.  }
```

注意 1-8-10

6. 区切り記号を入れずに入力する
   入れた場合は区切り記号まで読み込んでしまう

解答 1-8-11

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,b,c;
5.      printf("３つの整数( , 区切り) > ");
6.      scanf("%d,%d,%d",&a,&b,&c);
7.      printf("%d\n",a+b+c);
8.      printf("３つの整数( \\ 区切り) > ");
9.      scanf("%d\\%d\\%d",&a,&b,&c);
10.     printf("%d\n",a+b+c);
11. }
```

注意 1-8-11

6. , を区切りにして入力する
   空白, タブ, 改行も区切り記号として有効
9. \ を区切りにして入力する場合はエスケープ文字を利用する

解答 1-8-12

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,b,c;
5.      printf("９桁の連続する数(例:123456789) > ");
6.      scanf("%3d%3d%3d",&a,&b,&c);
7.      printf("%3d %3d %3d\n",a,b,c);
8.  }
```

注意 1-8-12

6．3 桁ずつ区切って読み込む

2章 条件判断

2-1) 条件判断

解答 2-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      if(a>0) printf("プラス\n");
8.  }
```

注意 2-1-1

7. a>0 以外は実行されない
   実行文が１つなので{}は省略出来る

解答 2-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      if(a>0) printf("プラス\n");
8.      else if(a<0) printf("マイナス\n");
9.      else printf("ゼロ\n");
10. }
```

注意 2-1-2

7,8. 実行文が１つなので{}は省略出来る

解答 2-1-3

```
1.  #include <stdio.h>

2.  main()
3.  {
4.      int a;
```

```
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      if(a>=0) printf("%d\n",a);
8.      else printf("%d\n",-a);
9.  }
```

注意 2-1-3

7,8. 絶対値は数値の符号を強制的に + にしたもの，負の数には-1 を掛ける

解答 2-1-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      if(a%2==0) printf("偶数\n");
8.      else printf("奇数\n");
9.  }
```

注意 2-1-4

7. 2 で割った余りが 0 なら偶数，等号は ==
8. その他の場合は奇数

解答 2-1-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.  char a;
5.  printf("文字 > ");
6.  scanf("%c",&a);
7.  if(a=='Y') printf("YES\n");
8.  else if(a=='N') printf("NO\n");
9.  else printf("?\n");
10. }
```

注意 2-1-5

6. リターンキーが押されるまで次が実行されない
7. 等号は ==

解答 2-1-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a;
5.      printf("文字 > ");
6.      scanf("%c",&a);
7.      if(a=='Y'||a=='y') printf("YES\n");
8.      else printf("NO\n");
9.  }
```

注意 2-1-6

6. リターンキーが押されるまで次が実行されない
7. ||: 論理演算子 OR, == が優先

解答 2-1-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("整数 > ");
6.      scanf("%d",&a);
7.      if(a>=0&&a<100) printf("領域内\n");
8.      else printf("領域外\n");
9.  }
```

注意 2-1-7

7. &&: 論理演算子 AND, >=,< が優先

解答 2-1-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      float x;
5.      printf("実数 > ");
6.      scanf("%f",&x);
7.      if(x>=-2.0&&x<=-1.0||x>=1.0&&x<=2.0)
8.          printf("領域内\n");
9.      else printf("領域外\n");
10. }
```

注意 2-1-8

7. || よりも && が優先，不安な場合は()でくくること，例えば
   if((x>=-2.0&&x<=-1.0)||(x>=1.0&&x<=2.0))

解答 2-1-9

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("数字(1-7) > ");
6.      scanf("%d",&a);
7.      switch(a){
8.          case 1 : printf("東京\n"); break;
9.          case 2 : printf("名古屋\n"); break;
10.         case 3 : printf("京都\n"); break;
11.         case 4 : printf("大阪\n"); break;
12.         case 5 : printf("岡山\n"); break;
13.         case 6 : printf("広島\n"); break;
14.         case 7 : printf("博多\n"); break;
15.         default: printf("?\n");
16.     }
17. }
```

注意 2-1-9

8-14. break; がなければ一致した所以下全て実行される
15. どれにも一致しない場合実行される。

解答 2-1-10

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a;
5.      printf("文字 > ");
6.      scanf("%c",&a);
7.      if(a>='a' && a<='z') printf("英小文字\n");
8.      else if(a>='A' && a<='Z') printf("英大文字\n");
9.      else if(a>='0' && a<='9') printf("数字\n");
10.     else printf("その他\n");
11. }
```

注意 2-1-10

6．リターンキーが押されるまで次が実行されない

解答 2-1-11

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("Q1 : ");
6.      scanf("%d",&a);
7.      if(a==1){
8.          printf("Q2 : ");
9.          scanf("%d",&a);
10.         if(a==1) printf("ANS=A");
11.         else if(a==0) printf("ANS=B");
12.         else printf("ERROR");
13.     }
14.     else if(a==0){
15.         printf("Q3 : ");
16.         scanf("%d",&a);
17.         if(a==1){
18.             printf("Q4 : ");
19.             scanf("%d",&a);
20.             if(a==1) printf("ANS=C");
21.             else if(a==0) printf("ANS=D");
22.             else printf("ERROR");
23.         }
24.         else if(a==0) printf("ANS=E");
25.         else printf("ERROR");
26.     }
27.     else printf("ERROR");
28. }
```

3章 繰返し

3-1) for 文

解答 3-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      for(i=1;i<=5;i++) printf("福井正康 ");
6.      printf("\n");
7.  }
```

注意 3-1-1

5. 実行文が 1 つなので {} は省略している
   i++ は i=i+1 の意味である
6. この行はなくてもよいが，この後他の出力がある場合は改行しておく方がよい

解答 3-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      for(i=1;i<=10;i++) printf("%d\n",i);
6.  }
```

注意 3-1-2

5. i++ は i=i+1 の意味である

解答 3-1-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      for(i=1;i<=10;i++) printf("%d ",i);
6.      printf("\n");
7.  }
```

注意 3-1-3

5. %d の後に空白を 1 つ入れる
6. この行はなくてもよいが，この後他の出力がある場合は改行しておく方がよい

解答 3-1-4

```
1. #include <stdio.h>
2. main()
3. {
4.     int i;
5.     for(i=1;i<=10;i++) printf("%4d",i);
6.     printf("\n");
7. }
```

注意 3-1-4

5. %4d: 4 桁右詰め

解答 3-1-5

```
1. #include <stdio.h>
2. main()
3. {
4.     int i;
5.     for(i=10;i>=1;i--) printf("%4d",i);
6.     printf("\n");
7. }
```

注意 3-1-5

5. for(i=0;i<10;i++) printf("%4d",10-i); 等も可能である

解答 3-1-6

```
1. #include <stdio.h>
2. main()
3. {
4.     int i;
5.     for(i=7;i<=77;i=i+7) printf("%4d",i);
6.     printf("\n");
7. }
```

注意 3-1-6

5. i=i+7 は i+=7 とも書ける

解答 3-1-7

```
1. #include <stdio.h>
2. main()
3. {
4.     int i;
5.     for(i=100;i>=40;i=i-20) printf("%d,",i);
6.     printf("%d\n",i);
7. }
```

注意 3-1-7

5. i=i-20 は i-=20 とも書ける
   100～40 は後ろにカンマを入れる
6. 20 は後ろに改行をいれる

解答 3-1-8

```
1. #include <stdio.h>
2. main()
3. {
4.     char i;
5.     for(i=65;i<=90;i++) printf("%c",i);
6. }
```

注意 3-1-8

5. 65,90 はそれぞれ文字 A,Z のアスキーコードである
   for(i='A';i<='Z';i++) ･･･ としてもよい

解答 3-1-9

```
1. #include <stdio.h>
2. main()
3. {
4.     int i,t=0;
5.     for(i=1;i<=100;i++) t=t+i;
6.     printf("%d\n",t);
7. }
```

注意 3-1-9

5. 合計を代入する t は 0 で初期化する
   t=t+i; は t+=i; とも書ける

解答 3-1-10

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,t1=0,t2=0;
5.      for(i=1;i<=10;i++){
6.          t1=t1+i;
7.          t2=t2+i*i;
8.      }
9.      printf("%d %d\n",t1,t2);
10. }
```

注意 3-1-10

4. t1 には合計 t2 には 2 乗の合計を代入する
   t1 と t2 は 0 で初期化する
6. t1+=i; とも書ける
7. t2+=i*i; とも書ける

解答 3-1-11

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,t1=0,t2=0;
5.      for(i=1;i<=99;i=i+2) t1=t1+i;
6.      for(i=2;i<=100;i=i+2) t2=t2+i;
7.      printf("奇数合計 = %d\n",t1);
8.      printf("偶数合計 = %d\n",t2);
9.  }
```

注意 3-1-11

4. t1 には奇数合計, t2 には偶数合計を代入する
5. i は奇数だけを動く
6. i は偶数だけを動く

解答 3-1-12

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long i,fact=1;
5.      for(i=1;i<=10;i++){
6.          fact=fact*i;
7.          printf("%ld ",fact);
8.      }
9.      printf("\n");
10. }
```

注意 3-1-12

4. 階乗の計算は桁数が大きくなるので long 型を使う
   もっと大きくなる場合は double 型にする
6. i だけ整数にしてキャスト演算子を用いてもよい
7. long 型 10 進数表示

解答 3-1-13

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      float x=0.0;
6.      for(i=1;i<=10;i++) x=x+(float)i/(i+1);
7.      printf("%f\n",x);
8.  }
```

注意 3-1-13

4. 始めから float 型で計算してもよい
5. 精度の必要な場合は double 型にする
6. 実数計算になるので float 型にキャストする
   (float)を忘れると x は 0 のままである

解答 3-1-14

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long i,t=0;
5.      for(i=1;i<=28;i=i+3) t=t+i*(i+1)*(i+2);
```

```
6.      printf("%ld\n",t);
7.  }
```

注意 13-1-4

4. int 型では桁数が足りない

解答 3-1-15

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      long a=1,s=1;
6.      for(i=2;i<=10;i++){
7.          a=5*a-1;
8.          s=s+a;
9.      }
10.     printf("a10 の値      = %ld\n",a);
11.     printf("a10 までの和 = %ld\n",s);
12. }
```

注意 3-1-15

5. int 型では桁数が足りない
   long a=s=1; という書式も可能である
6. for(i=0;i<9;i++){ 等でもよい
   9 回実行することが重要である
7. $a_n=5a_{n-1}-1$ を計算している
8. s+=a; でもよい

解答 3-1-16

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,j;
5.      for(i=1;i<=10;i++){
6.          for(j=1;j<=5;j++) printf("きよみ ");
7.          printf("\n");
8.      }
9.  }
```

注意 3-1-16

5. for(i=0;i<10;i++){ 等でもよい
6. for(j=0;j<5;j++) 等でもよい
7. 5 回連続して出力したら改行する

解答 3-1-17

```
1. #include <stdio.h>
2. main()
3. {
4.     int i,j;
5.     for(i=0;i<=9;i++){
6.         for(j=1;j<=10;j++) printf("%4d",i*10+j);
7.         printf("\n");
8.     }
9. }
```

注意 3-1-17

5,6. for(i=0;i<=90;i=i+10) として
printf("%4d",i+j); とする等, 様々な書式が可能である
7. 10 回出力したら改行する

解答 3-1-18

```
1. #include <stdio.h>
2. main()
3. {
4.     int i,j,t=0;
5.     for(i=1;i<=9;i++){
6.         for(j=1;j<=9;j++) t=t+i*j;
7.     }
8.     printf("%d\n",t);
9. }
```

注意 3-1-18

6. t+=i*j; でもよい

3-2) while 文

解答 3-2-1

```
1. #include <stdio.h>
```

```
2.  main()
3.  {
4.      int a;
5.      while(1){
6.          printf("data > ");
7.          scanf("%d",&a);
8.          if(a>0) printf("%d\n",a);
9.          else if(a<0) printf("%d\n",-a);
10.         else break;
11.     }
12. }
```

注意 3-2-1

5. 1 は真であり，無限ループを表す
8,9. 絶対値を表示している
10. 0 が入力されたらループから抜け出す
    最後に入力する 0 は表示されない
    最初に scanf()関数を用い，while(a!=0)としてループの最後で scanf()関数を用いる方法もある
    ここでは scanf()関数が少なくて済む break 文を用いた

解答 3-2-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,t=0;
5.      while(1){
6.          printf("data > ");
7.          scanf("%d",&a);
8.          if(a==-999) break;
9.          t=t+a;
10.     }
11.     printf("合計 = %d\n",t);
12. }
```

注意 3-2-2

5. 1 は真であり，無限ループを表す
8. -999 が入力されたらループから抜け出す
   最後に入力する-999 は加算されない
9. t+=a; でもよい
   最初に scanf()関数を用い，while(a!=0)としてループの最後で scanf()

関数を用いる方法もある
ここでは scanf()関数が少なくて済む break 文を用いた

解答 3-2-3

```
 1. #include <stdio.h>
 2. main()
 3. {
 4.     int n=0;
 5.     float x,t1=0.0,t2=0.0,mean,var;
 6.     while(1){
 7.         printf("data > ");
 8.         scanf("%f",&x);
 9.         if(x==-999.0) break;
10.         t1=t1+x;
11.         t2=t2+x*x;
12.         n++;
13.     }
14.     mean=t1/(float)n;
15.     var=t2/(float)n-mean*mean;
16.     printf("平均 = %f\n",mean);
17.     printf("分散 = %f\n",var);
18. }
```

注意 3-2-3

5. 平均は mean，分散は var で表す
9. -999 が入力されたらループから抜け出す
10,11. 平均と分散を求めるには，合計と 2 乗の合計が必要
12. n はデータ数を表すカウンタ
14. n は int 型なので float 型へキャストしている
この場合(float)はなくても float 型で計算する
15. 分散の公式を利用している

解答 3-2-4

```
 1. #include <stdio.h>
 2. main()
 3. {
 4.     int n=0,total=0;
 5.     while(total<500){
 6.         n++;
 7.         total=total+n;
```

```
8.      }
9.      printf("最大 = %d\n",n-1);
10.     printf("和 = %d\n",total-n);
11. }
```

注意 3-2-4

6,7. total+=++n; とまとめることも出来る
9. ループが終った時点で 1 増えているので 1 を引く
10. ループが終った時点で n 増えているので n を引く

解答 3-2-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int p,n;
5.      printf("data > ");
6.      scanf("%d",&p);
7.      n=p/2;
8.      while(p%n!=0) n--;
9.      printf("%d 以外の最大約数 = %d\n",p,n);
10. }
```

注意 3-2-5

7. 自分自身以外の約数で p/2 より大きいものはない
8. n を 1 ずつ減らして，割り切れるかどうか確かめる
   もっと速い方法も考えられようがこれが最も簡単である

４章 配列とポインタ

4-1) 配列

解答 4-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,a[10]={0,1,2,3,4,5,6,7,8,9};
5.      for(i=0;i<10;i++) printf("%d ",a[i]);
6.      printf("\n");
7.  }
```

注意 4-1-1

4. 配列の値の初期化
   a[10]の 10 は省略出来る
5. 先頭は a[0]である

解答 4-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,a[10];
5.      for(i=0;i<10;i++) a[i]=i;
6.      for(i=0;i<10;i++) printf("%d ",a[i]);
7.  }
```

注意 4-1-2

5. 配列の値の代入

解答 4-1-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,c[5],
5.          a[5]={1,2,3,4,5},b[5]={3,4,5,6,7};
6.      for(i=0;i<5;i++) c[i]=a[i]+b[i];
7.      for(i=0;i<5;i++) printf("%d ",c[i]);
8.      printf("\n");
```

```
9.  }
```

注意 4-1-3

5. 配列の値の初期化
   a[5],b[5]の 5 は省略出来る

解答 4-1-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,total=0,
5.          a[8]={3,9,1,4,3,7,8,5};
6.      for(i=0;i<8;i++) total=total+a[i];
7.      printf("合計 = %d\n",total);
8.  }
```

注意 4-1-4

4. total に合計が代入される
5. a[8]の 8 は省略出来る

解答 4-1-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,max,min,
5.          a[16]={5,4,6,8,2,3,5,1,3,6,2,4,8,9,9,7};
6.      max=min=a[0];
7.      for(i=1;i<16;i++){
8.          if(a[i]>max) max=a[i];
9.          if(a[i]<min) min=a[i];
10.     }
11.     printf("max = %d\n",max);
12.     printf("min = %d\n",min);
13. }
```

注意 4-1-5

5. a[16]の 16 は省略出来る
6. max=a[0]; min=a[0]; と分けて書いてもよい
   最大，最小を求める場合，初期値は最初のデータの値とする
7. 最大値の抽出

9. 最小値の抽出

解答 4-1-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      float a[5]={2.0,4.0,6.0,8.0,10.0};
6.      for(i=0;i<5;i++) printf("%6.1f",a[i]);
7.      printf("\n");
8.      for(i=0;i<5;i++) printf("%6.1f",*(a+i));
9.      printf("\n");
10.  }
```

注意 4-1-6

5. a[5]の 5 は省略出来る
6. 配列の要素としての表し方
8. ポインタの示す場所のデータとしての表し方

解答 4-1-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,j,
5.          a[4][3]={{34,17,44},
6.                   {24,59,47},
7.                   {91,11,61},
8.                   {89,22,65}};
9.      for(i=0;i<4;i++){
10.         for(j=0;j<3;j++) printf("%4d",a[i][j]);
11.         printf("\n");
12.     }
13. }
```

注意 4-1-7

5. a[4][3]の 4 は省略出来るが 3 は出来ない

5-8. 2 次元配列の初期化
メモリ上でのデータの順番は a[0][0],a[0][1],a[0][2],a[1][0],
a[1][1],･･･,a[3][2]である

10. 行列のように i 行 j 列として表示する

11．3つ横に表示したら改行する

解答 4-1-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,j,a[4][3];
5.      a[0][0]=34; a[0][1]=17; a[0][2]=44;
6.      a[1][0]=24; a[1][1]=59; a[1][2]=47;
7.      a[2][0]=91; a[2][1]=11; a[2][2]=61;
8.      a[3][0]=89; a[3][1]=22; a[3][2]=65;
9.      for(i=0;i<3;i++){
10.         for(j=0;j<4;j++) printf("%4d",a[j][i]);
11.         printf("\n");
12.     }
13. }
```

注意 4-1-8

5-8．2次元配列への値の代入
10．前問では縦に並んだ数字をi,jを入れ換えることにより横に並べている
11．4つ横に表示したら改行する

解答 4-1-9

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,j,c[3][2],
5.          a[3][2]={{32,55},{48,73},{63,35}},
6.          b[3][2]={{24,23},{64,79},{98,18}};
7.      for(i=0;i<3;i++)
8.          for(j=0;j<2;j++) c[i][j]=a[i][j]+b[i][j];
9.      for(i=0;i<3;i++){
10.         for(j=0;j<2;j++) printf("%4d",c[i][j]);
11.         printf("\n");
12.     }
13. }
```

