1. C 言語(1) C 言語の基本

1.1. プログラムができるまで

C 言語でプログラムを作成する流れを図 1.1 に示します。ユーザが作成するのは基本的に，プログラムの本体であるソースファイル(拡張子が.c)と，プログラム中に必要な定義を記述するヘッダファイル(拡張子が.h)です。これらに「コンパイル」という処理を行うと，オブジェクトファイル(拡張子が.obj)というファイルができます。この中には，プログラムを実行するのにはどのような標準的な機能(例えば，画面に表示したり，キーボードからの入力を理解したり)が必要かが記述されます。そして，「リンク」という処理をすると，オブジェクトファイルに記述してある標準機能をライブラリ(拡張子が.lib)から読み込み，組み合わせることで実行ファイル(.exe)を作成します。つまり，「**ソースファイルと**(必要なら)**ヘッダファイルを書いて，コンパイルとリンクをするとプログラムができる**。」ということです。

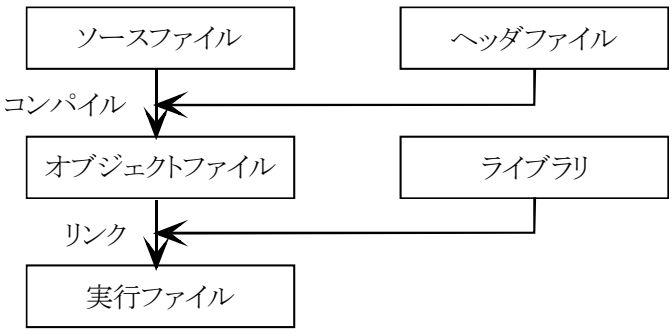

図1.1 プログラムができるまで

C 言語ソースは基本的に図 1.2 のような構造になっています。いくつかの基本規則があります。

1) ソースは 1 つまたは複数の関数からなり，main という関数から実行される。
2) 「#」で始まる命令はプリプロセッサ命令といい，コンパイル前に実行される。
3) 命令中の改行・スペースは無視され，「;」までを 1 行の命令とする。
4) 「/*」，「*/」で挟まれた部分はコメントとして，コンパイル時に無視される。
5) 「{」，「}」で挟まれた部分を 1 つのブロックとして処理する。

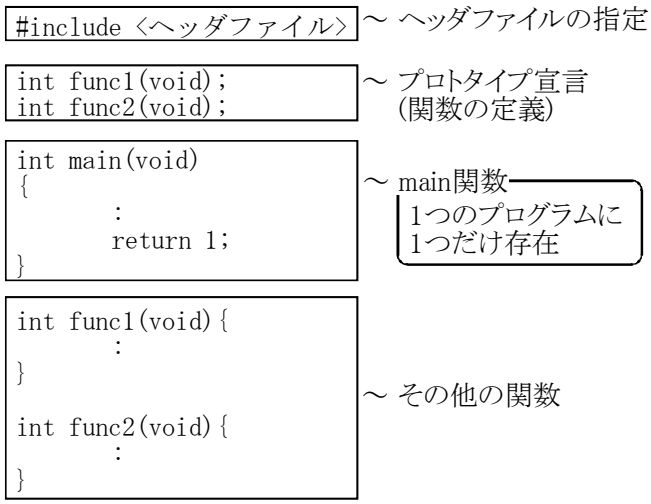

図1.2 ソースファイルの基本構造

実際に C 言語で既述したプログラムソースを list1.1 に示します。各行の意味は以下の通りです。

1 行目　stdio.h を読み込むプリプロセッサ命令
3 行目　メイン関数の定義。引数はなし，戻値は int(整数)。
5 行目　変数 a, b を int と定義し，1, 2 を代入する。
7・10 行目　コメント
8 行目　printf という標準関数を使用して，「Hello world。」という文字を表示し，改行する。
11・12 行目　a と b の和・差を計算し，表示する。
14 行目　戻値として 1 を返して，この関数(すなわち，このプログラム)を終了する。

```
 1:    #include      〈stdio.h〉
 2:
 3:    int main(void)
 4:    {
 5:          int a = 1, b = 2;
 5:
 7:          /* 文字の表示 */
 8:          printf("Hello world.¥n");
 9:
10:          /* 計算式 */
11:          printf("A + B = %d¥n", a + b);
12:          printf("A - B = %d¥n", a - b);
13:
14:          return 1;
15:    }
```

