プログラミング言語 C の用語集

作者：金子邦彦　クリエィティブコモンズ BY NC SA

アクセス

データの読み出し，書き込みを行うことをアクセスという．「メモリをアクセスする」，「ディスクをアクセスする」などのように言う．

値による呼び出し

関数の呼び出しにおいて，値が渡されることを「値による呼び出し」という． 関数に渡された値は，関数の仮引数にセットされるが， 関数の引数と，関数の仮引数とは本来別のものであるので，関数の中で仮引数の値を変更したとしても，その影響が，関数の引数に及ばない．

関連項目：参照による呼び出し

アドレス

アドレスとは，メモリ内での位置のこと． 次の C プログラムは，データとそのアドレスを表示する．

```
#include <stdio.h>

main()
{
  int data;
  printf( "データ=%d, アドレス=%d¥n", data, &data );
}
```

関連項目： ポインタ

アレイ

配列ともいう．

入れ子

次のプログラムでは，ループが入れ子になっている．

```
main()
{
  int i,j;
  int x[10][20];
  for( i=0; i<10; i++ ) {
    for( j=0; j<20; j++ ) {
      x[i][j] = 0;
    }
  }
}
```

インクリメント

値を1増やすことをインクリメントという．

関連項目：デクリメント

インデント

字下げともいう．

エディタ

エディタとは，編集する機能を持ったプログラムのこと．あるいは，編集を行うための画面のこと．

エラー

エラーには，次の種類がある．

- 構文エラー（シンタックスエラー）
  プログラムの構文上の誤りによって，プログラミング言語処理系での処理が中断すること．
- 実行時エラー
  実行時エラーには，定義域エラー（関数の入力が，定義域外であること．例えば，log 関数の引数が負である場合など）や， 範囲エラー（演算の結果が大きすぎたり，小さすぎたり，0に近すぎたりする場合のエラー）などがある．

- 意味的エラー
  プログラムの結果が期待通りではないこと.

演算子

演算子は，1個あるいは2個以上のオペランドを取って，何らかの演算を行うための記号．オペレータともいう．代表的な演算子は次の通り．

論理演算子
- ! 否定
- && 論理積（ＡＮＤ）
- || 論理和(OR)

比較演算子
- != 等しくないか
- < より小さい
- <= 等しいかより小さい
- == 等しいか
- > より大きい
- >= 等しいかより大きい

算術演算子
- % 剰余
- * 積
- + 和
- ++ インクリメント
- - 差
- -- デクリメント
- / 商

その他
- -> 構造体などへのポインタを使って，メンバへのアクセスを行う
- . 構造体などのメンバの選択
- = 代入演算子
- [] 配列の添字

など

オーバーフロー

演算結果の値が大きすぎて(演算結果の絶対値が大きすぎて)，正しい演算結果が得られなかったこと.

オープンモード

ファイルオープンの方法のことで、読み込みモード，書き出しモードなどがある． fopen 関数に関係する.

関連項目： ファイル

オペランド

演算子の対象となるものをオペランドという.

改行

現在の表示位置を，次の行の先頭に移す動作のこと． C 言語では，文字列の中に「¥n」を埋め込むと，改行を意味する.

関連項目：改行文字

改行文字

改行動作を行わせるための特別な文字．C 言語では，文字列の中で「¥n」と書く

型

型は，データや関数の意味を決めるもの．C 言語では，代表的な型には，次のようなものがある.

- 配列型（array types）
- 文字型（character types）
- 浮動小数点数型（floating types）
- 関数型（function types）
- 整数型（integral types）
- ポインタ型（pointer types）
- 構造体型（structure types）

  など

空文字

ヌル文字 ともいう． C 言語では，「'¥0'」とも書く．

仮引数

仮引数（パラメータともいう）は， 関数言の中で宣言された変数のこと． 例えば，C 言語の関数宣言「int foo(int x);」では，オブジェクト x が宣言されている． 関数は，呼び出された時点において，引数を受け取り，当該関数の仮引数である変数にセットする．

引数（argument）は，関数などに実際に渡される「もの」のこと． 仮引数（parameter）は，宣言が行われた「もの」のこと．

関数

プログラムでの関数は，ある値の並び（引数）を与えると，何らかの仕事を行って，何らかの値が帰ってくるような，あるひとまとまりのプログラム部分のこと．

[関数呼び出しでの制御の流れ]

普通，文は逐次的に実行されるが， 関数呼び出しがあれば，関数を呼び出した後に，元の場所に戻る． C 言語での関数呼び出しでは，関数名に続いて，カッコ「()」を書き，カッコの中には引数を並べる．