注意 4-1-9

5.6 2次元配列の初期化
7.　実行文が1つなので{}を省略している

8. 合計の計算

解答 4-1-10

```
1. #include <stdio.h>
2. main()
3. {
4.     int x,a[2][2]={{3,5},{2,8}};
5.     x=a[0][0]*a[1][1]-a[0][1]*a[1][0];
6.     printf("行列式 = %d\n",x);
7. }
```

注意 4-1-10

5. 行列式の計算

解答 4-1-11

```
1. #include <stdio.h>
2. main()
3. {
4.     int i,j,det,a[2][2]={{3,5},{2,8}};
5.     float b[2][2];
6.     det=a[0][0]*a[1][1]-a[0][1]*a[1][0];
7.     b[0][0]=(float)a[1][1]/det;
8.     b[0][1]=-(float)a[0][1]/det;
9.     b[1][0]=-(float)a[1][0]/det;
10.    b[1][1]=(float)a[0][0]/det;
11.    for(i=0;i<2;i++){
12.        for(j=0;j<2;j++) printf("%6.2f",b[i][j]);
13.        printf("\n");
14.    }
15. }
```

注意 4-1-11

6. 行列式の計算
7-10. 逆行列の各成分の計算
　　配列 a が int 型であるので float 型にキャストしている
11-14. 行列の形で表示している

解答 4-1-12

```
1. #include <stdio.h>
```

```
2.  main()
3.  {
4.      int i;
5.      char a[7];
6.      a[0]='P'; a[1]='R'; a[2]='O'; a[3]='G';
7.      a[4]='R'; a[5]='A'; a[6]='M';
8.      for(i=0;i<7;i++) printf("%c",a[i]);
9.      printf("\n");
10. }
```

注意 4-1-12

5. 半角文字は 1 バイトなので char 型でよい
   半角カナを利用する場合は負の値になるので注意する

6,7. 各文字のアスキーコードを代入する
   例えば'p'は文字 p のアスキーコードを表す

解答 4-1-13

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      char a[6]={'P','R','O','L','O','G'};
6.      for(i=0;i<6;i++) printf("%c",a[i]);
7.      printf("\n");
8.  }
```

注意 4-1-13

5. 配列 a[6]を各文字のアスキーコードで初期化する
6. 6 文字分出力する

解答 4-1-14

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a[7]={'P','R','O','L','O','G','\0'};
5.      printf("%s\n",a);
6.  }
```

注意 4-1-14

4. 配列 a[]を各文字のアスキーコードで初期化する

文字列として%s 指定で出力する場合は文字列の最後に NUL(0)を付ける
配列の大きさは前問に比べて 1 大きい 7 になっている
0,0x0,'\0'はどれもみな同じ NUL(0)である

5. NUL(0)コードがあるまで文字を出力する

解答 4-1-15

```
1. #include <stdio.h>
2. main()
3. {
4.     int i;
5.     char a[6]="BASIC";
6.     printf("%s\n",a);
7.     for(i=0;i<6;i++) printf("%c ",a[i]);
8.     printf("\n");
9. }
```

注意 4-1-15

5. 配列の文字列による初期化
   配列の要素数は初期化する文字列の文字数よりも 1 以上大きくする
   a[5]に NUL(0)が自動的に入る
6. 文字列として表示する
7. 文字として間に 1 つずつ空白を入れて表示する

解答 4-1-16

```
1. #include <stdio.h>
2. main()
3. {
4.     int n=0;
5.     char a[30];
6.     printf("文字列 > ");
7.     scanf("%s",a);
8.     while(a[n]!=0) n++;
9.     printf("字数 = %d\n",n);
10. }
```

注意 4-1-16

4. n を文字数を求めるカウンタとしている
7. 6 行の宣言から，読み込める文字数は 29 文字までである
8. NUL(0)に出会うまでカウントするが，最後の NUL(0)はカウントしない

解答 4-1-17

```
 1.  #include <stdio.h>
 2.  main()
 3.  {
 4.      char a[3][10];
 5.      int i;
 6.      a[0][0]='T'; a[0][1]='o'; a[0][2]='k';
 7.      a[0][3]='y'; a[0][4]='o'; a[0][5]=0x0;
 8.      a[1][0]='H'; a[1][1]='i'; a[1][2]='r'; a[1][3]='o';
 9.      a[1][4]='s'; a[1][5]='h'; a[1][6]='i'; a[1][7]='m';
10.     a[1][8]='a'; a[1][9]=0x0;
11.     a[2][0]='O'; a[2][1]='k'; a[2][2]='a'; a[2][3]='y';
12.     a[2][4]='a'; a[2][5]='m'; a[2][6]='a'; a[2][7]=0x0;
13.     for(i=0;i<3;i++) printf("%s\n",a[i]);
14. }
```

注意 4-1-17

4. a[3][10]の 10 は最長の文字数よりも 1 以上大きくする
6-12. 2 次元配列に文字列を代入する
      文字列の最後は NUL(0)である
13. 各文字列の先頭ポインタ a[i]を利用して表示する

解答 4-1-18

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      char a[3][7]={"Tokyo","Nagoya","Osaka"};
6.      for(i=0;i<3;i++) printf("%s\n",a[i]);
7.  }
```

注意 4-1-18

5. 2 次元配列を文字列で初期化する
    文字列の最後には自動的に NUL(0)が付く
    a[3][7]の 7 は最長の文字数よりも 1 以上大きくする
    a[3][7]の 3 は省略出来るが 7 は出来ない

解答 4-1-19

```
1.  #include <stdio.h>
2.  main()
```

```
3.  {
4.      int i;
5.      char *a[4];
6.      a[0]="BASIC";
7.      a[1]="PASCAL";
8.      a[2]="C";
9.      a[3]="LISP";
10.     for(i=0;i<4;i++) printf("%s\n",a[i]);
11. }
```

注意 4-1-19

5. char 型のポインタ配列の宣言
6-9. それぞれのポインタ配列に各文字列の先頭ポインタを代入する
10. ポインタ配列の各要素を利用して表示する

解答 4-1-20

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      char *a[]={"BASIC","PASCAL","C","LISP"};
6.      for(i=0;i<=3;i++) printf("%s\n",a[i]);
7.  }
```

注意 4-1-20

5. ポインタ配列の初期化
   要素数は自動的に 4 になる
   ポインタ配列でも各文字に対して a[i][j]のような指定法が出来る

4-2) ポインタ

解答 4-2-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("%p\n",&a);
6.  }
```

注意 4-2-1

5. %p はポインタの表示書式である

%p の p は小文字を用いる

解答 4-2-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,*p;
5.      a=7;
6.      p=&a;
7.      printf("%d\n",*p);
8.  }
```

注意 4-2-2

4. ポインタ変数 p の宣言
6. p に a のポインタを代入する
7. p の指すデータ(a の値)を表示する

解答 4-2-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a,*p;
5.      p=&a;
6.      *p=8;
7.      printf("%d\n",a);
8.  }
```

注意 4-2-3

4. ポインタ変数 p の宣言
5. p に a のポインタを代入する
6. p の指す位置に値を代入する
7. 代入した値が a の値となっている

解答 4-2-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char *a,*p,*q;
5.      p="北海道";
```

```
6.      q="沖縄";
7.      printf("%s %s\n",p,q);
8.      a=p;
9.      p=q;
10.     q=a;
11.     printf("%s %s\n",p,q);
12. }
```

注意 4-2-4

8-10. ポインタを交換する
11. "北海道"と"沖縄"が順序を入れ換えて表示される

解答 4-2-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,a[5]={6,3,4,2,1};
5.      for(i=0;i<5;i++) printf("%d ",*(a+i));
6.      printf("\n");
7.  }
```

注意 4-2-5

5. a[i]=*(a+i)である

解答 4-2-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char str[5]={'T','o','k','y','o'};
5.      int i;
6.      for(i=0;i<5;i++) printf("%c",str[i]);
7.      printf("\n");
8.      for(i=0;i<5;i++) printf("%c",*(str+i));
9.      printf("\n");
10. }
```

注意 4-2-6

6,8. str[i]=*(str+i)である

解答 4-2-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char str[5]={'T','o','k','y','o'};
5.      int i;
6.      for(i=0;i<5;i++) printf("%p ",str+i);
7.      printf("\n");
8.      for(i=0;i<5;i++) printf("%p ",&str[i]);
9.      printf("\n");
10. }
```

注意 4-2-7

6,8. %p はポインタの表示書式である
%p の p は小文字を用いる
str+i=&str[i]である

解答 4-2-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char c[10];
5.      int a[10];
6.      float x[10];
7.      printf("%p %p\n",c,c+1);
8.      printf("%p %p\n",a,a+1);
9.      printf("%p %p\n",x,x+1);
10. }
```

注意 4-2-8

7. char 型配列の 1 要素は 1 バイトである
8. int 型配列の 1 要素は 2 バイトである
9. float 型配列の 1 要素は 4 バイトである

解答 4-2-9

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char a[13]="東京にいます";
```

```
5.      printf("%s\n",a);
6.      printf("%s\n",a+6);
7.  }
```

注意 4-2-9

5. 全体を表示する
6. "います"の部分だけを表示する
   この文字列の先頭ポインタは a+6 である

解答 4-2-10

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char c[20];
5.      c[0]='T'; c[1]='o'; c[2]='k'; c[3]='y';
6.      c[4]='o'; c[5]=0x0;
7.      c[10]='K'; c[11]='y'; c[12]='o'; c[13]='t';
8.      c[14]='o'; c[15]=0x0;
9.      printf("%s\n",c);
10.     printf("%s\n",c+10);
11. }
```

注意 4-2-10

7. 10 バイト目から"Kyoto"を代入している
9. 先頭から NUL(0)まで表示する
10. 10 バイト目から NUL(0)まで表示する

解答 4-2-11

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,j;
5.      int a[2][3]={{1,2,3},{7,8,9}};
6.      for(i=0;i<2;i++){
7.          for(j=0;j<3;j++) printf("%p ",&a[i][j]);
8.          printf("\n");
9.      }
10. }
```

注意 4-2-11

7. a[i]+j を用いてもよい

解答 4-2-12

```
1. #include <stdio.h>
2. main()
3. {
4.     int i,j;
5.     int a[2][3]={{1,2,3},{7,8,9}};
6.     for(i=0;i<=1;i++){
7.         for(j=0;j<=2;j++) printf("%3d",*(*(a+i)+j));
8.         printf("\n");
9.     }
10. }
```

注意 4-2-12

7. a[i][j]=*(a[i]+j)=*(*(a+i)+j)である

解答 4-2-13

```
1. #include <stdio.h>
2. main()
3. {
4.     char *a[2];
5.     int i,j;
6.     a[0]="Osaka";
7.     a[1]="Hiroshima";
8.     for(i=0;i<2;i++){
9.         j=0;
10.        while(a[i][j]!=0){
11.            printf("%c",a[i][j]);
12.            j++;
13.        }
14.        printf("\n");
15.    }
16. }
```

注意 4-2-13

9-13. 各 i について j=0 から a[i][j]が NUL(0)になるまで文字を表示する
10. *(a[i]+j)も可能である

5章 関数

5-1) 関数 1

解答 5-1-1

```
1.  #include <stdio.h>
2.  void cls(void);
3.  main()
4.  {
5.      cls();
6.  }
7.  void cls(void)
8.  {
9.      printf("\x1b[2J");
10. }
```

注意 5-1-1

2. 関数のプロトタイプ宣言
5. 関数を呼び出す
7. 戻値なし 引数なし
9. <ESC>[2J はエスケープシーケンスの画面消去である

解答 5-1-2

```
1.  #include <stdio.h>
2.  int spc(int);
3.  main()
4.  {
5.      int i;
6.      for(i=0;i<5;i++){
7.          printf("A");
8.          spc(i);
9.          printf("B\n");
10.     }
11. }
12. int spc(int n)
13. {
14.     int i;
15.     if(n<=0) return 0;
16.     for(i=0;i<n;i++) printf(" ");
```

```
17.     return 1;
18. }
```

注意 5-1-2

2. 関数のプロトタイプ宣言
7-9.うまく空白が出力されているか確かめている
12. n は出力する空白の数である
15. 0 以下の値が与えられると 0 を返す
16. 指定した数だけ空白を出力する
17. 戻値を 1 にして関数を終了する

解答 5-1-3

```
1.  #include <stdio.h>
2.  double dabs(double);
3.  main()
4.  {
5.      double x;
6.      printf("数値 > ");
7.      scanf("%lf",&x);
8.      printf("絶対値 = %f\n",dabs(x));
9.  }
10. double dabs(double x)
11. {
12.     if(x>=0.0) return x;
13.     else return -x;
14. }
```

注意 5-1-3

2. 関数のプロトタイプ宣言
7. double 型の浮動小数点入力書式は%lf を用いる
8. 入力した数の絶対値を表示する関数を呼び出す
12. 0 以上の場合はそのまま戻値とする
13. その他の場合は-1 を掛ける

解答 5-1-4

```
1.  #include <stdio.h>
2.  double ipow(double,int);
3.  main()
4.  {
5.      printf("2^3 = %f\n",ipow(2.0,3));
```

```
6.  }
7.  double ipow(double x,int n)
8.  {
9.      int i;
10.     double ans=1.0;
11.     for(i=0;i<n;i++) ans=ans*x;
12.     return ans;
13. }
```

注意 5-1-4

5. 試みに 2^3 を計算させてみる
   値を入力するようにした方がよいが，見づらいので固定した
11. x^n を計算する
12. x^n を戻値にするが，n<0 の場合は考えていない

解答 5-1-5

```
1.  #include <stdio.h>
2.  int dispvec(int *,int);
3.  main()
4.  {
5.      int i,n=10,c[10],chk;
6.      for(i=0;i<n;i++) c[i]=i+1;
7.      chk=dispvec(c,n);
8.      printf("戻値 = %d\n",chk);
9.  }
10. int dispvec(int *c,int n)
11. {
12.     int i;
13.     if(n<=0) return 0;
14.     for(i=0;i<n;i++) printf("%d\n",c[i]);
15.     return 1;
16. }
```

注意 5-1-5

10. 配列を先頭ポインタで指定する

解答 5-1-6

```
1.  #include <stdio.h>
2.  int set0(int *,int);
3.  main()
```

```
4.  {
5.      int i,n=10,b[10];
6.      for(i=0;i<n;i++) b[i]=1;
7.      set0(b,n);
8.      for(i=0;i<n;i++) printf("%d ",b[i]);
9.      printf("\n");
10. }
11. int set0(int *b,int n)
12. {
13.     int i;
14.     if(n<=0) return 0;
15.     for(i=0;i<n;i++) b[i]=0;
16.     return 1;
17. }
```

注意 5-1-6

6. 予め他の数を代入しておくと確認がより明瞭になる
8. 0 になっているか確認する
11. 配列をポインタで指定する
15. 配列の要素に 0 を代入

解答 5-1-7

```
1.  #include <stdio.h>
2.  void setw(int *b,int,int);
3.  main()
4.  {
5.      int mat[5][6],j,k,m=5,n=6;
6.      for(j=0;j<m;j++){
7.          for(k=0;k<n;k++) mat[j][k]=1;
8.      }
9.      setw(*mat,m,n);
10.     for(j=0;j<m;j++){
11.         for(k=0;k<n;k++) printf("%2d",mat[j][k]);
12.         printf("\n");
13.     }
14. }
15. void setw(int *b,int m,int n)
16. {
17.     int j,k;
18.     for(j=0;j<m;j++){
```

```
19.          for(k=0;k<n;k++) b[j*n+k]=0;
20.      }
21. }
```

注意 5-1-7

9. 2 次元配列の先頭ポインタ*mat=mat[0]を渡す
10-13. 0 が代入されたことを確かめる
15. 2 次元配列の先頭ポインタを受け取る
19. 配列 mat[m][n]の j 行 k 列は先頭から j*n+k 番目の要素である

解答 5-1-8

```
1.  #include <stdio.h>
2.  int strlength(char *);
3.  main()
4.  {
5.      int len;
6.      char str[]="FUKUI";
7.      len=strlength(str);
8.      printf("%s の文字数 = %d\n",str,len);
9.  }
10. int strlength(char *str)
11. {
12.     int n=0;
13.     while(*str!=0x0){
14.         n++;
15.         str++;
16.     }
17.     return n;
18. }
```

注意 5-1-8

6. 文字型配列を文字列で初期化する
7. 文字数を取得する
13. NUL(0)に出会うまで繰り返す
13-16. while(*str++!=0x0) n++; とも出来る

解答 5-1-9

```
1.  #include <stdio.h>
2.  void dabs2(double,double *)
3.  main()
```

```
4.  {
5.      double x,ret;
6.      printf("data > ");
7.      scanf("%lf",&x);
8.      dabs2(x,&ret);
9.      printf("絶対値 = %f\n",ret);
10. }
11. void dabs2(double x,double *ret)
12. {
13.     if(x>=0) *ret=x;
14.     else *ret=-x;
15. }
```

注意 5-1-9

8. 結果を格納するためのポインタ&ret を渡す
11. 戻値がないので void 型である
    結果を格納するためのポインタを引数として受け取る
13-14. ポインタ ret の位置へ結果を返す

解答 5-1-10

```
1.  #include <stdio.h>
2.  int strcopy(char *,char *);
3.  main()
4.  {
5.      int len;
6.      char str[30],ret[30];
7.      printf("文字列 > ");
8.      scanf("%s",str);
9.      len=strcopy(str,ret);
10.     printf("%s %d\n",ret,len);
11. }
12. int strcopy(char *str,char *ret)
13. {
14.     int n=0;
15.     while(*str!=0x0){
16.         *ret++=*str++;
17.         n++;
18.     }
19.     *ret=0x0;
20.     return n;
```

```
21. }
```

注意 5-1-10

9. str に読み込んだ文字列を ret に複写し，文字数を len に代入する
12. 複写元と複写先文字列の先頭ポインタを引数とする
16. *ret=*str; ret++; str++; と分けてもよい
19. 最後の NUL(0)はコピーされないので付けておく
15-20. do{ *ret++=*str++; n++; }while(*str!=0x0);
return n-1; としてもよい

解答 5-1-11