list1.1 Cプログラムソースの例

このように，C 言語でプログラムを書くには，「**変数の使い方と標準関数を覚えて，独自の関数を書く。**」のが基本と言えます。

表 1.1 変数型

型	表記	意味	バイト	範囲	32bit
文字	char	文字	1	−128～128	1
整数	short (int)	整数	2	−32768～32768	2
	int		2	−32768～32767	4
	long (int)		4	-2^{31}～$2^{31}-1$	4
実数	float	浮動小数点実数(7 桁)	4	$\pm3.4\times10^{38}$	4
		実数部 24bits，指数部 8bits			
	double	倍精度浮動小数点実数(15 桁)	8	$\pm1.8\times10^{308}$	8
		実数部 53bits，指数部 11bits			
	long double	倍精度浮動小数点実数(15 桁)	10	$\pm1.14\times10^{4932}$	10
		実数部 64bits，指数部 16bits			

1.2. 変数の使い方

1.2.1. 変数型

C 言語で使用する基本変数型を表 1.1 に示します。それぞれの変数には，signed(符号つき)と unsigned(符号なし正数)の 2 種類があり，省略すると signed になります。(例えば，char は singed char と同意で−128～127 まで，unsigned char は 0～255 まで)

自動変数と静的変数

変数にはスコープと呼ばれる使用可能範囲があり，基本的に「{」から「}」までがその範囲となります。変数はスコープ内の宣言された時点でメモリ上に配置され，スコープがなくなった時点で消滅します。つまり，関数などで宣言した変数は，その関数の中でだけ使用でき，関数が終了すると共に消滅します。このように自動的に消滅する変数を自動変数と言います。通常の宣言で作成される変数は全て自動変数となります。

一方，変数の宣言時に「static」と付けるとある特定のメモリ領域が確保され，スコープがなくなっても変数は消滅しません。このような変数を静的変数と呼びます。静的変数も使用可能範囲はスコープ内なので他の関数から使用することはできませんが，スコープを失っても値を保持しているので，値を保存しておきたい時などに使用します。

ローカル変数(内部変数)とグローバル変数(外部変数)

変数はスコープによってローカル変数とグローバル変数に分けられます。ローカル変数は「{}から「}」までのスコープを持ち，その範囲内だけで使用できます。一方，グローバル変数は「{」から「}」の外で宣言をするので，スコープを持ちません(正確にはファイルスコープというスコープがある)。そのため，ファイル内のどの関数からも参照できます(「extern」という宣言を使うとファイル外からも参照できる)。関数が多くなると変数の受け渡しが面倒になるので，グローバル変数を多用したくなりますが，可搬性の問題から一般的には使用しないほうがいいとされています。

1.2.2. 定数

数定数 C 言語では 10 進数，16 進数が主に使用されます。10 進数はそのままの数字で，16 進数は数字の先頭に「0x」を付けて示します。また，long 型の定数には最後に「L」を付けます。

単一文字 char 型の変数は単一の文字を表すのにも使用され，「'(シングルクォーテーション)」で挟んで定義します。

文字列 文字列は char 型の配列として扱い，「"(ダブルクォーテーション)」で挟んで定義します。文字列の最後には必ず文字列の終端を示す NULL(ヌル，ナル)文字(¥0)が付けられます。

エスケープ文字 コントロールコードのように表現できない文字を示すためにはエスケープも時を使用します。エスケープ文字は記号「¥」の後に文字を続けて定義します。例えば，改行コードは「¥n」，タブコードは「¥t」です。

1.2.3.　演算子

算術演算子　加減乗除を示す「+」,「−」,「*」,「/」と, 剰余(モジュロ)を示す「%」があります。

代入演算子　代入式を示す「=」と, 複合演算子「+=」,「−=」,「*=」,「/=」があります。複合演算子の意味は以下の通りです。

```
x += 1;  →  x = x + 1;          x −= 1;  →  x = x − 1;
x *= 1;  →  x = x * 1;          x /= 1;  →  x = x / 1;
```