[関数宣言]

関数呼び出しができるようにするために前もって宣言すること． C のプログラムでの関数宣言では，関数の返す型及び関数名のあとに，カッコ「()」を書き，カッコの中に仮引数を並べる（このとき，仮引数の型も書かれねばならない）． 仮引数とは，ここで宣言された関数が受け取るべき引数のこと．

[関数定義]

関数定義とは，関数の実体をプログラムとして記述すること． 次の C 言語の関数は，整数データを受け取って，1 足して返すだけの関数である． この例から分かるように，C 言語での関数定義は，「戻り値の型，関数名，（仮引数の並 び）{ ブロック }」 のように書く． return 文では，関数の戻り値を書く．

```
int foo( int n )
{
    return n + 1;
}
```

なお，関数は，何らかの返り値を返すものと，返さないものとがある． C 言語では， 返り値を返さない場合には，次のように void を使う． このとき，return 文には，単に「return;」と書く． この return 文は，関数の実行を終わって，呼び出し側に戻ることを意味する．

```
void foo2( int n )
{
    printf( "data = %d¥n", n );
    return;
}
```

関連事項： 値による呼び出し，参照による呼び出し

間接参照

ポインタを介して，何かを参照することを間接参照という．

偽

C 言語では，式の値が 0 のとき，偽(false)である．

関連項目：真

擬似乱数

C 言語の rand 関数を用いて得られる数字の系列は擬似乱数である． 乱数とも言う．このとき，srand を使って，擬似乱数の系列の設定を使わないと，毎回同じ系列の数字が得られる．

繰り返し

繰り返しとは，ある条件が満たされるまで，同じことを繰り返すこと．ループともいう． C 言語では，繰り返しを行うための文として while 文，for 文 などがある．

case ラベル

C プログラム での case ラベルは，後ろにコロン「:」が付いた識別子で， switch 文の中でのみ使うことができる．

関連項目：選択文

コーディング

ソースプログラムを作成すること． プログラミングというときは，プログラムの設計と作成全般を言い，コーディングというときは，所定の設計をもとに、設計通りのプログラムを作成するという意味でいうことが多い．

構造体

構造体は，名前が付いたいくつかのメンバの集まり．

関連項目：struct

コマンドラインシェル

コマンドラインシェルには bash (Bourne-Again shell)，Windows のコマンドプロンプト（cmd.exe）などがある． プログラムの起動，プログラムの終了，プログラムの出力のファイルへのリダイレクト，ヒストリ，ワイルドカード等を用いたパターンマッチ，ファイル名の補完などの機能がある．

コメント

コメントは，プログラムの中に書く注釈のこと．プログラムの説明や使用上の注意などを書く． コメントは，プログラム実行では無視される． C 言語では，コメントの始まりは「/*」で，終わりは「*/」である．あるいは「//」を使って，1行分のコメントを書くこともある． コメントは，入れ子にすることができない．次のコメントは入れ子になっていて，間違いである．

```
/* n = n + 1; /* n は「空き領域」の先頭を指す */ */
```

コンパイル

プログラムをコンパイルするとは，コンパイラを使って，プログラムのファイルを，コンピュータが実行可能な形式なファイルである実行形式ファイルなど，より解釈実行しやすい別のプログラムのファイルに変換すること．

再帰的呼び出し

再帰的呼び出しとは，自分自身を呼び出すような関数呼び出しのこと． 再帰的呼び出しをする関数の例として，ハノイの塔がある． ハノイの塔とは，柱 X,Y,Z があって，最初，数枚の円盤が柱 X にある（円盤は，大きいものが下になるようにおいてある）とき，「大きな円盤は小さな円盤よりも上に置かない」ことを守りながら，円盤を1枚ずつ移動させて，すべての円盤を柱 X から別の柱に動かす問題である．

次はハノイの等の C プログラムである.

```
int hanoi( int n, char x, char y, char z )
{
    if ( n < 1 ) {
        return 0;
    }
    hanoi( n-1, x, z, y );  /* N-1 枚の円盤を X から Y へ */
    printf( "円盤(直径:%d)を, %c から %c へ", n, x, z );
    hanoi( n-1, y, z, z );  /* N-1 枚の円盤を Y から Z へ */
}

main()
{
    hanoi( 4, "X", "Y", "Z" );
}
```

三角関数

代表的な三角関数は次の通り.

- acos : 逆コサイン
- asin : 逆サイン
- atan : 逆タンジェント
- cos : コサイン
- sin : サイン
- tan : タンジェント

算術

正の整数, 小数, 分数に関する計算のこと.