```
1.  #include <stdio.h>
2.  int strsl(char *,char *);
3.  main()
4.  {
5.      int n;
6.      char str[30],ret[30];
7.      printf("文字列 > ");
8.      scanf("%s",str);
9.      n=strsl(str,ret);
10.     printf("%s %d\n",ret,n);
11. }
12. int strsl(char *str,char *ret)
13. {
14.     int n=0;
15.     while(*str!=0x0){
16.         if(*str>=0x61&&*str<=0x7a){
17.             *ret=*str-0x20;
18.             n++;
19.         }
20.         else *ret=*str;
21.         str++;
22.         ret++;
23.     }
24.     *ret=0x0;
25.     return n;
26. }
```

注意 5-1-11

10. 変換されたかどうか確かめる
12. 変換前と変換後の文字列の先頭ポインタを引数とする

14. n には変換した文字数を代入する
16. 英小文字のアスキーコードの範囲
17. 英小文字と英大文字のアスキーコードの差は 20_H である
24. 最後の NUL(0)はコピーされないので付けておく

5-2) 関数 2

解答 5-2-1

```
1.  #include <stdio.h>
2.  int dist(double,double,double);
3.  main()
4.  {
5.      int ans;
6.      double x,y,z;
7.      printf("x > ");
8.      scanf("%lf",&x);
9.      printf("y > ");
10.     scanf("%lf",&y);
11.     printf("距離 > ");
12.     scanf("%lf",&z);
13.     ans=dist(x,y,z);
14.     if(ans==1) printf("以上\n");
15.     else printf("未満\n");
16. }
17. int dist(double x,double y,double z)
18. {
19.     double p;
20.     p=x*x+y*y;
21.     if(p>=z*z) return 1;
22.     else return 0;
23. }
```

注意 5-2-1

17. 座標(x,y)と比較する距離 z を引数とする
20. 原点からの距離の 2 乗を求める
21. その値と z^2 の値を比較している

解答 5-2-2

```
1.  #include <stdio.h>
2.  float ohgi(float,float);
```

```
3.  main()
4.  {
5.      float r,deg,s;
6.      printf("半径 > ");
7.      scanf("%f",&r);
8.      printf("角度(°) > ");
9.      scanf("%f",&deg);
10.     s=ohgi(r,deg);
11.     printf("面積 = %f\n",s);
12. }
13. float ohgi(float r,float deg)
14. {
15.     float pi=3.14159,s;
16.     s=r*r*pi*deg/360.0;
17.     return s;
18. }
```

注意 5-2-2

13. 半径と角度(°)を引数として与える
15. 精度の必要な場合は double 型にする
16. 360°で円の面積となる
17. 面積を戻値としている

解答 5-2-3

```
1.  #include <stdio.h>
2.  double fact(int);
3.  main()
4.  {
5.      int n;
6.      double x;
7.      printf("整数 > ");
8.      scanf("%d",&n);
9.      x=fact(n);
10.     printf("%d! = %.0f\n",n,x);
11. }
12. double fact(int n)
13. {
14.     int i;
15.     double t=1.0;
16.     for(i=1;i<=n;i++) t=t*(double)i;
```

```
17.     return t;
18. }
```

注意 5-2-3

8-10. n の値を読み込んで確かめる
9. n!の値の計算
   n は int 型，結果 t は double 型であるのでキャストしている
12. n!の値を戻値としている
   戻値は非常に大きくなる可能性があるので double 型にしている

解答 5-2-4

```
1.  #include <stdio.h>
2.  void dispn(char **,int);
3.  main()
4.  {
5.      char *s[]={"TOKYO","NAGOYA","OSAKA"};
6.      dispn(s,3);
7.  }
8.  void dispn(char *str[],int n)
9.  {
10.     int i;
11.     for(i=0;i<n;i++) printf("%s\n",str[i]);
12. }
```

注意 5-2-4

2. ポインタ配列は char *str[]; で宣言されるので，プロトタイプ宣言では[]を省いて char **str とする
5. ポインタ配列を初期化する
8. ポインタ配列の先頭ポインタとその個数を引数にしている
11. ポインタ配列の要素を用いて文字列を表示する

解答 5-2-5

```
1.  #include <stdio.h>
2.  void maxmin(int *,int,int *,int *);
3.  main()
4.  {
5.      int n=5,a[5]={9,6,4,1,3};
6.      int max, min;
7.      maxmin(a,n,&max,&min);
8.      printf("min=%d\n",min);
```

```
9.      printf("max=%d\n",max);
10. }
11. void maxmin(int *a,int n,int *max,int *min)
12. {
13.     int i;
14.     *max=*min=a[0];
15.     for(i=1;i<n;i++){
16.         if(a[i]>*max) *max=a[i];
17.         if(a[i]<*min) *min=a[i];
18.     }
19. }
```

注意 5-2-5

7. 配列 a[5]の 5 つの要素内の最大値と最小値を求める
11. 配列の先頭ポインタ，配列の要素数，最大値と最小値を格納するポインタを引数とする
14. 最初の値をそれぞれの初期値とする
    *max=a[0]; *min=a[0]; と分けてもよい
16. 最大値の抽出
17. 最小値の抽出

解答 5-2-6

```
1.  #include <stdio.h>
2.  char strcatin(char *,char *,char *);
3.  main()
4.  {
5.      char c[30],*a,*b;
6.      a="Satoh";
7.      b="Kiyomi";
8.      strcatin(a,b,c);
9.      printf("%s\n",c);
10. }
11. char strcatin(char *a,char *b,char *c)
12. {
13.     int i=0,j=0;
14.     while(a[i]!=0x0){
15.         c[i]=a[i];
16.         i++;
17.     }
18.     while(b[j]!=0x0){
```

```
19.         c[i]=b[j];
20.         i++;
21.         j++;
22.     }
23.     c[i]=0x0;
24. }
```

注意 5-2-6

11. 標準ライブラリ関数には strcat()関数があるが，仕様が異なる
    先頭ポインタ a,b で表される文字列を連結してポインタ c で表される
    位置に代入する
14-17. 先頭ポインタ a で表される文字列を c の位置に代入する
    while(a[i]!=0x0) c[i]=a[i++]; とすることも出来る
18-23. 先頭ポインタ b で表される文字列を c の続きの位置に代入する
18-22. while(b[i]!=0x0) c[i++]=b[j++]; とすることも出来る
23. 最後の NUL(0)はコピーされないので付けておく

解答 5-2-7

```
1.  #include <stdio.h>
2.  char *charfind(char *,char);
3.  main()
4.  {
5.      char *p,a[30],c;
6.      printf("文字列 > ");
7.      scanf("%s",a);
8.      getchar();
9.      printf("検索文字 > ");
10.     scanf("%c",&c);
11.     p=charfind(a,c);
12.     if(p!=NULL) printf("%s\n",p);
13.     else printf("見つかりません。\n");
14. }
15. char *charfind(char *a,char c)
16. {
17.     int n;
18.     while(*a!=NULL){
19.         if(*a==c) return a;
20.         a++;
21.     }
22.     return NULL;
```

```
23. }
```

注意 5-2-7

7. 8.がなければ LF(0xa)が次の%c に読み込まれる
15. 標準ライブラリ関数には文字列中から指定文字列を検索する strstr() 関数がある
    検索される文字列の先頭ポインタと検索文字を引数とする
18. 文字列の最後に達すれば終了する
19. 見つければそのポインタを返す
22. 見つからなければ NULL ポインタを返す

5-3) main 関数の引数

解答 5-3-1

```
1.  #include <stdio.h>
2.  main(int argc,char *argv[]){
3.      int i;
4.      printf("argc = %d\n",argc);
5.      for(i=0;i<argc;i++)
6.          printf("argv[%d] = %s\n",i,argv[i]);
7.  }
```

注意 5-3-1

2. argc: 引数の個数+1(コマンド自身)
   argv[0]: 起動プログラム名の文字列へのポインタ
   argv[1]: 第 1 引数文字列へのポインタ
   ・・・
   argv[n]: 第 n 引数文字列へのポインタ
6. 引数の表示

解答 5-3-2

```
1.  #include <stdio.h>
2.  main(int argc,char *argv[])
3.  {
4.      int i;
5.      if(argc==3){
6.          printf("%s\n",argv[1]);
7.          printf("%s\n",argv[2]);
8.      }
9.      else printf("引数個数のエラー\n");
```

```
10. }
```

注意 5-3-2

5. 引数の個数が 2 つの場合
6,7. それぞれの引数を改行して表示する

解答 5-3-3

```
1.  #include <stdio.h>
2.  main(int argc,char *argv[])
3.  {
4.      if(argc>2) printf("ERROR!\n");
5.      else if(argc<2) printf("ARGUMENT!\n");
6.      else{
7.          if(*argv[1]=='-'||*argv[1]=='/')
8.              printf("YES\n");
9.          else printf("NO\n");
10.     }
11. }
```

注意 5-3-3

4. 引数の個数が 2 つ以上の場合
5. 引数がない場合
6. 引数が 1 つの場合
7. 引数の先頭文字が'-'または'/'の場合

解答 5-3-4

```
1.  #include <stdio.h>
2.  main(int argc,char *argv[])
3.  {
4.      char c;
5.      int i;
6.      for(i=1;i<argc;i++){
7.          if(*argv[i]=='/'){
8.              c=*(argv[i]+1);
9.              if(c=='a'||c=='A'){
10.                 if(*(argv[i]+2)==0x0){
11.                     printf("YES\n");
12.                     return;
13.                 }
14.             }
```

```
15.         }
16.     }
17.     printf("NO\n");
18. }
```

注意 5-3-4

6. 引数を最初から最後まで調べる
7. 引数の先頭が'/'の場合
8. 引数の 2 番目の文字を c に代入
9. 2 番目の文字が'a'または'A'の場合
10. 3 番目の文字が NUL(0)の場合

6章 標準ライブラリ関数

6-1) 標準ライブラリ関数

解答 6-1-1

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  main()
4.  {
5.      char str[30];
6.      printf("文字列 > ");
7.      scanf("%s",str);
8.      strupr(str);
9.      printf("%s",str);
10. }
```

注意 6-1-1

1. printf(),scanf()関数を使用するためのインクルードファイル
2. strupr()関数を使用するためのインクルードファイル
8. str 内の英大文字を英小文字に変える

解答 6-1-2

```
1.  #include <stdio.h>
2.  #include <ctype.h>
3.  #include <conio.h>
4.  main()
5.  {
6.      int c;
7.      while((c=getch())!=0x1b){
8.          if(isalnum(c)) putchar(c);
9.      }
10. }
```

注意 6-1-2

1. putchar()関数を使用するためのインクルードファイル
2. isalnum()関数を使用するためのインクルードファイル
3. getch()関数を使用するためのインクルードファイル
7. getch()関数はリターンを待たずにキーを押す毎にコードが入力される
   ESC($1B_H$)キーの押下でループから抜け出る
8. c が英数字でなければ偽，英数字なら真

解答 6-1-3

```
1.  #include <stdio.h>
2.  #include <direct.h>
3.  main()
4.  {
5.      char buff[80];
6.      getcwd(buff,80);
7.      printf("%s\n",buff);
8.  }
```

注意 6-1-3

1. printf()関数を使用するためのインクルードファイル
2. getcwd()関数を使用するためのインクルードファイル
6. パス名の最大長を表す 80 は 0(NUL)も含めた長さ

解答 6-1-4

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  main()
4.  {
5.      char str1[30],str2[30];
6.      int a;
7.      printf("変換前文字列 > ");
8.      scanf("%s",str1);
9.      a=atoi(str1);
10.     printf("数値変換後   > %d\n",a);
11.     itoa(a,str2,10);
12.     printf("文字列変換後 > %s\n",str2);
13. }
```

注意 6-1-4

1. printf(),scanf()関数を使用するためのインクルードファイル
2. atoi(),itoa()関数を使用するためのインクルードファイル
8. 数字の並んだ文字列を入力する
9. 整数値に変換する
11. 10 進数表示で数値 a を文字列に変換し，str2 へ格納する

解答 6-1-5

```
 1.  #include <stdio.h>
 2.  main()
 3.  {
 4.      char a[30];
 5.      int c;
 6.      printf("入力 > ");
 7.      c=getchar();
 8.      ungetc(c,stdin);
 9.      if(c>='0'&&c<='9'){
10.          scanf("%d",&c);
11.          printf("数値 = %d\n",c);
12.      }
13.      else{
14.          scanf("%s",a);
15.          printf("文字 = %s\n",a);
16.      }
17. }
```

注意 6-1-5

1. printf(),scanf(),getchar(),ungetc()関数を使用するためのインクルードファイル
7. 最初の 1 文字を読み込む
8. 読み込んだ文字 c を標準入力(stdin)に戻す
9. 読み込んだ文字が'0'～'9'であるか判定する
10. int 型の数値として読み込む
14. 文字列として読み込む

解答 6-1-6

```
 1.  #include <stdio.h>
 2.  #include <math.h>
 3.  main()
 4.  {
 5.      double i,x,x1,x2;
 6.      printf("         x    log(x)  log10(x)\n");
 7.      for(i=1.0;i<=20.0;i++){
 8.          x=i/10.0;
 9.          x1=log(x);
10.          x2=log10(x);
11.          printf("%10.5f%10.5f%10.5f\n",x,x1,x2);
```

```
12.     }
13. }
```

注意 6-1-6

1. printf()関数を使用するためのインクルードファイル
2. log(),log10()関数を使用するためのインクルードファイル
8. 関数表の見出し
9. 0.1 から増分 0.1 で始めると終了判定が誤差に影響される整数値の場合は誤差がない

9,10. for(x=0.1;x<=20.01;x+=0.1){ でもよい
20.01 の .01 は誤差対策である

解答 6-1-7

```
1. #include <stdio.h>
2. #include <time.h>
3. main()
4. {
5.     time_t t;
6.     time(&t);
7.     printf("%ld\n",t);
8.     printf("%s\n",ctime(&t));
9. }
```

注意 6-1-7

1. printf()関数を使用するためのインクルードファイル
2. time(),ctime()関数を使用するためのインクルードファイル
5. time_t 型は long 型に同じ

解答 6-1-8

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. main()
4. {
5.     int i;
6.     int ri;
7.     double rd;
8.     srand(1);
9.     for(i=1;i<=100;i++){
10.        rd=rand()/32768.0;
11.        printf("%8.4f",rd);
```

```
12.     }
13.     printf("\n");
14.     for(i=1;i<=100;i++){
15.         ri=(int)(rand()/32768.0*100.0)+100;
16.         printf("%8d",ri);
17.     }
18.     printf("\n");
19. }
```

注意 6-1-8

1. printf()関数を使用するためのインクルードファイル
2. srand(),rand()関数を使用するためのインクルードファイル
8. 初期化によって乱数の系列を決める
10. 0 以上 1 未満の乱数を作る
15. 100 以上 200 未満の整数乱数を作る

解答 6-1-9

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  main()
4.  {
5.      int i,no=0,max=1000;
6.      double x,y,z;
7.      srand(1);
8.      for(i=0;i<max;i++){
9.          x=(double)rand()/RAND_MAX;
10.         y=(double)rand()/RAND_MAX;
11.         if(x*x+y*y<1.0) no++;
12.     }
13.     printf("面積 = %.3f\n",(double)4*no/max);
14. }
```

注意 6-1-9

1. printf()関数を使用するためのインクルードファイル
2. srand(),rand()関数を使用するためのインクルードファイル
5. no は円内に入ったドットの数を数えるカウンタである
   max は全ドット数であるが，練習なので 1000 回にしている
9. ドットの x 座標(0≤ x ≤1)を決める
   RAND_MAX は rand()関数が発生する数の最大値である
10. y 座標についても x 座標と同様である
11. 円内に入った場合はカウンタを 1 つ増加させる

13. 円の面積 = 4×1×円内に入ったドットの割合

解答 6-1-10

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  main()
4.  {
5.      char *str[5]={"OKAYAMA","OSAKA",
6.                    "KYOTO","NAGOYA","TOKYO"};
7.      int i,min,max;
8.      min=max=0;
9.      for(i=1;i<5;i++){
10.         if(strcmp(str[min],str[i])>0) min=i;
11.         if(strcmp(str[max],str[i])<0) max=i;
12.     }
13.     printf("min = %s\n",str[min]);
14.     printf("max = %s\n",str[max]);
15. }
```

注意 6-1-10

1. printf()関数を使用するためのインクルードファイル
2. strcmp()関数を使用するためのインクルードファイル

5,6. ポインタ配列の初期化

7. min,max には最小，最大の文字列配列の要素番号が入る
8. 最初は 0 から始める
10. 最小文字列の抽出
11. 最大文字列の抽出

解答 6-1-11

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  main()
4.  {
5.      char str1[40],str2[20];
6.      strcpy(str1,"いちご");
7.      strcpy(str2,"大福");
8.      strcat(str1,str2);
9.      printf("%s\n",str1);
10. }
```

注意 6-1-11

1. printf()関数を使用するためのインクルードファイル
2. strcpy(),strcat()関数を使用するためのインクルードファイル

6,7. 文字列の代入，自動的に最後に NUL(0)が付く

8. str1 の文字列の最後に str2 の文字列を連結する
   自動的に最後に NUL(0)が付く
   str1 の内容は連結されたものに変わる

解答 6-1-12

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  main()
4.  {
5.      char str[80],buff[80],*s1,*s2,*p;
6.      int len1,len2;
7.      s1="BOOK";
8.      s2="LETTER";
9.      len1=strlen(s1);
10.     len2=strlen(s2);
11.     printf("BOOK を含む文字列 > ");
12.     gets(str);
13.     p=strstr(str,s1);
14.     while(p!=NULL){
15.         strcpy(buff,p+len1);
16.         *p=0x0;
17.         strcat(str,s2);
18.         strcat(str,buff);
19.         p=strstr(p+len2,s1);
20.     }
21.     printf("%s\n",str);
22. }
```

注意 6-1-12

1. gets(),printf()関数を使用するためのインクルードファイル
2. strlen(),strstr(),strcpy(),strcat()関数を使用するためのインクルードファイル

9,10. それぞれの文字列の長さが後で必要になる

14. str の先頭から文字列"BOOK"を探す

15. 見つかった場合

16. 見つかった文字列より後ろの文字列を buff に格納する

17. 見つかった位置へNUL(0)を代入し文字列の最後とする
18. 前の操作で短くなったstrの後ろに文字列"LETTER"を連結する
19. さらにbuffに入っている後続文字列を連結する
20. 再度今変換した"LETTER"の後ろから"BOOK"を検索してゆく