インクリメント／デクリメント演算子　変数の値を 1 増減させます。頻繁に使用しますが前置と後置で高価が違うので注意してください。

```
i++;  →  i = i + 1;             i−−;  →  i = i − 1;
++i;  →  i = i + 1;             −−i;  →  i = i − 1;
j = i++;  →  j = i; i = i + 1;  j = i−−;  →  j = i; i = i − 1;
j = ++i;  →  i = i + 1; j = i;  j = −−i;  →  i = i − 1; j = i;
```

関係演算子　条件を表す「==」,「!=」,「>」,「<」,「>=」,「<=」があります。

論理演算子　変数の論理演算を行う演算子です。NOT は「!」, 論理 AND は「&&」, 論理 OR は「||」を使用します。例えば,「もし, a < x < b ならば･･･をしなさい」は以下のようになります。

```
if((a < x) && (x < b))  ･･･;
```

条件演算子　条件分岐に使用します。「?」は条件に。「:」は分岐を示します。例えば,「a と b の大きいほうを max に代入する」は以下のようになります。

```
max = (a > b) ? a : b;
```

ビット演算子　ビット単位の演算を行います。ビット AND は「&」, ビット OR は「|」, XOR は「^」, 左シフトは「<<」, 右シフトは「>>」, 1 の補数[1)]は「~」を使用します。

キャスト演算子　変数型を強制的に変換する時に使用します。「int の変数 i に double の変数 d を代入する」は以下のようになります。

```
i = (int)d;
```

ポインタ演算子　C 言語の変数型の一つに, ある変数のメモリアドレスとその型を保持する型, ポインタがあります。ポインタ演算子「*」はあるポインタ ptr の頭に*ptr と付けることで, その変数の実体を示すことができます。ポインタの扱いはかなり複雑なので改めて取り上げることにします。

アドレス演算子　変数の格納されるメモリアドレスを取り出す場合にはアドレス演算子「&」を使用します。

1.2.4.　制御文

プログラムの流れを制御するための文を制御文と言います。

if 文　条件分岐を制御します。

```
if(条件式){
        実行される文;
}
```

実行される文が 1 行の場合は「{」,「}」を省略することもできます。また, 条件にあてはまらない時に実行する文は「else」で指定します。

```
if(条件式){
        実行される文;
}else{
        実行される文;
```

1) 1 の補数: 各ビットに対して 1 を 0 に, 0 を 1 にした値. 2 の補数は 1 の補数に 1 を加えたもので, 負数を示すのに用いられる.

```
}
```

if～else～を組み合わせることで多重分岐をすることもできます。

for 文 数値を変化させながら，ある処理を繰り返す制御を行います。

```
for(初期設定; 実行条件; 繰返し文){
        実行される文;
}
```

実行の流れは，まず初期設定が行われ，実行条件を評価します。実行条件を満たしていれば，ブロック内の文を実行し，繰返し文を実行します。その結果，実行条件を満たしていれば，繰り返しブロック内を実行し，条件を満たしていなければ，ブロックを抜けます。初期設定には繰り返しに使用する変数以外の変数を指定してもかまいません。

```
int     a, b, i;
for(a = 0, b = 0, i = 0; i < 100; i += 2){
        実行される文;
}
```

また，実行条件を設定しなければ無限ループになります。無限ループは抜けることができなくなる時もあるので，注意しましょう。

while 文 for 文と同様に繰り返しを制御するために使用します。

```
while(条件式){
        実行される文;
}
```

実行の流れは，まず条件式が評価され，条件式が真(TRUE)であれば，ブロック内を実行します。そのため，条件式が初めから偽であれば，文は実行されません。

do～while 文 while 文の変形で，条件判断はループの出口で行われます。そのため必ず 1 度はブロック内の文は実行されます。

```
do{
        実行される文;
}while(条件式);
```

最後の「;」を忘れずに。

switch 文 多重分岐をするために用います。

```
switch(変数){
case 条件 1:        実行される文 1;
                    break;
case 条件 2:        実行される文 2;
                    break;
・・・・・・・・・・
default:            実行される文 n;
}
```