参照による呼び出し

関数の呼び出しにおいて, ポインタが渡されることを「参照による呼び出し」という. 関数の引数として, ポインタを渡すと, 関数の内部で, ポインタが指し示しているデータを読み書きすることがで

きるし， 関数を呼び出した側に戻った後に，呼び出し側で，関数内部での書き出し結果を読み込むことができる． 関数に渡されたポインタ（中身はメモリアドレス）は，関数の仮引数にセットされるが， 関数の中で，渡されたポインタを介して，値の読み書きができる．

関連項目：値による呼び出し

式

プログラミングでは，演算子とオペランドの並びのことを式といい，数学でいう式とは違う． 例えば，次に挙げるのは，C プログラムの式である．

```
30
x
x[100]
a+b
a*b
a++
a<100
c=a+b
sqrt( (x*x) + (y*y) )
```

関連項目：式文

式文

式のみからなる文を式文という．

識別子

C 言語では，識別子とは，関数，case ラベル，構造体，変数等に付けられる名前のこと． 識別子は，半角のアルファベット（小文字、大文字），半角のアンダーバー（_），半角の数字を使って書く．但し，識別子の先頭はアルファベットかアンダーバーでなければならない． また，C 言語の予約語を，識別子として用いることはできない．

自己参照構造体

自身へのポインタを持つような構造体のこと．

字下げ

プログラムで、行の先頭に，ある規則でタブや空白文字（半角の空白文字）を入れること． 次の C プログラムでは，if 文に続くブロックで，字下げが行われている．

```
#include <stdio.h>
main()
{
    int x;
    if ( x < 100 ) {
        printf( "x is small¥n");
    }
}
```

実行形式ファイル

機械語（マシン語）のプログラムが格納されたファイルで，コンピュータが実行可能な形式になっているファイルのこと．実行型ファイル，実行ファイル，実行可能ファイルのようにもいう．

ジャンプ文

他の場所へジャンプするための文． C プログラムでのジャンプ文には，continue 文，break 文，return 文などがある．

初期化

初期化とは，変数などのオブジェクトに，最初に値をセットすること． 初期化を忘れた変数は，どのような値になるかは，普通，前もって分からない C プログラムでの初期化は次の通り．

- 整数データの初期化

= に続いて，数値や式を書く．

```
int x = 10;
int y = 20;
```

- 浮動小数点数データの初期化

= に続いて，数値や式を書く．

```
double rate = 124.10;
```

- 構造体データの初期化

数値や式を並べて,「{}」でくくる.

```
struct person {
  int age;
  double height;
};
struct person p = {22, 171.20};
```

- 文字列データの初期化

文字列は,半角のダブルクオーテーション「"」でくくる. 次のように,[] と書いた場合には,文字列の終わりを示すヌル文字がつけられる. つまり,文字配列の長さは,文字列 "Please input data" とヌル文字とがぴったり入る大きさ である.

```
char message[] = "Please input data";
```

次のように,[32] と書いたときは,長さ 32 の文字配列であって,先頭部分に,"Please input data" とヌル文字が格納される.

```
char message[32] = "Please input data";
```

書式文字列

C プログラムでの書式文字列は, printf 関数, fprintf 関数, sscanf 関数などで,入出力のための変換方式を示す文字列のこと.

真

値 0 は,真(true)である

関連項目:偽

数字

数字は,次の 10 個

0 1 2 3 4 5 6 7 8 9

制御文

プログラムの実行順序を変えるための文を「制御文」のようにいう． C 言語では，制御文がない限り，文は逐次的に実行される． 制御文には，次の3種類がある．

1. 繰り返し： C プログラムでは while 文, for 文など
2. ジャンプ文： C プログラムでは continue 文, break 文, return 文など
3. 選択文： C プログラムでは if 文, switch 文

宣言

宣言を行う文では，識別子の名前と型を指定する． 次の C プログラムでは，age という名前の整数の変数の宣言を行っている．

```
  int age;
```

次の C プログラムでは， FILE へのポインタを宣言している．

```
FILE *fp;
```

次の C プログラムでは，配列の宣言を行っている．

```
int table[100];
```

次の C プログラムでは，関数を宣言している．

```
int max3( const int x, const int y, const int z )
{
    if ( ( x > y ) && ( x > z ) ) {
        return x;
    }
    else if ( y > z ) {
        return y;
    }
    else {
        return z;
    }
}
```