7章 演算子

7-1) 論理演算子

解答 7-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0,b=1,c=2;
5.      printf("!%d = %d\n",a,!a);
6.      printf("!%d = %d\n",b,!b);
7.      printf("!%d = %d\n",c,!c);
8.  }
```

注意 7-1-1

5-7. !a は a の論理演算の否定である
0 は偽, 0 以外は真である

解答 7-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0,b=1,c=2;
5.      printf("%d && %d = %d\n",a,b,a&&b);
6.      printf("%d && %d = %d\n",a,c,a&&c);
7.      printf("%d && %d = %d\n",b,c,b&&c);
8.      printf("%d || %d = %d\n",a,b,a||b);
9.      printf("%d || %d = %d\n",a,c,a||c);
10.     printf("%d || %d = %d\n",b,c,b||c);
11.     printf("%d xor %d = %d\n",a,b,!a&&b||a&&!b);
12.     printf("%d xor %d = %d\n",a,c,!a&&c||a&&!c);
13.     printf("%d xor %d = %d\n",b,c,!b&&c||b&&!c);
14. }
```

注意 7-1-2

5-7. a&&b は a と b の論理演算の AND である
8-10. a||b は a と b の論理演算の OR である
11-13. A XOR B = ((NOT A) AND B) OR (A AND (NOT B))

7-2) 代入演算子

解答 7-2-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i=0,a[10];
5.      while(i<10) a[i++]=i;
6.      for(i=0;i<10;i++) printf("%d ",a[i]);
7.      printf("\n");
8.  }
```

注意 7-2-1

4. i の初期値は 0 である
5. while(i<10){ a[i]=i; i=i+1; } と同じ手順

解答 7-2-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i=-1,a[10];
5.      while(i<10) a[++i]=i;
6.      for(i=0;i<10;i++) printf("%d ",a[i]);
7.      printf("\n");
8.  }
```

注意 7-2-2

4. i の初期値は-1 である
5. while(i<10){ i=i+1; a[i]=i; } と同じ手順

解答 7-2-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int total=0,i;
5.      for(i=1;i<=100;i++) total+=i;
6.      printf("合計 = %d\n",total);
7.  }
```

注意 7-2-3

```
5.  total=total+i;
```

解答 7-2-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long fact=1l,i;
5.      for(i=1;i<=10;i++) fact*=i;
6.      printf("%ld\n",fact);
7.  }
```

注意 7-2-4

```
5.  fact=fact*i;
```

解答 7-2-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=2;
5.      printf("(a==4) = %d\n",a==4);
6.      printf("(a=4)  = %d\n",a=4);
7.  }
```

注意 7-2-5

5. a==4 式の値は真のとき 1, 偽のとき 0 である
6. a=4 式の値は 4 である

解答 7-2-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i,*a,c[30],str[30];
5.      printf("文字列 > ");
6.      scanf("%s",str);
7.      a=str;
8.      i=0;
9.      while((c[i++]=*a++)!=0);
10.     printf("%s\n",c);
```

```
11. }
```

注意 7-2-6

9. 演算子の優先順位に注意
   while(c[i++]=*a,*a++!=0); としてもよい

7-3) シフト演算子

解答 7-3-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=45;
5.      printf("%d %d\n",a>>1,a<<3);
6.      printf("%d %d\n",a/2,a*8);
7.  }
```

注意 7-3-1

5. 1 ビット右シフトと 3 ビット左シフト
   1 ビットの右シフトは 2^1=2 で割ることに等しい
   3 ビットの左シフトは 2^3=8 を掛けることに等しい

解答 7-3-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a1=0x2222,b1=0xaaaa;
5.      unsigned int a2=0x2222,b2=0xaaaa;
6.      printf("%4x %4x\n",a1<<4,a1>>4);
7.      printf("%4x %4x\n",a2<<4,a2>>4);
8.      printf("\n");
9.      printf("%4x %4x\n",b1<<4,b1>>4);
10.     printf("%4x %4x\n",b2<<4,b2>>4);
11. }
```

注意 7-3-2

5,6. 0x2222 のように先頭ビットが 0 の場合
   int 型も unsigned int 型も同じ結果を得る
9,10. 0xaaaa のように先頭ビットが 1 の場合
   int 型では 4 ビットの右シフトが 0xfaaa になる

解答 7-3-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long a=0x12345678;
5.      printf("%8lx %8lx\n",a<<8,a>>8);
6.  }
```

注意 7-3-3

5. シフト演算は long 型や unsigned long 型でも同様に行われる
   long 型の場合の右シフトは先頭ビットに注意する

解答 7-3-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0x1234;
5.      printf("0x%x\n",a>>8);
6.  }
```

注意 7-3-4

5. 右に 8 ビットシフトすれば 0x12 となる

解答 7-3-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0x120;
5.      printf("0x%x 0x%x\n",a>>4,a<<4);
6.  }
```

注意 7-3-5

5. 0x12 は 4 ビット右にシフトし，0x1200 は 4 ビット左にシフトする

解答 7-3-6

```
1.  #include <stdio.h>
2.  main()
3.  {
```

```
4.      unsigned int a=0xff12;
5.      printf("0x%x\n",a>>8);
6.  }
```

注意 7-3-6

4. a が int 型だと 8 ビットの右シフトで 0xffff となる

解答 7-3-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=200;
5.      printf("%d\n",a>>3);
6.  }
```

注意 7-3-7

5. 8 で割る演算は 3 ビット右シフトするのに等しい

解答 7-3-8

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=10;
5.      printf("%d\n",a<<6);
6.  }
```

注意 7-3-8

5. 64 を掛ける演算は 6 ビット左シフトするのに等しい

7-4) ビット演算子

解答 7-4-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a=0xaa55;
5.      printf("%x\n",~a);
6.  }
```

注意 7-4-1

5. ~ はビットごとの NOT 演算子

解答 7-4-2

```
1. #include <stdio.h>
2. main()
3. {
4.     int a=5,b=12;
5.     printf("%d&%d = %d\n",a,b,a&b);
6.     printf("%d|%d = %d\n",a,b,a|b);
7.     printf("%d^%d = %d\n",a,b,a^b);
8. }
```

注意 7-4-2

5. & はビットごとの AND 演算子
6. | はビットごとの OR 演算子
7. ^ はビットごとの XOR 演算子

解答 7-4-3

```
1. #include <stdio.h>
2. main()
3. {
4.     int a;
5.     printf("整数 > ");
6.     scanf("%d",&a);
7.     printf("3bit 目 = %d\n",(a&0x4)==0x4);
8.     printf("8bit 目 = %d\n",(a&0x80)==0x80);
9. }
```

注意 7-4-3

7. a&0x4 が 0x4 ならば下位 3 ビット目が 1 である
8. a&0x80 が 0x80 ならば下位 8 ビット目が 1 である

$$\begin{array}{r} 0001\ 0010\ 0001\ 0111_B \\ \underline{\&\ 0000\ 0000\ 0000\ 0100_B} \\ 0000\ 0000\ 0000\ 1000_B \end{array} \qquad \begin{array}{r} 0001\ 0010\ 0001\ 0011_B \\ \underline{\&\ 0000\ 0000\ 1000\ 0000_B} \\ 0000\ 0000\ 0000\ 0000_B \end{array}$$

解答 7-4-4

```
1. #include <stdio.h>
2. main()
```

```
3.  {
4.      int a;
5.      printf("16進数 > ");
6.      scanf("%x",&a);
7.      printf("%x\n",a|0x88);
8.  }
```

注意 7-4-4

7. 下位4,8ビット目を1にするには0x88とのビットごとのORをとる

$$\begin{array}{r} 0001\ 0010\ 0001\ 1111_B \\ \underline{|\ 0000\ 0000\ 1000\ 1000_B} \\ 0001\ 0010\ 1000\ 1111_B \end{array}$$

解答 7-4-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      printf("16進数 > ");
6.      scanf("%x",&a);
7.      printf("%x\n",a&0xf7df);
8.  }
```

注意 7-4-5

7. 下位6,12ビット目を0にするには0xf7dfとのビットごとのANDをとる

$$\begin{array}{r} 0001\ 1010\ 0001\ 0011_B \\ \underline{\&\ 1111\ 0111\ 1101\ 1111_B} \\ 0001\ 0010\ 0001\ 0011_B \end{array}$$

解答 7-4-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      unsigned int a=0x1234,b,c;
5.      b=a<<4;
6.      c=a>>12;
7.      printf("0x%x\n",b|c);
8.  }
```

注意 7-4-6

5. b は 0x2340 になる
6. : c は 0x0001 になる
7. b|c は 0x2341 になる

解答 7-4-7

```
1.  #include <stdio.h>
2.  void binary(int x);
3.  main()
4.  {
5.      int a;
6.      printf("整数 > ");
7.      scanf("%d",&a);
8.      binary(a);
9.  }
10. void binary(int x)
11. {
12.     int i;
13.     for(i=15;i>=0;i--)
14.         printf("%d",(x>>i)&0x1);
15.     printf("B");
16.     printf("\n");
17. }
```

注意 7-4-7

13,14. 上位ビットから順番に再下位バイトに持ってゆき，0 か 1 かを調べる

8章 構造体と共用体

8-1) 構造体

解答 8-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      struct shain {
5.          char *name;
6.          int age;
7.          int income;
8.      }a;
9.      a.name="佐々木康宏";
10.     a.age=22;
11.     a.income=380;
12.     printf("%s %d %d\n",a.name,a.age,a.income);
13. }
```

注意 8-1-1

4-8.　構造体型の宣言
5-7.　メンバの宣言
8.　構造体変数 a の宣言
9-11. データの代入
　　a が値の場合は a.name のように ． を付けて表す

解答 8-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      struct shain {
5.          char *name;
6.          int age;
7.          int income;
8.      };
9.      struct shain a={"佐々木康宏",22,380};
10.     printf("%s %d %d\n",a.name,a.age,a.income);
11. }
```

注意 8-1-2

4-8．構造体型の宣言
9．　構造体変数の初期化
10．a が値の場合は a.name のように ．を付けて表す

解答 8-1-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      struct shain {
5.          char *name;
6.          int age;
7.          int income;
8.      };
9.      struct shain a={"佐々木康宏",22,380},*p;
10.     p=&a;
11.     printf("%s %d %d\n",p->name,p->age,p->income);
12. }
```

注意 8-1-3

4-8．構造体型の宣言
9．　構造体変数の初期化とポインタ変数の宣言
10．p がポインタの場合は p->name のように -> を付けて表す

解答 8-1-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      typedef struct uriage{
5.          char *name;
6.          long price;
7.          int num;
8.      }SALES;
9.      SALES a={"PR1100",148000,8};
10.     printf("%s %ld %d\n",a.name,a.price,a.num);
11. }
```

注意 8-1-4

4-8．typedef による SALES 型の宣言
9．　SALES 型変数の初期化

解答 8-1-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      struct shain{
6.          char *name;
7.          int age;
8.          int inc;
9.          int year;
10.     }a[3];
11.     a[0].name="安部　健"; a[0].age=38;
12.     a[0].inc=562; a[0].year=16;
13.     a[1].name="石丸敬二"; a[1].age=30;
14.     a[1].inc=430; a[1].year=8;
15.     a[2].name="鎌刈智恵"; a[2].age=24;
16.     a[2].inc=358; a[1].year=2;
17.     for(i=0;i<3;i++)
18.         printf("%10s%4d%6d%4d\n",
19.             a[i].name,a[i].age,a[i].inc,a[i].year);
20. }
```

注意 8-1-5

5-10. 構造体型と構造体配列の宣言
11-16. 構造体配列メンバへのデータの代入
17-19. 構造体配列を利用した表示

解答 8-1-6

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      struct shohin{
6.          char *name;
7.          float price;
8.          int num;
9.      };
10.     struct shohin a[]={{"PR1100",148000,8},
11.                        {"PR2100",218000,15},
```

```
12.                         {"PR3100",284000,4}};
13.     for(i=0;i<3;i++)
14.         printf("%8s%8.0f%4d\n",
15.             a[i].name,a[i].price,a[i].num);
16. }
```

注意 8-1-6

10-12. 構造体配列の初期化
14,15. 構造体配列を利用した表示

解答 8-1-7

```
1.  #include <stdio.h>
2.  #include <dos.h>
3.  main()
4.  {
5.      struct date date;
6.      struct time time;
7.      getdate(&date);
8.      gettime(&time);
9.      printf("%04d-%02d-%02d/%02d:%02d:%02d\n",
10.         date.da_year,date.da_mon,date.da_day,
11.         time.ti_hour,time.ti_min,time.ti_sec);
12. }
```

注意 8-1-7

2. getdate(),gettime()関数を使用するためのインクルードファイル
5. getdate()関数で使用する構造体で dos.h の中で定義されている
6. gettime()関数で使用する構造体で dos.h の中で定義されている
9-11. date 型と time 型構造体のメンバ表示

8-2) 共用体

解答 8-2-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      long x;
5.      union body{
6.          unsigned int a;
7.          long b;
```

```
8.      } dat;
9.      printf("整数 > ");
10.     scanf("%ld",&x);
11.     if(x<0x10000l){
12.         dat.a=(unsigned int)x;
13.         printf("unsinged : %u\n",dat.a);
14.     }
15.     else{
16.         dat.b=x;
17.         printf("long : %ld\n",dat.b);
18.     }
19. }
```

注意 8-2-1

5-8. unsigned int 型メンバと long 型メンバの共用体
10. 最初は long 型変数に読み込む
11. unsigned int 型は 0～$FFFF_H$ の値をとる
13. 符号なし整数の表示には%u を用いる

解答 8-2-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      union body{
5.          char name[21];
6.          int age;
7.          int income;
8.      } dat;
9.      strcpy(dat.name,"安部健");
10.     printf("%s\n",dat.name);
11.     dat.age=22;
12.     printf("%d\n",dat.age);
13.     dat.income=450;
14.     printf("%d\n",dat.income);
15. }
```

注意 8-2-2

9-14. データを代入しながら表示している
最終的には最後の data.income=450 が残るだけである

解答 8-2-3

```
1.  #include <stdio.h>
2.  void bitpat(unsigned long);
3.  main()
4.  {
5.      union frep{
6.          float x;
7.          unsigned long a;
8.      }z;
9.      printf("float > ");
10.     scanf("%f",&(z.x));
11.     bitpat(z.a);
12. }
13. void bitpat(unsigned long n)
14. {
15.     int i;
16.     for(i=31;i>=0;i--){
17.         if(i==30||i==22) printf(" ");
18.         printf("%d",(n>>i)&1);
19.     }
20.     printf("B\n");
21. }
```

注意 8-2-3

7. ビット表示には符号なしがよい
17. 先頭ビットが符号、次の 8 桁が指数部、最後の 23 ビットが仮数部

解答 8-2-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      struct crep{
5.          unsigned char a,b;
6.      };
7.      union irep{
8.          int x;
9.          struct crep y;
10.     }z;
11.     z.x=0x1234;
12.     printf("z.x=%xH\n",z.x);
13.     printf("z.y.a=%xH z.y.b=%xH\n",z.y.a,z.y.b);
```

```
14. }
```

注意 8-2-4

5. メモリの番地の小さい順に a, b と割り当てられる
13. メモリの番地の小さい純に 34H, 12H と表示される

解答 8-2-5

```
1.  #include <stdio.h>
2.  struct wregs{
3.      unsigned int ax,bx,cx,dx;
4.  };
5.  struct bregs{
6.      unsigned char ah,al,bh,bl,ch,cl,dh,dl;
7.  };
8.  union myregs{
9.      struct wregs x;
10.     struct bregs h;
11. };
12. main()
13. {
14.     union myregs regs;
15.     regs.h.ah=10; regs.h.al=0; regs.x.bx=1;
16.     regs.h.ch=0; regs.h.cl=1; regs.x.dx=5;
17.     printf("%4d%4d\n",regs.h.ah,regs.h.al);
18.     printf("%4d\n",regs.x.bx);
19.     printf("%4d%4d\n",regs.h.ch,regs.h.cl);
20.     printf("%4d\n",regs.x.dx);
21. }
```

注意 8-2-5

2-11: 8086 のレジスタを操作するための標準の共用体に類似している
8-10: 共用体のメンバとして構造体を用いている
15,16: 共用体のメンバである構造体のメンバにデータを代入する
17-20: 共用体のメンバである構造体のメンバを参照する
　　変数名とメンバ名を . または -> 演算子で連結する

解答 8-2-6

```
1.  #include <stdio.h>
2.  struct wregs{
3.      unsigned int ax,bx,cx,dx;
4.  };
```

```
 5.  struct bregs{
 6.      unsigned char ah,al,bh,bl,ch,cl,dh,dl;
 7.  };
 8.  union myregs{
 9.      struct wregs x;
10.     struct bregs h;
11. };
12. main()
13. {
14.     union myregs regs,*p;
15.     regs.h.ah=10; regs.h.al=0; regs.x.bx=1;
16.     regs.h.ch=0; regs.h.cl=1; regs.x.dx=5;
17.     p=&regs;
18.     printf("%4d%4d\n",p->h.ah,p->h.al);
19.     printf("%4d\n",p->x.bx);
20.     printf("%4d%4d\n",p->h.ch,p->h.cl);
21.     printf("%4d\n",p->x.dx);
22. }
```

注意 8-2-6

2-11. 8086 のレジスタを操作するための標準の共用体に類似している
8-10. 共用体のメンバとして構造体を用いている
15,16. 共用体のメンバである構造体のメンバにデータを代入する
18-21. 共用体のメンバである構造体のメンバを参照する
　　変数名とメンバ名を . または -> 演算子で連結する

9章 プリプロセッサ

9-1) プリプロセッサ

解答 9-1-1

```
1.  #include <stdio.h>
2.  #define PI 3.14159
3.  #define PRINT printf
4.  main()
5.  {
6.      float r=10.0;
7.      PRINT("面積 = %f\n",PI*r*r);
8.  }
```

注意 9-1-1

1. stdio.h をシステム標準のディレクトリから読み込む
2. PI を 3.14159 に置き換える
3. PRINT を printf に置き換える
7. コンパイル時に PRINT は printf に置き換えられる
   コンパイル時に PI は 3.14159 に置き換えられる

解答 9-1-2

```
1.  #include <stdio.h>
2.  #define INPUT scanf
3.  #define PRINT printf
4.  main()
5.  {
6.      char str[30];
7.      PRINT("名前 > ");
8.      INPUT("%s",str);
9.      PRINT("%s\n",str);
10. }
```

注意 9-1-2

1. stdio.h をシステム標準のディレクトリから読み込む
2. INPUT を scanf に置き換える
3. PRINT を printf に置き換える
7,9. コンパイル時に PRINT は printf に置き換えられる
9. コンパイル時に INPUT は scanf に置き換えられる

解答 9-1-3

```
1.  #include <stdio.h>
2.  #define GREEN printf("\x1b[32m")
3.  #define NORMAL printf("\x1b[0m")
4.  main()
5.  {
6.      GREEN;
7.      printf("GREEN\n");
8.      NORMAL;
9.  }
```

注意 9-1-3

2. GREEN をエスケープシーケンスのテキスト属性緑色指定で置き換える
3. NORMAL をエスケープシーケンスのテキスト属性標準で置き換える
6. コンパイル時に printf("\x1b[32m"); に置き換えられる
8. コンパイル時に printf("\x1b[0m"); に置き換えられる

解答 9-1-4