実行は指定された変数をもとに条件分岐を行います。変数には整数を使用します。条件にあてはまるラベルがあった場合はそのラベルにあたる文を実行します。文の途中に break 文があると実行を中止し，switch 文の最後までジャンプします。break 文がない場合は，次のラベルに行っても処理を継続します。条件にあてはまるラベルがない時は，default で指定された処理を行います。default で何もしたくない時は continue;を記述しておきます。

1.3. 標準関数

1.3.1. 標準入出力関数 （使う時は，stdio.h を include する）

1) 1 文字の入出力

`int getchar(void);`

標準入力(普通はキーボード)から 1 文字入力する。成功すると読み込んだ文字を，エラーで「EOF」(End Of File: stdio.h で定義済みのファイルエンドを示す値)を返します。

`int putchar(int c);`

標準出力(普通は画面)に 1 文字出力する。成功すると文字「c」を，エラーで「EOF」を返します。

2) 文字列の入出力

`char *gets(char *s);`

標準入力から復帰文字(リターン)で終了する文字列を読み込んで，s に格納します。s の中の復帰文字は NULL に変換されます。成功すると文字列 s を，エラーで NULL を返します。

`int puts(const char *s);`

標準出力に NULL で終了する文字列 s を出力し，最後に改行をします。成功すると負でない値を，エラーで EOF を返します。

3) 書式付き入出力

`int printf(const char *format, ...);`

format に文字列や書式文字列，その後ろに書式指定に対応した引数を指定することで，書式付きの出力を行います。書式文字列に変換指定子を入れることで，様々な出力をすることができます。各変換指定子は「%」で始まり，左から右に解釈されます。format の左から最初の書式指定子を(もしあれば)見つけると，format の後の引数をその書式に従って変換して出力します。2 つ以上の書式指定子がある場合も同様です。各書式指定子は，

% [flag] [width] [.prec] [h | l | | I64 | L] type_char

という形式で記述され，それぞれ以下のような意味を持っています。[ ]のものは省略も可能です。

[flags]　出力の位置決めと符号，空白，小数点，10 進数・8 進数・16 進数の出力を制御します。1 つの書式指定に，複数のフラグを指定できます。

- `-`　左詰めで表示する。
- `+`　出力値が符号つきの場合，「+」または「-」を表示する。
- `0`　最小幅まで 0 が付加される。0 フラグと-フラグを指定すると無視される。整数書式(i, u, x, X, o, d)では無視される。
- `' '`(空白)　出力値が符号つきで整数であると，出力値の前に空白もしくは「-」を表示する。+フラグと指定すると無視される。
- `#`　o, x, X で 0 以外のすべての出力の前に 0, 0x, 0X が着く。e, E, f で出力値に強制的に小数点が着く。c, d, i, u, s では無視される。

[width]　出力する最小文字数を指定する。

[.prec]　出力する最大文字数，または出力精度を指定します。

[h | l | | I64 | L]　入力引数のサイズを指定します。「h」は short，「l」，「L」は long，「I64」は_int64(Windows 用)を示します。

type_char　変数の種類を指定します。

- c　1 個の文字 （ただし，1byte 文字）
- d　符号付き 10 進整数
- i　符号付き 8 進整数
- o　符号なし 8 進整数
- u　符号なし 10 進整数

x	符号なし 16 進整数（a, b, c, d, e, f を使用）
X	符号なし 16 進整数（A, B, C, D, E, F を使用）
e	[-]d.dddd e [sign]ddd 形式符号付きの値。d は 1 個の 10 進数，dddd は 1 個または複数の 10 進数，ddd は正確に 3 桁の 10 進数，sign は + または -。
E	e と同じ。「e」の代わりに「E」を用いる。
f	小数
g	f または e の書式を，数値の精度によって使い分ける。
G	g と同じ。「e」の代わりに「E」を用いる。
p	ポインタ
s	文字列

成功すると出力した文字数を，エラーで負の値を返します。

```
int scanf(const char *format, ...);
```

書式付きの読込みを行います。format には printf と同様の書式指定子を用います。ただし，後続の引数にはポインタを使用します。成功すると変換された入力フィールドの数を，エラーで EOF を返します。scanf で書式指定子に%s を用いたときは空白文字(空白，タブ，または改行)までをひとまとまりとして読み込みます。空白文字で区切られていない文字列を読み取るには%s の代わりに%[]を指定します。対応する入力フィールドには[]で囲まれた文字セットに含まれていない最初の文字が見つかるまで読み取られます。また文字セットの最初の文字がカレット(^)のときは逆に作用します。例，%[a-z]は"a～z"以外の文字まで，%[^,]は","の直前までの文字が読み込まれます。

4) 文字列に対する書式付き入出力

```
int sprintf(char *buffer, const char *format, ...);
```

文字列 buffer に書式付き出力を行います。仕様は printf と同じです。

```
int sscanf(const char *buffer, const char *format, ...);
```

文字列 buffer から書式付き入力を行います。使用は scanf と同じです。

1.3.2. ファイル入出力関数 （使う時は，stdio.h を include する）

データをファイルに保存したり，読み込んだりする場合に使用します。C 言語でファイルを読み書きするには，あらかじめファイルをオープンしておく必要があります。オープンされたファイルはファイルストリームと呼ばれるデータで管理され，どのようなモード(読み込みのみ，書き込みのみ，読み書き両用など)か，どこまで読み書きしたか，などが管理されています。このストリームの中には標準入力(stdin)，標準出力(stdout)，標準エラー出力(stderr)なども含まれ，あらかじめオープンしてあります。

1) ファイルストリーム操作関数

```
FILE *fopen(const char *filename, const char *mode);
```

filename で指定されたファイルを mode で指定されたモードでオープンし，ストリームに結びつけます。モード文字列は以下のようになっています。

"r"	読出モード
"w"	書込モード:ファイルがない時は新しく作成します。
"a"	追加モード:ファイルがない時は新しく作成します。
"r+"	読出，書込両用モード:既存ファイルのみ対象となります。
"w+"	読出，書込両用モード:既存ファイルがあっても上書きします。
"a+"	読出，追加両用モード:ファイルがない時は作成します。

モード文字列に「b」を追加するとバイナリモードに，「t」を追加するとテキストモードになります。指定しないとグ

ローバル変数「_fmode」に従ってオープンします。標準ではテキストモードです。

成功するとストリームをさすポインタを，エラーのときは NULL を返します。

`int fclose(FILE *stream);`

オープンしたストリーム stream をクローズします。成功すると 0 を，エラーで EOF を返します。

`int fcloseall(void);`

オープンしているすべてのストリームをクローズします。成功するとクローズしたストリーム数を，エラーで EOF を返します。

2) ファイルに対する 1 文字の入出力

`int getc(FILE *stream);`

stream から 1 文字を読み込みます。成功すると読み込んだ文字を，エラーもしくはファイルエンドで EOF を返します。

`int putc(const int c, FILE *stream);`

stream に 1 文字を出力します。成功すると文字 c を，エラーで EOF を返します。

3) ファイルに対する文字列の入出力

`char *fgets(char *s, int n, FILE *stream);`

stream から文字列を読み込んで s に格納します。読込みは，n-1 個の文字を読み込むか，改行文字を読み込んだときに終了します。改行文字で終了した場合は改行文字も保存し，その後ろに NULL 文字を付加します。成功すると文字列 s を，エラーで NULL を返します。

`int fputs(const char *s, FILE *stream);`

stream に文字列 s を書き込みます。成功すると最後に書き込まれた文字を，エラーで EOF を返します。

4) ファイルに対する書式付き入出力

`int fprintf(FILE *stream, const char *format, ...);`

`int fscanf(FILE *stream, const char *format, ...);`

ファイルストリームに入出力を行う以外は printf, scanf と同じです。

ファイル入出力の一般的な例

ファイルからの読み込み

```
FILE * fp;
fp = fopen("xxxxx.txt", "rt");
if(fp == NULL){
    printf("Cannot open file¥n");
    return 0;
}
fscanf(fp, ...)
fclose(fp);
```