```
1.  #include <stdio.h>
2.  #define CLS printf("\x1b*")
3.  #define cjump(x,y) printf("\x1b[%d;%dH",(y)+1,(x)+1)
4.  main()
5.  {
6.      CLS;
7.      cjump(30,10);
8.      printf("x(30,10)");
9.      cjump(0,0);
10. }
```

注意 9-1-4

2 CLS をエスケープシーケンスの画面消去で置き換える
3 cjump(変数,変数)をエスケープシーケンスのカーソル移動で置き換える
6 printf("\x1b*"); に置き換えられる
7 printf("\x1b[%d,%dH",(30)+1,(10)+1); に置き換えられる
9 printf("\x1b[%d,%dH",(0)+1,(0)+1); に置き換えられる

解答 9-1-5

```
1.  #include <stdio.h>
2.  #define PC98
```

```
3.  #define CLS printf("\x1b*")
4.  #define NORMAL printf("\x1b[0m")
5.  #define COLOR1 printf("\x1b[34m")
6.  #define COLOR2 printf("\x1b[31m")
7.  #define COLOR3 printf("\x1b[35m")
8.  #define COLOR4 printf("\x1b[32m")
9.  #define COLOR5 printf("\x1b[36m")
10. #define COLOR6 printf("\x1b[33m")
11. #define COLOR7 printf("\x1b[37m")
12. #define cjump(x,y) printf("\x1b[%d;%dH",(y)+1,(x)+1)
13. #if defined PC98
14.     #define CREP printf("\x1b[>5l")
15.     #define CVAN printf("\x1b[>5h")
16. #elif defined FMR
17.     #define CREP printf("\x1b[0v")
18.     #define CVAN printf("\x1b[1v")
19. #else
20.     #define CREP
21.     #define CVAN
22. #endif

23. main()
24. {
25.     CLS;
26.     COLOR1; printf("color\n");
27.     COLOR2; printf("color\n");
28.     COLOR3; printf("color\n");
29.     COLOR4; printf("color\n");
30.     COLOR5; printf("color\n");
31.     COLOR6; printf("color\n");
32.     COLOR7; printf("color\n");
33.     CVAN;
34.     cjump(30,15);
35.     printf("Hit Enter key!");
36.     getchar();
37.     NORMAL;
38.     CREP;
39. }
```

注意 9-1-5

2. 以下の #if 〜 #endif を用いるために定義する

3-12．ＰＣ９８とＦＭＲで共通なエスケープシーケンスのマクロ化
13．PC98 が定義されていれば以下 2 行がコンパイルされる
　　このプログラムでは定義されているのでコンパイルされる
14．CREP をエスケープシーケンスのカーソル表示に置き換える
15．CVAN をエスケープシーケンスのカーソル消去に置き換える
16．FMR が定義されていれば以下 2 行がコンパイルされる
　　このプログラムではコンパイルされない
17．CREP をエスケープシーケンスのカーソル表示に置き換える
18．CVAN をエスケープシーケンスのカーソル消去に置き換える
19．その他の場合以下 2 行がコンパイルされる
　　このプログラムではコンパイルされない
27-33．色を付けて color と表示する
34．カーソルの消去
35．printf("\x1b[%d,%dH",(30)+1,(15)+1); に置き換えられる
37．リターンを受け付ける
38．テキスト属性を標準に設定

解答 9-1-6

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  #define CLS printf("\x1b*")
4.  #define NORMAL printf("\x1b[0m")
5.  #define REVYEL printf("\x1b[33;7m")
6.  #define cjump(x,y) printf("\x1b[%d;%dH",(y)+1,(x)+1)
7.  #define CREP printf("\x1b[>5l")
8.  #define CVAN printf("\x1b[>5h")
9.  int select(int,int,int,char **);

10. main()
11. {
12.     char *menu[3];
13.     int ans;
14.     menu[0]="ファイル入力";
15.     menu[1]="ファイル出力";
16.     menu[2]="  終     了  ";
17.     CLS;
18.     ans=select(30,10,3,menu);
19.     cjump(26,20);
20.     printf("選択 : %s\n",menu[ans]);
21. }
```

```
22. int select(int x,int y,int max,char *str[])
23. {
24.     int i,c,n;
25.     CVAN;
26.     REVYEL;
27.     cjump(x,y);
28.     printf("%s",str[0]);
29.     NORMAL;
30.     for(i=1;i<max;i++){
31.         cjump(x,y+i);
32.         printf("%s",str[i]);
33.     }
34.     n=0;
35.     while(1){
36.         c=getch();
37.         if(c==0xa){
38.             NORMAL; cjump(x,y+n);
39.             printf("%s",str[n]);
40.             n=(n+1)%max;
41.             REVYEL; cjump(x,y+n);
42.             printf("%s",str[n]);
43.         }
44.         else if(c==0xb){
45.             NORMAL; cjump(x,y+n);
46.             printf("%s",str[n]);
47.             n=(n+max-1)%max;
48.             REVYEL; cjump(x,y+n);
49.             printf("%s",str[n]);
50.         }
51.         else if(c==0xd){
52.             NORMAL; cjump(x,y+n);
53.             printf("%s",str[n]);
54.             CREP;
55.             return n;
56.         }
57.         else ;
58.     }
59. }
```

注意 9-1-6

2． getch()関数のためのインクルードファイル
3-8．マクロ定義
9． 関数のプロトタイプ宣言，char ** はポインタ配列の宣言
11 (x,y)はメニューの表示位置，max はメニューの数
str[]はメニュー項目名文字列のポインタ配列
12．メニュー項目名文字列用のポインタ配列の宣言
14-16．メニュー項目名の先頭ポインタを代入する
17．画面消去
18．(30,10)の位置に項目数 3 で menu[]で与えられる項目名のメニューを表示し，選択させる
選択した番号は ans に代入される
20．選択した番号のメニュー項目名が表示される
25．カーソルを消去する
26-28．最初のメニュー項目名を黄色反転で表示する
30-33．2 番目以降のメニュー項目を 1 行ずつずらして標準色で表示する
34．n は 0≤n≤max-1 で，現在の黄色反転のメニュー項目番号を表す
35．メニューが選択されるまでの無限ループ
36．キーボードからのエコーバック(入力キーの表示)なしの入力
37．下向き矢印キーが押された場合
38,39．黄色反転していたメニュー項目名を標準色に変えて表示する
40．n の値を 1 つ増やす，n=max-1 の場合は 0 にする
41,31．1 つ次のメニュー項目名を黄色反転して表示する
44．上向き矢印キーが押された場合
45,46．黄色反転していたメニュー項目名を標準色に変えて表示する
47．n の値を 1 つ減らす，n=0 の場合は max-1 にする
48,49．1 つ前のメニュー項目名を黄色反転して表示する
51．リターンキーが押された場合
52,53．1 つ次のメニュー項目名を黄色反転して表示する
54．カーソルを表示する
55．戻値は選択した項目番号とする
57．これら以外の場合は何の処理も行わない

１０章 ファイル処理

10-1) 高水準入出力

解答 10-1-1

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      FILE *fp;
5.      fp=fopen("natu.dat","wt");
6.      if(fp==NULL){
7.          printf("ファイルがオープン出来ません。\n");
8.          return;
9.      }
10.     fprintf(fp,"沖縄\n 北海道\n");
11.     fclose(fp);
12. }
```

注意 10-1-1

1. printf(),fopen(),fprintf()関数のためのインクルードファイル
4. ファイルポインタ fp の宣言
5. ファイル natu.dat をテキスト書き込みモードでオープンする
6. ファイルがオープン出来なかった場合は fp に NULL が返る

5,6. if((fp=fopen("natu.dat","wt"))==NULL){ としてもよい

8. プログラムの終了
10. 改行しながら出力する
11. ファイルのクローズを忘れてはいけない

解答 10-1-2

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      FILE *fp;
5.      fp=fopen("natu.dat","at");
6.      if(fp==NULL){
7.          printf("ファイルがオープン出来ません。\n");
8.          return;
9.      }
10.     fprintf(fp,"海！山！\n");
```

```
11.     fclose(fp);
12. }
```

注意 10-1-2

```
1.  printf(),fopen(),fprintf()関数のためのインクルードファイル
4.  ファイルポインタ fp の宣言
5.  ファイル natu.dat をテキスト追加モードでオープンする
    ファイルの書き込み位置はファイルエンドになる
6.  ファイルがオープン出来なかった場合は fp に NULL が返る
5,6. if((fp=fopen("natu.dat","wt"))==NULL){ としてもよい
8.  プログラムの終了
10. ファイルエンドに追加する
11. ファイルのクローズを忘れないように
```

解答 10-1-3

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file3.dat";
5.      int i,a[10]={1,2,3,4,5,0,9,8,7,6},b;
6.      FILE *fp;
7.      fp=fopen(fname,"w");
8.      if(fp==NULL){
9.          printf("ファイルがオープン出来ません。\n");
10.         return;
11.     }
12.     for(i=0;i<10;i++) fprintf(fp,"%d ",a[i]);
13.     fclose(fp);
14.     fp=fopen(fname,"r");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     for(i=0;i<10;i++){
20.         fscanf(fp,"%d",&b);
21.         printf("%d ",b);
22.     }
23.     printf("\n");
24.     fclose(fp);
25. }
```

注意 10-1-3

4. ファイル名は変数に入れておく
5. データの初期化
6. ファイルポインタ fp の宣言
7. 書き込みモードでオープンする
   出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
12. 間に空白を１つずつ入れてデータを書き込む
13. ファイルのクローズ
    書き込みと読み込みをはっきりさせるために一度クローズしている
14. 読み込みモードでオープンする
    入力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
20. 空白，タブ，リターンを区切りとしたデータの読み込み
24. ファイルのクローズを忘れないように

解答 10-1-4

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file4.dat";
5.      int i,a[5]={115,33,257,118,5},b;
6.      FILE *fp;
7.      fp=fopen(fname,"w");
8.      if(fp==NULL){
9.          printf("ファイルがオープン出来ません。\n");
10.         return;
11.     }
12.     for(i=0;i<5;i++) fprintf(fp,"%3d",a[i]);
13.     fclose(fp);
14.     fp=fopen(fname,"r");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     for(i=0;i<5;i++){
20.         fscanf(fp,"%3d",&b);
21.         printf("%3d ",b);
22.     }
23.     printf("\n");
24.     fclose(fp);
25. }
```

注意 10-1-4

5. データの初期化
6. ファイルポインタ fp の宣言
7. 書き込みモードでオープンする
   出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
12. 10 進数 3 桁右詰めでファイル出力する
13. ファイルのクローズ
    書き込みと読み込みをはっきりさせるために一度クローズしている
14. 読み込みモードでオープンする
    入力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
20. 3 桁区切りでファイル入力する
24. ファイルのクローズを忘れないように

解答 10-1-5

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file5.dat";
5.      int i;
6.      float a[5]={123.4,44.6,-556.34,0,38.45},b;
7.      FILE *fp;
8.      fp=fopen(fname,"w");
9.      if(fp==NULL){
10.         printf("ファイルがオープン出来ません。\n");
11.         return;
12.     }
13.     for(i=0;i<5;i++) fprintf(fp,"%10.3f",a[i]);
14.     fclose(fp);
15.     fp=fopen(fname,"r");
16.     if(fp==NULL){
17.         printf("ファイルがオープン出来ません。\n");
18.         return;
19.     }
20.     for(i=0;i<5;i++){
21.         fscanf(fp,"%10f",&b);
22.         printf("%10.3f",b);
23.     }
24.     printf("\n");
25.     fclose(fp);
26. }
```

注意 10-1-5

6. データの初期化
7. ファイルポインタ fp の宣言
8. 書き込みモードでオープンする
   出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
13. %10.3f でファイル出力する
14. ファイルのクローズ
   書き込みと読み込みをはっきりさせるために一度クローズしている
15. 読み込みモードでオープンする
   入力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
21. 10 桁区切りでファイル入力する　"%10.3f" は不可
25. ファイルのクローズを忘れないように

解答 10-1-6

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>
4.  main()
5.  {
6.      char fname1[]="even.dat",fname2[]="odd.dat";
7.      int i,a;
8.      FILE *fp1,*fp2;
9.      fp1=fopen(fname1,"w");
10.     if(fp1==NULL){
11.         printf("ファイルがオープン出来ません。\n");
12.         return;
13.     }
14.     fp2=fopen(fname2,"w");
15.     if(fp2==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         fclose(fp1);
18.         return;
19.     }
20.     srand((unsigned int)time(NULL))
21.     for(i=0;i<100;i++){
22.         a=rand()%100+1;
23.         if(a%2==0) fprintf(fp1,"%d ",a);
24.         else fprintf(fp2,"%d ",a);
25.     }
26.     fclose(fp1);
```

```
27.     fclose(fp2);
28. }
```

注意 10-1-6

2. srand(),rand()関数のためのインクルードファイル
3. time()関数のためのインクルードファイル
9,14. 書き込みモードでオープンする
　　出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
17. fp1 はすでにオープンされているのでクローズする
23. 偶数なら fp1 に書き込む
24. 奇数なら fp2 に書き込む
26,27. ファイルのクローズを忘れないように

解答 10-1-7

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int a;
5.      FILE *fp;
6.      fp=fopen("file7.dat","w");
7.      if(fp==NULL){
8.          printf("ファイルがオープン出来ません。\n");
9.          return;
10.     }
11.     while(1){
12.         printf("data > ");
13.         scanf("%d",&a);
14.         if(a==-999) break;
15.         fprintf(fp,"%d ",a);
16.     }
17.     fclose(fp);
18. }
```

注意 10-1-7

6. 書き込みモードでオープンする
　ここでは直接ファイル名の文字列を用いている
14. もし-999 が入力されたらループから抜け出る
15. それ以外はファイルに出力する
17. ファイルのクローズを忘れないように
　ファイルを常にオープンしている状態はあまり安全ではない
　実際はメモリ上にデータを残して，なるべく一度に出力する

解答 10-1-8

```
 1. #include <stdio.h>
 2. main()
 3. {
 4.     char fname[30];
 5.     int c;
 6.     FILE *fp;
 7.     printf("表示するファイル名 > ");
 8.     scanf("%s",fname);
 9.     fp=fopen(fname,"rt");
10.     if(fp==NULL){
11.         printf("ファイルがオープン出来ません。\n");
12.         return;
13.     }
14.     while((c=getc(fp))!=EOF) putchar(c);
15.     fclose(fp);
16. }
```

注意 10-1-8

9. テキスト読み込みモードでオープン
   c の中で改行は 0xa となるが putchar()でもとの 0xd0xa に戻る
   バイナリ読み込みモードでも可で，この場合 c の中でも putchar()の出力も 0xd0xa である
14. fp から getc()関数で 1 文字ずつ読み込み
   putchar()関数で 1 文字ずつ表示する
   ファイルの終わりになると getc()関数は EOF を返す
   EOF は stdio.h の中で定義されている
15. ファイルのクローズを忘れないように

解答 10-1-9

```
 1. #include <stdio.h>
 2. #include <math.h>
 3. void setdata(char *),rdata(char *);

 4. main()
 5. {
 6.     char fname[]="file9.dat";
 7.     setdata(fname);
 8.     rdata(fname);
```

```
9.  }

10. void setdata(char *fname)
11. {
12.     int i,n=10,a[10]={4,6,8,3,4,6,7,4,9,3};
13.     FILE *fp;
14.     fp=fopen(fname,"wt");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     fprintf(fp,"%d\n",n);
20.     for(i=0;i<n-1;i++) fprintf(fp,"%d,",a[i]);
21.     fprintf(fp,"%d\n",a[n-1]);
22.     fclose(fp);
23. }

24. void rdata(char *fname)
25. {
26.     int i,no,a;
27.     double t1,t2,mean,var,dev;
28.     FILE *fp;
29.     fp=fopen(fname,"rt");
30.     if(fp==NULL){
31.         printf("ファイルがオープン出来ません。\n");
32.         return;
33.     }
34.     t1=t2=0.0;
35.     fscanf(fp,"%d",&no);
36.     for(i=0;i<no;i++){
37.         fscanf(fp,"%d,",&a);
38.         t1+=(double)a;
39.         t2+=(double)a*a;
40.     }
41.     fclose(fp);
42.     mean=t1/(double)no;
43.     var=t2/(double)no-mean*mean;
44.     dev=sqrt(var);
45.     printf("平均 = %f\n",mean);
46.     printf("分散 = %f\n",var);
47.     printf("標準偏差 = %f\n",dev);
```

```
48. }
```

注意 10-1-9

```
2.  sqrt()関数を使用するためのインクルードファイル
3.  使用する関数のプロトタイプ宣言
7.  データの書き込み
8.  データの読み出しと計算
10. 指定ファイルにデータを書き込む関数
14. 出力に'\n'があるのでテキスト書き込みモードでオープンする
19. データ数を書き込み改行する
20. 最後から 2 番目までは後ろにカンマを付ける
21. 最後のデータの後ろは改行する
22. ファイルのクローズを忘れないように
24. 指定ファイルのデータを読み込み統計量を計算する関数
29. 入力に'\n'があるのでテキスト読み込みモードでオープンする
34. t1 には合計，t2 には 2 乗の合計が代入される
35. データ数の値を読み込む
37. データをカンマ区切りで読み込む
    空白，タブ，リターンも区切り記号となる
38. キャスト演算子を用いて double 型でデータの合計を計算する
39. 同じく double 型でデータの 2 乗の合計を計算する
41. ファイルのクローズを忘れないように
42. 平均の計算
43. 分散の計算
44. 標準偏差の計算
```

解答 10-1-10

```
1.  #include <stdio.h>
2.  #include <math.h>

3.  void setdata(char *),rdata(char *);

4.  main()
5.  {
6.      char fname[]="file10.dat";
7.      setdata(fname);
8.      rdata(fname);
9.  }

10. void setdata(char *fname)
```

```
11. {
12.     int i,n=10,a[10]={4,6,8,3,4,6,7,4,9,3};
13.     FILE *fp;
14.     fp=fopen(fname,"w");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     for(i=0;i<n-1;i++) fprintf(fp,"%d,",a[i]);
20.     fprintf(fp,"%d\n",a[n-1]);
21.     fclose(fp);
22. }