ファイルへの書き出し

```
FILE * fp;
fp = fopen("xxxxx.txt", "wt");
if(fp == NULL){
    printf("Cannot create file¥n");
    return 0;
}
fprintf(fp, ...);
fclose(fp);
```

1.3.3. コンソール入出力関数 （使う時は，conio.h を include する）

ハードウェアから直接入出力する関数。Windows 上ではあまり使用しないほうが良い。

`int kbhit(void);`

キーボードが押されたかをチェックします。押されたキーは getch か getche で取得できます。キーが押されていれば 0 以外の整数，押されていなければ 0 を返します。

`int getch(void);`

`int getche(void);`

getch はエコーバック(画面表示)なし，getche はエコーバック付きでキーボードから文字を読み込みます。文字コードを返します。

1.3.4. 文字列操作関数 (使う時は, string.h を include する)

1) 文字列のコピー

```
char *strcpy(char *dst, const char *src);
char *strncpy(char *dst, const char *src, size_t maxlen);
```

文字列 src を dst にコピーします。コピーは NULL 文字をコピーしたところで終了します。strncpy は最大文字数 maxlen 個までコピーし, 必要なら NULL を付加します。文字列 dst へのポインタを返します。

2) 文字列の追加

```
char *strcat(char *dst, const char *src);
char *strncat(char *dst, const char *src, size_t maxlen);
```

文字列 src を dst の最後に追加します。得られる文字列の長さは dst の長さ+src の長さになるの, dst のサイズに注意してください。srrncat は最大文字数 maxlen 個まで追加し, 必要なら NULL を付加します。文字列 dst へのポインタを返します。

3) 文字列の比較

```
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t maxlen);
```

文字列 s1 と文字列 s2 の各文字を比較します。strncmp は最大文字数 maxlen 個まで比較します。両者とも以下のような値を返します。

s1 が s2 よりも小さければ	< 0
s1 が s2 と等しければ	= 0
s1 が s2 よりも大きければ	> 0

4) 文字列の長さを知る

```
size_t strlen(const char  *s);
```

文字列 s の長さを返します。NULL 文字は数に含まれません。

1.3.5. その他の標準関数 (使う時は, stdlib.h を include する)

1) 乱数の発生

```
int rand(void);
void srand(unsigned int seed);
```

周期 2^{32} の 乗法合同法を用いて乱数を発生し, 呼び出されるたびに 0～RAND_MAX (設定されていないものもある)の間で擬似乱数を発生させます。srand は, 適当な seed を与えることで, 乱数発生の新しい開始点にセットします。

2) クイックソートアルゴリズム

```
void qsort(void *base, size_t num, size_t width, int (*compare)(const void *elem1,
        const void *elem2));
```

クイックソートアルゴリズムを使用して, 配列 base の num 個, サイズ width バイトのデータを関数 compare に従ってソートします。比較関数を func(a, b)とした時, 戻値を下のように定義すると昇順にソートされます。

比較		戻値	
	a < b		< 0
	a = b		= 0
	a > b		> 0

1.3.6. 数学関数 (使う時は, math.h を include する)

1) 三角関数

```
double sin(double x);  (cos, tan, asin, acos, atan も型は同じ)
```

`double atan2(double y, double x);`

sin, cos, tan はそれぞれ x の三角関数値を返します。角度の単位は rad です。asin, acos, atan はそれぞれ x の逆参画関数値を返します。atan2 は y/x の atan 値を返します。atan は角度が$\pi/2$, $-\pi/2$ のとき誤差が大きいので, 一般的に atan2 を使用します。

2) 対数･指数関数

`double log(double x);` (log10, exp も型は同じ)