23. void rdata(char *fname)
24. {
25.     int i,no,a;
26.     double t1,t2,mean,var,dev;
27.     FILE *fp;
28.     fp=fopen(fname,"r");
29.     if(fp==NULL){
30.         printf("ファイルがオープン出来ません。\n");
31.         return;
32.     }
33.     no=0;
34.     t1=t2=0.0;
35.     while(fscanf(fp,"%d,",&a)!=EOF){
36.         no++;
37.         t1+=(double)a;
38.         t2+=(double)a*a;
39.     }
40.     fclose(fp);
41.     mean=t1/(double)no;
42.     var=t2/(double)no-mean*mean;
43.     dev=sqrt(var);
44.     printf("平均 = %f\n",mean);
45.     printf("分散 = %f\n",var);
46.     printf("標準偏差 = %f\n",dev);
47. }
```

注意 10-1-10

2. sqrt()関数を使用するためのインクルードファイル

3. 使用する関数のプロトタイプ宣言
7. データの書き込み
8. データの読み出しと計算
10. 指定ファイルにデータを書き込む関数
14. 出力に'\n'があるのでテキスト書き込みモードでオープンする
19. 最後から 2 番目までは後ろにカンマを付ける
20. 最後のデータの後ろは改行する
21. ファイルのクローズを忘れないように
23. 指定ファイルのデータを読み込み統計量を計算する関数
28. 入力に'\n'があるのでテキスト読み込みモードでオープンする
33. データ数のカウンタを初期化する
34. t1 には合計，t2 には 2 乗の合計が代入される
35. データをカンマ区切りでファイルエンドまで読み込む
    空白，タブ，リターンも区切り記号となる
    fscanf()関数はファイルエンドに達すると EOF を返す
36. データを読み込むごとにカウンタを 1 つずつ増加させる
37. キャスト演算子を用いて double 型でデータの合計を計算する
38. 同じく double 型でデータの 2 乗の合計を計算する
40. ファイルのクローズを忘れないように
41. 平均の計算
42. 分散の計算
43. 標準偏差の計算

解答 10-1-11

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file11.dat";
5.      int i,a,p[]={90,59,32,87};
6.      FILE *fp;
7.      fp=fopen(fname,"w");
8.      if(fp==NULL){
9.          printf("ファイルがオープン出来ません。\n");
10.         return;
11.     }
12.     for(i=1;i<=100;i++) fprintf(fp,"%4d",i);
13.     fclose(fp);

14.     fp=fopen(fname,"r");
15.     if(fp==NULL){
```

```
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     for(i=0;i<4;i++){
20.         fseek(fp,(p[i]-1)*4,SEEK_SET);
21.         fscanf(fp,"%4d",&a);
22.         printf("%4d",a);
23.     }
24.     printf("\n");
25.     fclose(fp);
26. }
```

注意 10-1-11

5. 配列 p には読み込むデータの番号を入れる
7. 書き込みモードでオープンする
   出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
12. 4 桁右詰めで出力する
14. 読み込みモードでオープンする
   入力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
20. ファイル中の指定した位置へ読み込み位置を移す
   1 つのデータ 4 バイトで p[i]-1 番目の位置へ移動する
   SEEK_SET(0)はファイルの先頭からの意味である
21. 4 桁区切りの読み込み
22. 90　59　32　87 と表示される

解答 10-1-12

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file12.dat";
5.      int i,a,p[]={50,60};
6.      FILE *fp;
7.      fp=fopen(fname,"w");
8.      if(fp==NULL){
9.          printf("ファイルがオープン出来ません。\n");
10.         return;
11.     }
12.     for(i=0;i<100;i++) fprintf(fp,"%5d",i+1);
13.     fclose(fp);
```

```
14.     fp=fopen(fname,"r+");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     for(i=0;i<2;i++){
20.         fseek(fp,(p[i]-1)*5,SEEK_SET);
21.         fscanf(fp,"%5d",&a);
22.         fseek(fp,-5,SEEK_CUR);
23.         fprintf(fp,"%5d",-a);
24.     }
25.     fclose(fp);
26. }
```

注意 10-1-12

5. 配列 p には読み込むデータの番号を入れる
7. 書き込みモードでオープンする
   出力に'\n'も^Z もないのでテキスト，バイナリモードとも同じ
12. 100 個の整数乱数 5 桁右詰めの出力
14. 更新モードでのオープン
20. 1 つのデータ 5 バイトで p[i]-1 番目の位置へ移動する
    SEEK_SET(0)はファイルの先頭からの意味である
21. データを読み込む
22. ポインタを現在値より 5 バイト前に移す
    SEEK_CUR(1)はファイルの現在位置からの意味である
23. 符号を変えて上書きする

解答 10-1-13

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file13.dat";
5.      int i,a;
6.      FILE *fp;
7.      fp=fopen(fname,"wt");
8.      if(fp==NULL){
9.          printf("ファイルがオープン出来ません。\n");
10.         return;
11.     }
12.     for(i=1;i<=20;i++) fprintf(fp,"%4d\n",i);
```

```
13.     fclose(fp);

14.     fp=fopen(fname,"rt");
15.     if(fp==NULL){
16.         printf("ファイルがオープン出来ません。\n");
17.         return;
18.     }
19.     fseek(fp,9*6,SEEK_SET);
20.     fscanf(fp,"%5d",&a);
21.     printf("%5d\n",a);
22.     fclose(fp);
23. }
```

注意 10-1-13

7. 出力に'\n'があるのでテキスト書き込みモードでオープンする
12. 改行しながら 4 桁右詰めで出力する
14. テキスト書き込みモードでオープンする
19. ファイルの先頭から(10-1)*(4+2)バイト目へ移動する
    SEEK_SET(0)はファイルの先頭からの意味である
    *(4+2)の 2 は改行 CR･LF(0xd0xa)の 2 バイトである

解答 10-1-14

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="file14.dat";
5.      int i,a[100],b[100];
6.      FILE *fp;
7.      for(i=0;i<100;i++) a[i]=i+1;
8.      fp=fopen(fname,"wb");
9.      if(fp==NULL){
10.         printf("ファイルがオープン出来ません。\n");
11.         return;
12.     }
13.     fwrite((int *)a,sizeof(int),100,fp);
14.     fclose(fp);

15.     fp=fopen(fname,"rb");
16.     if(fp==NULL){
17.         printf("ファイルがオープン出来ません。\n");
```

```
18.         return;
19.     }
20.     fread((int *)b,sizeof(int),100,fp);
21.     fclose(fp);
22.     for(i=0;i<100;i++) printf("%4d",b[i]);
23.     printf("\n");
24. }
```

注意 10-1-14

7. 配列にデータを代入
8. バイナリ書き込みモードでオープンする
   0xa,0x1a の入る可能性のあるデータを変換せず書き出すのでバイナリモードは不可欠である
13. int 型のデータをポインタ a から 100 個バイナリ出力する
15. バイナリ読み込みモードでオープンする
   0xa,0x1a の入る可能性のあるデータを変換せず読み込むのでバイナリモードは不可欠である
20. int 型のデータをポインタ b に 100 個バイナリ入力する
22. 読み込んだデータを表示する

解答 10-1-15

```
1.  #include <stdio.h>
2.  void setdata(char *),rdata(char *);
3.  struct dbase{
4.      char name[30];
5.      int age;
6.      float hight,weight;
7.  };

8.  main()
9.  {
10.     char fname[]="file15.dat";
11.     setdata(fname);
12.     rdata(fname);
13. }

14. void setdata(char *fname)
15. {
16.     int i;
17.     float x;
```

```
18.     struct dbase a[3];
19.     FILE *fp;
20.     for(i=0;i<3;i++){
21.         printf("氏名 > ");
22.         scanf("%s",a[i].name);
23.         printf("年齢 > ");
24.         scanf("%d",&a[i].age);
25.         printf("身長 > ");
26.         scanf("%f",&x);
27.         a[i].hight=x;
28.         printf("体重 > ");
29.         scanf("%f",&x);
30.         a[i].weight=x;
31.     }
32.     fp=fopen(fname,"wb");
33.     if(fp==NULL){
34.         printf("ファイルがオープン出来ません。\n");
35.         return;
36.     }
37.     fwrite(a,sizeof(struct dbase),3,fp);
38.     fclose(fp);
39. }

40. void rdata(char *fname)
41. {
42.     int i;
43.     struct dbase a[3];
44.     FILE *fp;
45.     fp=fopen(fname,"rb");
46.     if(fp==NULL){
47.         printf("ファイルがオープン出来ません。\n");
48.         return;
49.     }
50.     fread(a,sizeof(struct dbase),3,fp);
51.     fclose(fp);
52.     for(i=0;i<3;i++){
53.         printf("No. %d\n",i+1);
54.         printf("氏名 . %s\n",a[i].name);
55.         printf("年齢 . %d\n",a[i].age);
56.         printf("身長 . %.1f\n",a[i].hight);
57.         printf("体重 . %.1f\n",a[i].weight);
```

```
58.     }
59. }
```

注意 10-1-15

2. 使用する関数のプロトタイプ宣言
3-7. データを代入する構造体の宣言
20-31. 構造体へのデータの代入
32. バイナリ書き込みモードでオープンする
    0xa,0x1a の入る可能性のあるデータを変換せず書き出すのでバイナリモードは不可欠である
32. a から構造体 dbase のサイズ 3 個分のデータをファイルに書き込む
45. バイナリ読み込みモードでオープンする
    0xa,0x1a の入る可能性のあるデータを変換せず読み込むのでバイナリモードは不可欠である
50. ファイルから構造体 dbase のサイズ 3 個分のデータを a に読み込む
52-58. 3 個分のデータを表示する

解答 10-1-16

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      char fname[]="aki.dat";
5.      FILE *fp;
6.      fp=fopen(fname,"wt");
7.      if(fp==NULL){
8.          printf("ファイルがオープン出来ません。\n");
9.          return;
10.     }
11.     fprintf(fp,"嵐山の紅葉\n%c",0x1a);
12.     fclose(fp);
13.     fp=fopen(fname,"r+t");
14.     if(fp==NULL){
15.         printf("ファイルがオープン出来ません。\n");
16.         return;
17.     }
18.     fseek(fp,-1,SEEK_END);
19.     fprintf(fp,"保津川下り\n%c",0x1a);
20.     fclose(fp);
21. }
```

注意 10-1-16

7. 出力に'\n'があるのでテキスト書き込みモードでオープンする
11. 改行後^Z(0x1a)を書き込む
    エディタの中にはファイルの最後に^Z(0x1a)を付けるものもある
13. テキスト更新モードでオープンする
    ファイルは既に存在していること
18. 入出力の現在位置をファイルの最後から 1 バイト前に移す
    すなわち^Z(0x1a)に上書きするようにする
    追加モードでオープンしても^Z(0x1a)はそのまま残る

解答 10-1-17

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int i;
5.      char *str[]={"きんかん","みかん","夏みかん"};
6.      for(i=0;i<3;i++) fprintf(stdprn,"%s\x0d\x0a",str[i]);
7.  }
```

注意 10-1-17

5. プリンタのファイルポインタは stdprn と定義されている
6. プリンタのオープンモードはバイナリであるから'\n'は 0xa しか送らない

解答 10-1-18

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      fprintf(stdprn,"\x0e すいか\x0f\x0d\x0a");
5.  }
```

注意 10-1-18

5. 例えば PC-PR プリンタの制御コードでは E_H は拡大指定, F_H は拡大解除
   プリンタによって制御コードは異るので各自のプリンタで調べること

10-2) 低水準入出力

解答 10-2-1

```
1.  #include <stdio.h>
```

```
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      int i,fl,a[100];
8.      for(i=0;i<100;i++) a[i]=i+1;
9.      fl=open("bfile1.dat",O_WRONLY|O_CREAT|O_BINARY,
10.                                  S_IREAD|S_IWRITE);
11.     if(fl==-1){
12.         printf("ファイルがオープン出来ません。\n");
13.         return;
14.     }
15.     write(fl,a,sizeof(int)*100);
16.     close(fl);
17. }
```

注意 10-2-1

2. open(),write()関数を使用するためのインクルードファイル
3,4. open()関数を使用するためのインクルードファイル
8. データの代入
9,10. 書き込み，作成，バイナリモードで読み書き属性ファイルとして
      オープンする
      int 型の戻値 fl をファイルハンドルと呼ぶ
11. open()関数はオープンに失敗した場合は-1 を返す
15. a から sizeof(int)*100 バイトファイル fl に出力する。
16. ファイルのクローズ

解答 10-2-2

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      int i,fl,a[100];
8.      fl=open("bfile1.dat",O_RDONLY|O_BINARY,
9.                                  S_IREAD|S_IWRITE);
10.     if(fl==-1){
11.         printf("ファイルがオープン出来ません。\n");
```

```
12.         return;
13.     }
14.     read(fl,a,sizeof(int)*100);
15.     close(fl);
16.     for(i=0;i<100;i++) printf("%4d",a[i]);
17.     printf("\n");
18. }
```

注意 10-2-2

2. open(),read()関数を使用するためのインクルードファイル
3,4. open()関数を使用するためのインクルードファイル
8,9. 読み込み，作成，バイナリモードで読み書き属性ファイルとして
オープンする
10. open()関数はオープンに失敗した場合は-1 を返す
14. ファイル fl から sizeof(int)*100 バイト a に出力する。
15. ファイルのクローズ
16. 読み込んだデータの確認

解答 10-2-3

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      int i,fl;
8.      double a[100];
9.      for(i=0;i<100;i++) a[i]=(double)(i+1)/100.0;
10.     fl=open("bfile3.dat",O_WRONLY|O_CREAT|O_BINARY,
11.                                 S_IREAD|S_IWRITE);
12.     if(fl==-1){
13.         printf("ファイルがオープン出来ません。\n");
14.         return;
15.     }
16.     write(fl,a,sizeof(double)*100);
17.     close(fl);
18. }
```

注意 10-2-3

10,11. 書き込み，作成，バイナリモードで読み書き属性ファイルとして
オープンする

12. open()関数はオープンに失敗した場合は-1 を返す
16. ファイル fl から sizeof(double)*100 バイト a に出力する。

解答 10-2-4

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      int i,fl;
8.      double a[100];
9.      fl=open("bfile3.dat",O_RDONLY|O_BINARY,
10.                              S_IREAD|S_IWRITE);
11.     if(fl==-1){
12.         printf("ファイルがオープン出来ません。\n");
13.         return;
14.     }
15.     read(fl,a,sizeof(double)*100);
16.     close(fl);
17.     for(i=0;i<100;i++) printf("%10.5f",a[i]);
18. }
```

注意 10-2-4

2. open(),read()関数を使用するためのインクルードファイル
3,4. open()関数を使用するためのインクルードファイル
9,10. 読み込み, 作成, バイナリモードで読み書き属性ファイルとして
オープンする
11. open()関数はオープンに失敗した場合-1 を返す
15. ファイル fl から sizeof(double)*100 バイト a に出力する。
16. ファイルのクローズ
17. 読み込んだデータの確認

解答 10-2-5

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
```

```
7.      int i,fl,a,b[]={40,50,60};
8.      fl=open("bfile1.dat",O_RDONLY|O_BINARY,
9.                              S_IREAD|S_IWRITE);
10.     if(fl==-1){
11.         printf("ファイルがオープン出来ません。\n");
12.         return;
13.     }
14.     for(i=0;i<3;i++){
15.         lseek(fl,sizeof(int)*(b[i]-1),SEEK_SET);
16.         read(fl,&a,sizeof(int));
17.         printf("%d\n",a);
18.     }
19.     close(fl);
20. }
```

注意 10-2-5

2. open(),read(),lseek()関数を使用するためのインクルードファイル

8,9. 読み込み，作成，バイナリモードで読み書き属性ファイルとして
オープンする

15. 読み込み位置をファイルの先頭から sizeof(int)*(b[i]-1)の位置にする
b[i]-1 は 39,49,59 となる
SEEK_SET(0)はファイルの先頭からという意味になる

16. fl から sizeof(int)バイト分 a へ読み込む

17. 読み込んだデータの表示

解答 10-2-6

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      int i,fl,a,b[100];
8.      fl=open("bfile1.dat",O_RDWR|O_BINARY,S_IREAD|S_IWRITE);
9.      if(fl==-1){
10.         printf("ファイルがオープン出来ません。\n");
11.         return;
12.     }
13.     lseek(fl,sizeof(int)*29,SEEK_SET);
```

```
14.     a=0;
15.     write(fl,&a,sizeof(int));
16.     lseek(fl,0,SEEK_SET);
17.     read(fl,b,sizeof(int)*100);
18.     close(fl);
19.     for(i=0;i<100;i++) printf("%4d",b[i]);
20. }
```

注意 10-2-6

8. 更新, バイナリモードで読み書き属性ファイルとしてオープンする
13. 先頭から 30 番目のデータの位置へ書き込み位置を移動させる
14. 予め a を 0 にしておいて
15. a の位置から sizeof(int)バイトファイルに書き込む
16. 読み込み位置をファイルの先頭に移して
17. b の位置に sizeof(int)*100 バイトファイルから読み込む

解答 10-2-7

```
1.  #include <stdio.h>
2.  #include <io.h>
3.  #include <fcntl.h>
4.  #include <sys\stat.h>
5.  main()
6.  {
7.      char c,fname1[30],fname2[30];
8.      int fl1,fl2,chk;
9.      printf("コピー元 > ");
10.     scanf("%s",fname1);
11.     printf("コピー先 > ");
12.     scanf("%s",fname2);
13.     fl1=open(fname1,O_RDONLY|O_BINARY,S_IREAD|S_IWRITE);
14.     if(fl1==-1){
15.         printf("ファイルがオープン出来ません。\n");
16.         return;
17.     }
18.     fl2=open(fname2,O_WRONLY|O_CREAT|O_BINARY,
19.                             S_IREAD|S_IWRITE);
20.     if(fl2==-1){
21.         printf("ファイルがオープン出来ません。\n");
22.         return;
23.     }
```

```
24.     while(1){
25.         chk=read(fl1,&c,1);
26.         if(chk==0) break;
27.         write(fl2,&c,1);
28.     }
29.     close(fl1);
30.     close(fl2);
31. }
```

注意 10-2-7

13. 読み込み，バイナリモードで読み書き属性ファイルとしてオープンする
18,19. 書き込み，作成，バイナリモードで読み書き属性ファイルとしてオープンする
25. fl1 から c の位置へ 1 バイト読み込む
26. ファイルエンドの場合 chk に 1 が返る
正常終了の場合はリード出来たバイト数，エラーの場合は-1 が返る
27. c の位置から fl2 へ 1 バイト書き込む

１１章 再帰処理}

11-1) 再帰処理

解答 11-1-1

```
 1. #include <stdio.h>
 2. double fact(int);
 3.
 4. double fact(int no)
 5. {
 6.     if(no==0) return 1.0;
 7.     return (double)no*fact(no-1);
 8. }
 9.
10. main()
11. {
12.     int n;
13.     double ans;
14.     printf("階乗の計算 n > ");
15.     scanf("%d",&n);
16.     ans=fact(n);
17.     printf("%d! = %.0f\n",n,ans);
18. }
```

注意 11-1-1

2. 関数のプロトタイプ宣言
   このように関数定義が呼出し位置の前にある場合は特に必要ない
4. 結果は大きな数になるので戻値は double 型にしている
   引数はあまり大きな数を扱わないので int 型にしている
6. 0!=1
7. n×(n-1)!を計算して戻値にする
17. 値は整数なので小数点以下は表示しない

解答 11-1-2

```
 1. #include <stdio.h>
 2. double prog(int);
 3.
 4. double prog(int no)
 5. {
```

```
 6.     if(no==1) return 2.0;
 7.     return 3.0*prog(no-1)+2.0;
 8. }
 9.
10. main()
11. {
12.     int n;
13.     double ans;
14.     printf("数列の計算 n > ");
15.     scanf("%d",&n);
16.     ans=prog(n);
17.     printf("a%d = %.0f\n",n,ans);
18. }
```

注意 11-1-2

2. 関数のプロトタイプ宣言
   このように関数定義が呼出し位置の前にある場合は特に必要ない
4. 結果は大きな数になるので戻値は double 型にしている
6. a_1=2
7. $3a_{n-1}$+2 を計算して戻値にする
17. 値は整数なので小数点以下は表示しない

解答 11-1-3

```
 1. #include <stdio.h>
 2. double prog2(int);
 3.
 4. double prog2(int no)
 5. {
 6.     double x;
 7.     if(no==1) return 1.0;
 8.     if(no==2) return 2.0;
 9.     x=2.0*prog2(no-1)+prog2(no-2)+1.0;
10.     return x;
11. }
12.
13. main()
14. {
15.     int n;
16.     double x;
17.     printf("数列の計算 n > ");
```

```
18.     scanf("%d",&n);
19.     x=prog2(n);
20.     printf("a%d = %.0f\n",n,x);
21. }
```

注意 11-1-3

2. 関数のプロトタイプ宣言
   このように関数定義が呼出し位置の前にある場合は特に必要ない
7. a_1=1
8. a_2=2
9. $2a_{n-1}+a_{n-2}+1$ を計算して戻値にする
20. 値は整数なので小数点以下は表示しない

解答 11-1-4

```
1. #include <stdio.h>
2. void soinsu(long);
3.
4. void soinsu(long n)
5. {
6.     long a=2;
7.     while(a<n){
8.         if(n%a==0){
9.             printf("%ld*",a);
10.            soinsu(n/a);
11.            return;
12.        }
13.        a++;
14.    }
15.    printf("%ld\n",n);
16. }
17.
18. main()
19. {
20.    long n;
21.    printf("素因数分解 n > ");
22.    scanf("%ld",&n);
23.    soinsu(n);
24. }
```

注意 11-1-4

2. 関数のプロトタイプ宣言

このように関数定義が呼出し位置の前にある場合は特に必要ない
7. n が a で割り切れるまで a を増加しながら繰り返す
8. 割り切れた場合には
9. 割った数を表示し * 記号を付ける
10. 商を引数として再帰処理を行う
15. これ以上割り切れない場合(素数)は値を書いて改行する

解答 11-1-5

```
1. #include <stdio.h>
2. void hanoi(int,char,char,char);
3.
4. void hanoi(int n,char a,char b,char c)
5. {
6.     if(n==1) printf("円盤 %d を %c から %c へ\n",n,a,b);
7.     else{
8.         hanoi(n-1,a,c,b);
9.         printf("円盤 %d を %c から %c へ\n",n,a,b);
10.        hanoi(n-1,c,b,a);
11.    }
12. }
13.
14. main()
15. {
16.    int n;
17.    printf("円盤の数 n > ");
18.    scanf("%d",&n);
19.    printf("A の柱から B の柱へ円盤を動かします。\n");
20.    hanoi(n,'A','B','C');
21. }
```

注意 11-1-5

2. 関数のプロトタイプ宣言
このように関数定義が呼出し位置の前にある場合は特に必要ない
7. n(>1)枚の円盤を a から c を経由して b へ移すには
8. n-1 枚の円盤を a から b を経由して c へ移す
9. n 枚目の円盤を a から b へ移す
10. n-1 枚の円盤を c から a を経由して b へ移す

解答 11-1-6

```
1. #include <stdio.h>
```

```
 2. #include <string.h>
 3. #include <dir.h>
 4. #include <dos.h>
 5. void rep_dir(char *);
 6.
 7. void rep_dir(char *dir)
 8. {
 9.     char path[80],nextdir[80];
10.     int chk;
11.     struct ffblk fb;
12.     strcpy(path,dir);
13.     strcat(path,"*.*");
14.     chk=findfirst(path,&fb,FA_DIREC);
15.     while(chk==0){
16.         if(*fb.ff_name!='.'&&(fb.ff_attrib&FA_DIREC)!=0){
17.             strcpy(nextdir,dir);
18.             strcat(nextdir,fb.ff_name);
19.             printf("%s\n",nextdir);
20.             strcat(nextdir,"\\");
21.             rep_dir(nextdir);
22.         }
23.         chk=findnext(&fb);
24.     }
25. }
26.
27. main()
28. {
29.     char drv[80];
30.     printf("ドライブ名(A.etc.) > ");
31.     scanf("%s",&drv);
32.     strupr(drv);
33.     strcat(drv,"\\");
34.     rep_dir(drv);
35. }
```

注意 11-1-6

2. strcpy(),strcat(),strupr()関数のためのインクルードファイル
3. findfirst(),findnext()関数のためのインクルードファイル
4. findfirst()関数のためのインクルードファイル
5. 関数のプロトタイプ宣言
   このように関数定義が呼出し位置の前にある場合は特に必要ない

```
 7. 引数より下位のディレクトリを検索する関数
    後ろに\記号の付いたディレクトリ名を引数とする
 9. path[]には引数の後ろに*.*を付けた検索ファイル名を入れる
    nextdir[]には見つかった下位のディレクトリ名を入れて再帰処理
    する
11. 検索されたファイル(ディレクトリも含む)の情報が格納される構造体
14. ディレクトリ名に*.*を加えたパス名でファイルを検索する
    FA_DIREC を指定してもすべてのファイルが検索される
15. ファイルが見つかった場合
16. ディレクトリが自分自身"."または親".."でない場合
    かつ，ファイルがディレクトリでない場合
18. 見つかったディレクトリ名を元のディレクトリ名の後ろに繋いで
19. 新しいディレクトリ名を表示して
20. 後ろに \ 記号を付けて
23. さらに下位のディレクトリを検索するために再帰処理する
31. 入力は A.のようなドライブ名だけでもよいし，
    A.\TC のようにして下位ディレクトリを検索してもよい
33. ディレクトリ名の後ろに \ を付けておく
```

１２章 メモリ管理とデータ構造

12-1) メモリ管理

解答 12-1-1

```
 1. #include <stdio.h>
 2. #include <stdlib.h>
 3. #include <string.h>
 4. main()
 5. {
 6.     char *str;
 7.     str=(char *)malloc(10);
 8.     if(str==NULL){
 9.         printf("メモリが確保出来ません。\n");
10.         return;
11.     }
12.     strcpy(str,"Turbo C");
13.     printf("%s\n",str);
14.     free(str);
15. }
```

注意 12-1-1

2. malloc(),free()関数のためのインクルードファイル
3. strcpy()関数のためのインクルードファイル
7. メモリを 10 バイト確保して先頭ポインタを str に代入する
   メモリが確保できない場合は NUL を返す
12. "Turbo C"の文字列を str へ代入
14. 確保したメモリの解放

解答 12-1-2

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. main()
4. {
5.     int *a,i,n=100;
6.     a=(int *)malloc(sizeof(int)*n);
7.     if(a==NULL){
8.         printf("メモリが確保出来ません。\n");
9.         return;
```

```
10.     }
11.     for(i=0;i<n;i++) a[i]=i+1;
12.     for(i=0;i<n;i++) printf("%4d",a[i]);
13.     free(a);
14. }
```

注意 12-1-2

2. malloc(),free()関数のためのインクルードファイル
6. sizeof(int)×100 バイトの int 型メモリを確保する
   メモリが確保出来ない場合は NULL を返す
   a=(int *)calloc(n,sizeof(int)); も可能
11. 配列の要素に 1～100 までの数を代入する
13. 確保したメモリの解放

解答 12-1-3

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  main()
4.  {
5.      int i,j,m,n,*mat;
6.      printf("行数 > ");
7.      scanf("%d",&m);
8.      printf("列数 > ");
9.      scanf("%d",&n);
10.     mat=(int *)malloc(sizeof(int)*m*n);
11.     if(mat==NULL){
12.         printf("メモリが確保出来ません。\n");
13.         return;
14.     }
15.     for(i=0;i<m;i++){
16.         for(j=0;j<n;j++)
17.             mat[i*n+j]=i*n+j+1;
18.     }
19.     for(i=0;i<m;i++){
20.         for(j=0;j<n;j++)
21.             printf("%4d",mat[i*n+j]);
22.         printf("\n");
23.     }
24.     free(mat);
25. }
```

注意 12-1-3

2. malloc(),free()関数のためのインクルードファイル
10. int 型データ m 行 n 列分の領域確保
17. i+1 行 j+1 列目の要素は mat[i*n+j]である
24. メモリの解放

解答 12-1-4

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  main()
4.  {
5.      int i,n=100;
6.      double *vec;
7.      vec=(double *)malloc(sizeof(double)*n);
8.      if(vec==NULL){
9.          printf("メモリが確保出来ません。\n");
10.         return;
11.     }
12.     for(i=0;i<n;i++) vec[i]=(double)(i+1)/100.0;
13.     for(i=0;i<n;i++) printf("%8.4f",vec[i]);
14.     free(vec);
15. }
```

注意 12-1-4

2. malloc(),free()関数のためのインクルードファイル
7. double 型データ 100 個分のメモリ確保
14. メモリの解放

解答 12-1-5

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <string.h>
4.  main()
5.  {
6.      char *str;
7.      int i;
8.      str=(char *)malloc(10*5);
9.      if(str==NULL){
10.         printf("メモリが確保出来ません。\n");
```

```
11.         return;
12.     }
13.     for(i=0;i<5;i++) strcpy(str+i*10,"京都");
14.     for(i=0;i<5;i++) printf("%s\n",str+i*10);
15.     free(str);
16. }
```

注意 12-1-5

8. 10 バイトデータ 5 個分のメモリ確保
   他の例として構造体としてメモリ確保すると配列成分が使える
13,14. i+1 番目の文字列が始まるポインタは str+i*10 である
15. メモリの解放

解答 12-1-6

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <string.h>
4.  main()
5.  {
6.      char *str[5];
7.      int i,j;
8.      for(i=0;i<5;i++){
9.          str[i]=(char *)malloc(10);
10.         if(str[i]==NULL){
11.             printf("メモリが確保出来ません。\n");
12.             for(j=0;j<i;j++) free(str[i]);
13.             return;
14.         }
15.         strcpy(str[i],"大阪");
16.     }
17.     for(i=0;i<5;i++) printf("%s\n",str[i]);
18.     for(i=0;i<5;i++) free(str[i]);
19. }
```

注意 12-1-6

6. 5 個分のポインタ配列の宣言
9. ポインタ配列の各成分に対して 10 バイトのメモリを確保する
10. 1 箇所でメモリが確保出来ない場合
12. これまで確保したメモリをすべて解放して終了させる
15. 確保した領域へデータを代入する
18. メモリの解放

解答 12-1-7

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  main()
4.  {
5.      struct name{
6.          char name[20];
7.          int age;
8.      } *ptr;
9.      int i;
10.     ptr=(struct name *)malloc(sizeof(struct name)*5);
11.     if(ptr==NULL){
12.         printf("メモリが確保出来ません。\n");
13.         return;
14.     }
15.     for(i=0;i<5;i++){
16.         printf("名前 > ");
17.         scanf("%s",ptr[i].name);
18.         printf("年齢 > ");
19.         scanf("%d",&ptr[i].age);
20.     }
21.     printf("%20s%3d\n",ptr[2].name,ptr[2].age);
22.     free(ptr);
23. }
```

注意 12-1-7

5-8. 構造体の定義と構造体ポインタの宣言
10. name 型構造体 5 個分のメモリ確保
17. (ptr+i)->name も可能
19. &(ptr+i)->age も可能
21. (ptr+2)->name,(ptr+2)->age も可能
22. メモリの解放

12-2) データ構造

解答 12-2-1

```
1.  #include <stdio.h>
2.  typedef struct numlist{
3.      struct numlist *prev;
```

```
4.      int data;
5.      struct numlist *next;
6.  } DATA;
7.  DATA *makelist(int);
8.  void displist(DATA *), freelist(DATA *);

9.  main()
10. {
11.     int i,n;
12.     DATA *p,*first;
13.     first=makelist(100);
14.     n=0;
15.     p=first;
16.     while(p!=NULL){
17.         p->data=n++;
18.         p=p->next;
19.     }
20.     displist(first);
21.     freelist(first);
22. }

23. DATA *makelist(int no)
24. {
25.     int i;
26.     DATA *p,*pf,*pl;
27.     for(i=0;i<no;i++){
28.         p=(DATA *)malloc(sizeof(DATA));
29.         if(i==0){
30.             p->prev=NULL;
31.             p->next=NULL;
32.             pf=p;
33.             pl=p;
34.         }
35.         else{
36.             pl->next=p;
37.             p->prev=pl;
38.             p->next=NULL;
39.             pl=p;
40.         }
41.         p->data=0;
42.     }
```

```
43.     return pf;
44. }

45. void displist(DATA *fst)
46. {
47.     DATA *p;
48.     p=fst;
49.     while(p!=NULL){
50.         printf("%4d",p->data);
51.         p=p->next;
52.     }
53.     printf("\n\n");
54. }

55. void freelist(DATA *fst)
56. {
57.     DATA *p,*pn;
58.     p=fst;
59.     while(p!=NULL){
60.         pn=p->next;
61.         free(p);
62.         p=pn;
63.     }
64. }
```

注意 12-2-1

2-6. リスト型構造体 numlist を DATA 型と定義する
7. makelist()関数のプロトタイプ宣言
8. displist(),freelist()関数のプロトタイプ宣言
13. 100 個分のデータをリストに繋いだメモリを確保する
最初に first を先頭要素へのポインタとする
17. n の値を 1 ずつ増加させながら各要素の data へのポインタとする
18. p を次の要素へのポインタとする
20. 先頭要素から順番に data を表示する
21. メモリをすべて解放する
23. no の数だけリストを繋いで構造体中の data を 0 で初期化する関数
リストの先頭要素へのポインタを戻値にする
26. pf は先頭要素へのポインタ
pl は最終要素へのポインタ
29-40. もう少し短く出来るが分かり易さを優先した
29-34. 先頭要素における処理

30. 先頭要素中の prev は NULL にしておく
38. 最終要素中の next は NULL にしておく
41. 構造体中の data 変数を 0 で初期化する
43. 先頭要素へのポインタを戻値とする
45. リストの各要素中の data の値を表示する関数
48. 最初に先頭要素へのポインタを p とする
49. p の値が NULL になるまで表示を続ける
50. データの表示
51. p->next を新しく p とする
55. 確保したリスト構造のメモリを順番に解放する関数
59. p の値が NULL になるまで解放を実行する
60. pn を次の要素へのポインタとする
61. メモリの解放
62. 新しく次の要素へのポインタを p とする

解答 12-2-2

*A. [12-2-1]の 1-8 行を挿入する。

```
1.   DATA *dlist(DATA *);
2.  main()
3.  {
4.      int i,n,no;
```

*B. [12-2-1]の 12-19 行を挿入する。

```
5.      printf("消去する要素番号 > ");
6.      scanf("%d",&no);
7.      p=first;
8.      for(i=0;i<no;i++) p=p->next;
9.      p=dlist(p);
10.     displist(first);
11.     freelist(first);
12. }
```

*C. [12-2-1]の 23-64 行を挿入する。

```
13. DATA *dlist(DATA *ptr)
14. {
15.     DATA *p,*pp,*pn;
16.     if(ptr==NULL) return NULL;
17.     pp=ptr->prev;
18.     pn=ptr->next;
19.     if(pp==NULL && pn==NULL){
20.         free(ptr);
```

```
21.         p=NULL;
22.     }
23.     else if(pp==NULL){
24.         pn->prev=NULL;
25.         free(ptr);
26.         p=pn;
27.     }
28.     else if(pn==NULL){
29.         pp->next=NULL;
30.         free(ptr);
31.         p=pp;
32.     }
33.     else {
34.         pp->next=pn;
35.         pn->prev=pp;
36.         free(ptr);
37.         p=pn;
38.     }
39.     return p;
40. }
```

注意 12-2-2

*A. DATA 型の定義他
*B. リストの作成
7,8. 指定要素の探索
　　no の値が不適切な場合の処理は簡単のため考えていない
10. 正しく解放されたかどうか表示してみる
11. ポインタ first で指定された要素の解放
*C. makelist(),displist(),freelist()関数の定義
17. 指定要素の prev を pp とし,
18. 指定要素の next を pn とする
19. pp と pn が共に NULL の場合, 即ち要素が 1 個だけの場合
20. その要素のメモリを解放して
21. p に NULL を返す
23. pp だけが NULL の場合, 即ち先頭要素の場合
24. 次の要素の prev を NULL にして
25. その要素のメモリを解放して
26. 次の要素へのポインタを p に返す
28. pn だけが NULL の場合, 即ち最終要素の場合
29. 前の要素の next を NULL にして
30. その要素のメモリを解放して

```
31. 前の要素へのポインタを p に返す
33. その他前後に要素のある場合
34. 前の要素の next を次の要素へのポインタにし
35. 次の要素の prev を前の要素へのポインタにし
36. その要素のメモリを解放して
37. 次の要素へのポインタを p に返す
39. 戻値をポインタ p とする
```

解答 12-2-3

```
*A. [12-2-1]の 1-8 行を挿入する。
1.  DATA *ilist(DATA *);
2.  main()
3.  {
4.      int i,n,no;
*B. [12-2-1]の 12-19 行を挿入する。
5.      printf("挿入する要素番号 > ");
6.      scanf("%d",&no);
7.      p=first;
8.      for(i=0;i<no;i++) p=p->next;
9.      p=ilist(p);
10.     displist(first);
11.     freelist(first);
12. }
*C. [12-2-1]の 23-64 行を挿入する。

13. DATA *ilist(DATA *ptr)
14. {
15.     DATA *p,*pp,*pn;
16.     p=(DATA *)malloc(sizeof(DATA));
17.     if(p==NULL) return NULL;
18.     p->data=0;
19.     if(ptr==NULL){
20.         p->prev=NULL;
21.         p->next=NULL;
22.         return p;
23.     }
24.     pp=ptr->prev;
25.     if(pp==NULL){
26.         p->prev=NULL;
27.         p->next=ptr;
```