`double pow10(int p);`

log, log10 はそれぞれ x の自然対数, 常用対数を示します。exp, pow10 はそれぞれ基数 *e*, 10 の指数関数値を示します。

3) その他の関数

`double ceil(double x);`

`double floor(double x);`

ceil は x の小数点以下を切り上げ, floor は小数点以下を切り捨てた値を返します。

`double fmod(double x, double y);`

x を y で割った剰余を返します。y = 0 のときは 0 を返します。

`double pow(double x, double y);`

x の y 乗を返します。

`double fabs(double x);`

x の絶対値を返します。

`double modf(double x, double *ipart);`

x を整数部と小数部に分割し, 整数部を ipart に格納し, 小数部を返します。

`double sqrt(double x);`

x の平方根を返します。

`double hypot(double x, double y);`

直角三角形の 2 辺の長さ x, y から斜辺の長さを返します。

`double poly(double x, int degree, double *coeffs);`

係数 coeffs[0], coeffs[1], ‥‥, coeffs[degree]をもつ degree 次の x に関する多項式の x における値を返します。

【練習問題 1.1】 if 文

数字を入力し, 入力されたのが 1 なら"Yes"を, それ以外なら"No"を表示するプログラムを作成しなさい。

【練習問題 1.2】 for 文

入力した数までの階乗を求めて表示するプログラムを作成しなさい。

【練習問題 1.3】 while 文

入力された数字が 0 になるまで, 数字の入力と入力した数字の表示を繰り返すプログラムを作成しなさい。

【練習問題 1.4】 switch 文

1 が入力されたら「One.」と表示, 2 が入力されたら「Two.」と表示, その他は「It is neither one nor two.」と表示するプログラムを作成しなさい。

【練習問題 1.5】 コンソール関数

キーボードから入力があるまで"Waiting for keyin."と何度も表示する無限ループ(for 文や while 文)をつくり, キー入

力されたら押されたキーのキーコードを表示して終了するプログラムを作成しなさい。

【練習問題 1.6】 数学関数

以下のものを計算し，表示せよ。

(1) 2^8　(2) sin(67°)　(3) –23.456 の絶対値　(4) 7÷3 の剰余　(5) 0～1 の範囲の乱数を 10 個

【練習問題 1.7】 標準入出力関数

100 個のデータ列 t (整数), x, y, z (0～1 のランダムな実数)を以下の形式でファイルに出力するプログラムを作成しなさい。データはカンマ区切りで出力するものとする。

例
```
t,x,y,z
0,0.567464,0.4544894,0.448468
1,0.164894,0.159618,0.4848975
:
:
99,0.4488,0.97845,0.2712634
```

【練習問題 1.8】 標準入出力関数

練習問題 1.7 で作成したデータファイルを読み込みながら，表示するプログラムを作成しなさい。

【練習問題 1.9】 標準入出力関数

練習問題 1.7 で作成したデータファイルを読み込んで，x に対する y (もしくは z)の線形回帰直線を求め，表示するプログラムを作成しなさい。

[線形回帰]

n 個の観測値 x ， y に対して $y = ax + b$ の線形関係があるとすると，観測点 i の自乗誤差 ε_i^2 は，

$$\varepsilon_i^2 = \{y_i - (ax_i + b)\}^2 = y_i^2 + a^2 x_i^2 + b^2 - 2ax_i y_i + 2abx_i - 2by_i$$

となり，観測値全部の二乗誤差和は，

$$S_\varepsilon = \sum \varepsilon_i^2 \ , \ Sxx = \sum x_i^2 \ , \ Syy = \sum y_i^2 \ , \ Sxy = \sum x_i y_i \ , \ Sx = \sum x_i \ , \ Sy = \sum y_i$$

とおくと，

$$S_\varepsilon = Syy + a^2 \cdot Sxx + nb^2 - 2a \cdot Sxy + 2ab \cdot Sx - 2b \cdot Sy$$

となる。この誤差を最小(=0)とする a ， b は，

$$\partial S_\varepsilon / \partial a = 2a \cdot Sxx - 2Sxy + 2b \cdot Sx = 0$$

$$\partial S_\varepsilon / \partial b = 2nb + 2a \cdot Sx - 2Sy = 0$$

より，

$$\begin{pmatrix} Sxy \\ Sy \end{pmatrix} = \begin{pmatrix} Sxx & Sx \\ Sx & n \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix}$$

よって，

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{n \cdot Sxx - Sx^2} \begin{pmatrix} n & -Sx \\ -Sx & Sxx \end{pmatrix} \cdot \begin{pmatrix} Sxy \\ Sy \end{pmatrix}$$