```
28.         ptr->prev=p;
29.     }
30.     else{
31.         pp->next=p;
32.         p->prev=pp;
33.         p->next=ptr;
34.         ptr->prev=p;
35.     }
36.     return p;
37. }
```

注意 12-2-3

*A. DATA 型の定義他
 1. 関数のプロトタイプ宣言
*B. リストの作成
7,8. 指定要素の探索
     no の値が不適切な場合の処理は簡単のため考えていない
*C. makelist(),displist(),freelist()関数の定義
13. ptr で指定された要素の前に 1 つ要素を挿入する関数
16. 要素 1 個分のメモリ確保
18. 作られた要素中の data を 0 で初期化する
19. リストが何もない状態でこの関数が呼ばれた場合
20,21. prev と next を NULL として
22. p を戻値にする
24. ptr の前の要素へのポインタを調べる
25. ptr が先頭要素の場合
26. 新しい要素の prev を NULL とし
27. 新しい要素の next を ptr とし
28. ptr の prev を新しい要素へのポインタとする
30. 先頭要素でない場合
31. ptr の前の要素の next を新しい要素へのポインタとし
32. 新しい要素の prev を ptr の前の要素へのポインタとし
33. 新しい要素の next を ptr とし
34. ptr の prev を新しい要素へのポインタとする
36. 新しい要素へのポインタ p を戻値とする

解答 12-2-4

*A. [12-2-1]の 1-8 行を挿入する。
1. DATA *alist(DATA *);
2. main()

```
3.  {
*B. [12-2-1]の 12-19 行を挿入する。
4.      alist(first);
5.      displist(first);
6.      freelist(first);
7.  }
*C. [12-2-1]の 23-64 行を挿入する。

8.  DATA *alist(DATA *ptr)
9.  {
10.     DATA *p,*pn;
11.     p=(DATA *)malloc(sizeof(DATA));
12.     if(p==NULL) return NULL;
13.     p->data=0;
14.     if(ptr==NULL){
15.         p->prev=NULL;
16.         p->next=NULL;
17.         return p;
18.     }
19.     pn=ptr;
20.     while(pn->next!=NULL) pn=pn->next;
21.     pn->next=p;
22.     p->prev=pn;
23.     p->next=NULL;
24.     return p;
25. }
```

注意 12-2-4

*A. DATA 型の定義他
1. 関数のプロトタイプ宣言
*C. makelist(),displist(),freelist()関数の定義
8. ptr で指定された要素からリストの最後を求め,要素を 1 つ追加する関数
11. 要素 1 個分のメモリ確保
12. ポインタが NULL の場合
15-17. 要素が 1 つだけのリストを作る
19,20. 最終要素へのポインタ pn を求める
21-22. pn の後ろに新しい要素を追加する

解答 12-2-5

```
 *A. [12-2-1]の 1-8 行を挿入する。
 1.  DATA *dellist(DATA *,int,int);
 2.  main()
 3.  {
 4.      int n1,n2;
*B. [12-2-1]の 12-19 行を挿入する。
 5.      printf("消去を始める要素番号 > ");
 6.      scanf("%d",&n1);
 7.      printf("消去する個数 > ");
 8.      scanf("%d",&n2);
 9.      first=dellist(first,n1,n2);
 10.     displist(first);
 11.     freelist(first);
 12. }
*C. [12-2-1]の 23-64 行を挿入する。

 13. DATA *dellist(DATA *fst,int st,int no)
 14. {
 15.     int i;
 16.     DATA *p,*pp,*pn;
 17.     p=fst;
 18.     for(i=0;i<st;i++){
 19.         p=p->next;
 20.         if(p==NULL) return NULL;
 21.     }
 22.     pp=p->prev;
 23.     for(i=0;i<no;i++){
 24.         pn=p->next;
 25.         free(p);
 26.         p=pn;
 27.         if(p==NULL) break;
 28.     }
 29.     if(pp==NULL){
 30.         if(p==NULL) return NULL;
 31.         else{
 32.             p->prev=NULL;
 33.             return p;
 34.         }
 35.     }
```

```
    else{
        if(p==NULL){
            pp->next=NULL;
            return fst;
        }
        else{
            pp->next=p;
            p->prev=pp;
            return fst;
        }
    }
}
```

注意 12-2-5

*A. DATA 型の定義他
9. first の n1 番目の要素から n2 個の要素を消去する
*C. makelist(),displist(),freelist()関数の定義
13. fst の st 番目の要素から no 個の要素を取り除く関数
17-21. fst から st 番目の要素へのポインタを求める
23-28. no 個の要素の消去
27. 最終要素を越えた場合は中止する
29. 指定要素が先頭要素の場合
30. すべての要素が消去された
36. 指定要素が最終要素でない場合
37. 最終要素まで消去する場合
38. 中間の要素を消去する場合

解答 12-2-6

*A. [12-2-1]の 1-8 行を挿入する。
```
1.  DATA *inslist(DATA *,int,int);
2.  main()
3.  {
4.      int n1,n2;
```
*B. [12-2-1]の 12-19 行を挿入する。
```
5.      printf("挿入を始める要素番号 > ");
6.      scanf("%d",&n1);
7.      printf("挿入する個数 > ");
8.      scanf("%d",&n2);
9.      first=inslist(first,n1,n2);
10.     displist(first);
```

```
    freelist(first);
}
```

*C. [12-2-1]の 23-64 行を挿入する。

```
DATA *inslist(DATA *fst,int st,int no)
{
    int i;
    DATA *p,*pp,*pn,*pf,*pl;
    for(i=0;i<no;i++){
        p=(DATA *)malloc(sizeof(DATA));
        if(i==0){
            p->prev=NULL;
            p->next=NULL;
            pf=p;
            pl=p;
        }
        else{
            pl->next=p;
            p->prev=pl;
            p->next=NULL;
            pl=p;
        }
        p->data=0;
    }
    p=fst;
    for(i=0;i<st;i++){
        pp=p;
        p=p->next;
        if(p==NULL) break;
    }
    if(p==NULL){
        pp->next=pf;
        pf->prev=pp;
        return fst;
    }
    else if(p->prev==NULL){
        pl->next=fst;
        fst->prev=pl;
        return pf;
    }
    else{
```

```
50.         pp->next=pf;
51.         pf->prev=pp;
52.         pl->next=p;
53.         p->prev=pl;
54.         return fst;
55.     }
56. }
```

注意 12-2-6

*A. DATA 型の定義他
*B. makelist(),displist(),freelist()関数の定義
17-32. 要素 no 個分のメモリ確保
　　メモリ確保出来ない場合の処理は簡単のため行っていない
　　先頭要素が pf 最終要素が pl となる
33-38. 指定要素から st 番目の要素の探索
35. pp は p の常に 1 つ前の要素となる
　p が NULL の場合は最終要素である
39. pp が最終要素の場合
40,41. 新しい要素を最終要素の後ろに繋ぐ
44. p が先頭要素の場合
45,46. 先頭要素の前に新しい要素を挿入する
49. 中間の要素の場合

解答 12-2-7

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  typedef struct tree{
4.      struct tree *left;
5.      int data;
6.      struct tree *right;
7.  }DTREE;
8.  void freetree(DTREE *);
9.  DTREE *malloctree(int);
10. void addtree(DTREE *,int);

11. main()
12. {
13.     int i,a;
14.     DTREE *first;
15.     a=rand()%1000;
```

```
16.     first=malloctree(a);
17.     for(i=1;i<100;i++){
18.         a=rand()%1000;
19.         addtree(first,a);
20.     }
21.     freetree(first);
22. }

23. void freetree(DTREE *p)
24. {
25.     if(p==NULL) return;
26.     freetree(p->left);
27.     freetree(p->right);
28.     free(p);
29. }

30. DTREE *malloctree(int no)
31. {
32.     DTREE *p;
33.     p=(DTREE *)malloc(sizeof(DTREE));
34.     p->left=NULL;
35.     p->data=no;
36.     p->right=NULL;
37.     printf("%4d ",p->data);
38.     return p;
39. }

40. void addtree(DTREE *p,int no)
41. {
42.     DTREE *pp;
43.     if(p==NULL) return;
44.     if(no>=p->data){
45.         if(p->right==NULL) p->right=malloctree(no);
46.         else addtree(p->right,no);
47.     }
48.     else{
49.         if(p->left==NULL) p->left=malloctree(no);
50.         else addtree(p->left,no);
51.     }
52. }
```

注意 12-2-7

4-8．DTREE 型の定義
9-11．関数のプロトタイプ宣言
13．DTREE 型データの領域解放
16．最初の節を作る
17-20．乱数を発生させデータを表示しながらツリー構造を構築してゆく
21．メモリの解放
26-28．左右の枝が全てなくなったとき 1 つの節を解放する
34-36．データの初期設定
37．結果参照のために入れている
44．データが節の値以上の場合
45．節の right が NULL の場合，right に節を付ける
46．節の right が NULL でない場合，right の節を探索する
48．データが節の値未満の場合
49．節の left が NULL の場合，left に節を付ける
50．節の left が NULL でない場合，left の節を探索する

解答 12-2-8

```
*A. [12-2-7]の 1-10 行を挿入する。
1.  int maxtree(DTREE *)
2.  int mintree(DTREE *)
3.  main()
4.  {
*B. [12-2-7]の 13-20 行を挿入する。
5.      a=maxtree(first);
6.      printf("max=%d\n",a);
7.      a=mintree(first);
8.      printf("min=%d\n",a);
9.      freetree(first);
10. }
*C. [12-2-7]の 23-52 行を挿入する。

11. int maxtree(DTREE *p)
12. {
13.     int a;
14.     if(p->right==NULL) a=p->data;
15.     else a=maxtree(p->right);
16.     return a;
17. }
18.
```

```
19. int mintree(DTREE *p)
20. {
21.     int a;
22.     if(p->left==NULL) a=p->data;
23.     else a=mintree(p->left);
24.     return a;
25. }
26.
```

注意 12-2-8

11. p の節から最大値を探索する関数
14. right が NULL なら値が求まる
15. right が NULL でなければ right を探索する
19. p の節から最小値を探索する関数
22. left が NULL なら値が求まる
23. left が NULL でなければ left を探索する

解答 12-2-9

```
*A. [12-2-7]の 1-11 行を挿入する。
1.  void disptree(DTREE *)
2.  main()
3.  {
*B. [12-2-7]の 13-20 行を挿入する。
4.      disptree(first);
5.      printf("\n");
6.      freetree(first);
7.  }
*C. [12-2-7]の 23-52 行を挿入する。

8.  void disptree(DTREE *p)
9.  {
10.     if(p==NULL) return;
11.     disptree(p->left);
12.     printf("%4d ",p->data);
13.     disptree(p->right);
14. }
```

注意 12-2-9

8. p 以下のデータを小さい順に表示する関数
11. 最初に left を全て探索し表示する
12. p のデータを表示する

13. 最後に right を全て探索し表示する

解答 12-2-10

```
*A. [12-2-7]の 1-11 行を挿入する。
1.  int searchtree(DTREE *, int)
2.  main()
3.  {
*B. [12-2-7]の 13-20 行を挿入する。
4.      a=searchtree(first,500);
5.      printf("search=%d\n",a);
6.      freetree(first);
7.  }
*C. [12-2-7]の 23-52 行を挿入する。

8.  int searchtree(DTREE *p,int no)
9.  {
10.     int a;
11.     if(p->data==no) a=p->data;
12.     else if(p->data<no){
13.         if(p->right==NULL) a=-1;
14.         else a=searchtree(p->right,no);
15.     }
16.     else{
17.         if(p->left==NULL) a=p->data;
18.         else a=searchtree(p->left,no);
19.     }
20.     return a;
21. }
22.
```

注意 12-2-10

8. p から調べて no 以上で no に最も近い値を求める関数
11. 現在の節の値が no に等しい場合はその値
12. 節の値が no より小さい場合
13. 節の right が NULL ならば-1 を返す
    これはみつからないことを表すが，答の-1 との区別が問題
14. 節の right が NULL でなければ right を探索する
16. 節の値が no より大きい場合
17. 節の left が NULL の場合はその節の値
18. 節の left が NULL でない場合は left を探索する

１３章 割込み

13-1) 割り込み

解答 13-1-1

```
 1. #include <dos.h>
 2. main()
 3. {
 4.     unsigned char c;
 5.     union REGS inregs,outregs;
 6.     while(1){
 7.         inregs.h.ah=0x8;
 8.         intdos(&inregs,&outregs);
 9.         c=outregs.h.al;
10.         if(c==0x1b) break;
11.         inregs.h.ah=0x2;
12.         inregs.h.dl=c;
13.         intdos(&inregs,&outregs);
14.     }
15. }
```

注意 13-1-1

1. intdos()関数のためのインクルードファイル
5. 構造体 BYTEREGS と WORDREGS で作られた共用体
7. AH レジスタには機能番号 8_H を代入する
8. ファンクションコール(INT 21_H)を実行する
9. AL レジスタの中に入力文字のアスキーコードが入っている
10. 入力が ESC ならば終了する
11. AH レジスタには機能番号 2_H を代入する
12. DL レジスタには出力したい文字のアスキーコードを代入する
13. ファンクションコール(INT 21_H)を実行する

ファンクションコール

08_H. 標準入力から AL に 1 文字入力する
02_H. DL に与えられた文字を標準出力に 1 文字出力する

解答 13-1-2

```
1. #include <stdio.h>
2. #include <dos.h>
3. main()
```

```
4.  {
5.      unsigned char c;
6.      union REGS inregs,outregs;
7.      inregs.h.ah=0x19;
8.      intdos(&inregs,&outregs);
9.      c=outregs.h.al;
10.     printf("%c.\n",'A'+c);
11. }
```

注意 13-1-2

2. intdos()関数のためのインクルードファイル
6. 構造体 BYTEREGS と WORDREGS で作られた共用体
7. AH レジスタには機能番号 19_H を代入する
8. ファンクションコール(INT 21_H)を実行する
9. AL レジスタにドライブ番号に相当した値が入っている
   A. 00 , B. 01 , ･･･
10. A., B. ･･･ の形で出力する

ファンクションコール

19_H. カレントドライブを取得する

解答 13-1-3

```
1.  #include <stdio.h>
2.  #include <dos.h>
3.  main()
4.  {
5.      char str[80];
6.      union REGS inregs,outregs;
7.      struct SREGS sregs;
8.      inregs.h.ah=0x47;
9.      inregs.h.dl=0;
10.     segread(&sregs);
11.     inregs.x.si=(unsigned int)str;
12.     intdosx(&inregs,&outregs,&sregs);
13.     printf("current dir=%s\n",str);
14. }
```

注意 13-1-3

このプログラムは S モデル用です

2. intdosx(),segread()関数のためのインクルードファイル
7. ES,CS,SS,DS レジスタ設定用の構造体
8. AH レジスタには機能番号 47_H を代入する

9. DL レジスタにはドライブ番号を代入する
   カレント. 00 , A. 01 , ･･･
10. 現在の ES,CS,SS,DS レジスタ値を取得する
11. 現在のポインタ値を SI レジスタに代入する
   S モデルの場合ポインタはオフセット値になる
10,11. セグメントとオフセットの値の取得には FP_SEG(),FP_OFF()関数も
   利用出来る
12. DS レジスタの値を指定したファンクションコールの実行
ファンクションコール
  47_H. 指定ドライブのカレントディレクトリ名を取得する

解答 13-1-4

```
1.  #include <stdio.h>
2.  #include <dos.h>
3.  main()
4.  {
5.      unsigned int year;
6.      unsigned char month,day;
7.      unsigned char hour,min,sec;
8.      union REGS inregs,outregs;

9.      inregs.h.ah=0x2a;
10.     intdos(&inregs,&outregs);
11.     year=outregs.x.cx;
12.     month=outregs.h.dh;
13.     day=outregs.h.dl;

14.     inregs.h.ah=0x2c;
15.     intdos(&inregs,&outregs);
16.     hour=outregs.h.ch;
17.     min=outregs.h.cl;
18.     sec=outregs.h.dh;

19.     printf("%04d-%02d-%02d/",year,month,day);
20.     printf("%02d.%02d.%02d\n",hour,min,sec);
21. }
```

注意 13-1-4

2. intdos()関数のためのインクルードファイル
9. AH レジスタには機能番号 $2A_H$ を代入する

10. ファンクションコールの実行
11. CX レジスタ(2 バイト)にはシステム時計の年の値が返される
12. DH レジスタには月の値が返される
13. DL レジスタには日の値が返される
14. AH レジスタには機能番号 $2C_H$ を代入する
15. ファンクションコールの実行
16. CH レジスタにはシステム時計の時間の値が返される
17. CL レジスタには分の値が返される
18. DH レジスタには秒の値が返される
19. 年月日の表示
20. 時分秒の表示

ファンクションコール

$2A_H$. システム時計の年月日，曜日の取得
$2C_H$. システム時計の時分秒の取得

解答 13-1-5

```
1.  #include <stdio.h>
2.  #include <dos.h>
3.  #include <conio.h>

4.  main()
5.  {
6.      int c,i;
7.      while(1){
8.          keyclear();
9.          c=getch();
10.         if(c==0x1b) break;
11.         for(i=0;i<800;i++) printf("a");
12.         printf("\n\n");
13.     }
14. }

15. void keyclear(void)
16. {
17.     unsigned char c;
18.     union REGS inregs,outregs;
19.     inregs.h.ah=0xb;
20.     intdos(&inregs,&outregs);
21.     c=outregs.h.al;
22.     if(c!=0x0){
```

```
23.             inregs.h.ah=0xc;
24.             inregs.h.al=0x0;
25.             intdos(&inregs,&outregs);
26.         }
27. }
```

注意 13-1-5

8. キーボードバッファを空にする
9. キーボードからの 1 文字入力
10. 入力が ESC の場合は終了する
11. 800 個の"a"を出力する
    1 回の処理に時間がかかるのでキーを押し続けた場合キーボードバッファに文字がたまってくる
19. AH レジスタには機能番号 $0B_H$ を代入する
20. ファンクションコールの実行
21. AL レジスタに入力バッファの情報が返される
    FF_H. 入力バッファに文字が入っている
    00_H. 入力バッファに文字が入っていない
22. 入力バッファに文字が入っている場合
23. AH レジスタには機能番号 C_H を代入する
24. AL レジスタに代入する値によって機能が異なる
    AL が 01,06,07,08,0A 以外はバッファを空にするだけ
25. ファンクションコールの実行

ファンクションコール

$0C_H$. キーボードバッファを空にして入力する