関連項目：初期化

選択文

式の値によって，実行すべき文を選ばせるための文． C プログラムの選択文には，if 文，switch 文などがある．

ソースファイル

プログラムが入ったテキストファイルのこと．

ソースプログラム

プログラムの文字列，あるいは，プログラムが入ったテキストファイルのこと． ソースコードともいう． プログラミング言語処理系で処理される．

注釈

注釈は，コメントともいう．

データ構造

データの集まりを，コンピュータで効率よく扱うための、データの形式のことである．データ構造には，配列，スタック，キュー，ハッシュテーブル、リスト，木，グラフなどがある．

テキストファイル

人間が見て読めるような文字だけで構成されたファイルのこと．

テスト

プログラムが所定通りに動作しているか検査し，プログラム中の不具合であるバグを発見すること．

デバッガ

プログラム中の不具合であるバグの発見と解決に役立つ機能を備えたプログラムのこと．

デバッグ

プログラム中の不具合であるバグの発見と解決を行うこと．

デクリメント

値を1減らすことをデクリメントという.

関連項目： インクリメント

動的メモリ管理

プログラムの実行時にメモリブロックを確保，解放すること. new 演算子，delete 演算子などを用いる.

NULL ポインタ

NULL ポインタは，どこも指していないポインタのこと.

ヌル文字

文字列の末端を表す特別な文字のこと.

バイト

バイトはデータの基本単位. 基本文字（英文字，英記号，数値など）を 1 つ表現できるだけの大きさ.

配列

配列とは，同じ型を持ったデータが，メモリ上に連続して並んでいるようなデータの集まりのこと. 例えば，C プログラムの「int table[100];」では，整数データが，メモリ上に100個連続して並んでいる. このとき配列の要素には，0から99までの番号がふられている.

```
int table[100];
```

関連項目： 配列の宣言

配列の宣言

配列の宣言では，配列，名前，要素の数，及び配列の要素になるデータの型を指定する. 例えば，C プログラムの「int table[100];」では，table という名前で， int 型の要素を 100 個持った配列の宣言を行う.

```
int table[100];
```

配列の宣言では，複数個の[]を付けて，宣言することもできる. このような場合を多次元の配列という.

```
int table2[10][20];
```

関連項目： 配列

ハノイの塔

ハノイの塔については，「再帰的呼び出し」の項を見よ．

引数

関数呼び出しにおいて，関数に対して渡される値あるいは式．

関連項目：仮引数

評価

式から，その値を求めることを評価という．

標準出力

標準出力は，普通はディスプレイである．設定を行うことによって，標準出力をファイル等に切り替える（つまりディスプレイの替わりに，ファイルに書き出す）ことが可能である． C プログラムでは，標準出力は，stdout と書く．

標準入力

標準入力は，普通はキーボードである．設定を行うことによって，標準入力をファイル等に切り替える（つまりキーボードの代わりに，ファイルから読み込む）ことが可能である． C プログラムでは，標準入力は， stdin と書く．

ファイル

ファイルは，ディスクなどに保存されるデータ． オペレーティングシステムが，ファイルを管理する．ファイルを読み込み，書き出しする前と終わった後には，オペレーティングに通知（オープン，クローズ）しなければならない．

[ファイル位置指示子]

ファイル位置指示子は，ファイル内で現在ファイルが読み書きされている位置を示す． ファイルをオープンした直後は，ファイル位置指示子は，ファイルの先頭を指し示している． ファイルの読み込み，書き出しを行うと，ファイル位置指示子は動いていく． ファイル位置指示子を操作することによって，ファイル内の読み書き位置を移動し，ファイル内の好きな位置を読み書きできるようにな

る． C プログラムでは，ファイル位置指示子は，FILE オブジェクト内に記憶される． C プログラムでは，ファイル位置指示子の操作には，fseek 関数，ftell 関数等を用いる．

[ファイル名]

ファイルの名前のこと． C プログラムでは，fopen 関数を使ってファイルをオープンするときには，ファイル名を指定せねばならない．

[ファイルのオープン]

プログラムからファイルの読み書きを行う前に，ファイルはオープンされねばならない． C プログラムでは，ファイルをオープンするには，fopen 関数等を用いる．

[ファイルのクローズ]

ファイルの読み書きが終わったら，ファイルをクローズせねばならない． C プログラムでは，ファイルをクローズするには，fclose 関数等を用いる．

[ファイルの読み込み]

ファイルを読んで入力すること（つまりファイルからプログラムにデータを読み込むこと）．

[ファイルの書き出し]

ファイルへ書いて出力すること（つまりプログラムからファイルへデータを書き出すこと）．

[ファイルを扱うライブラリ関数]

ファイルを扱うライブラリ関数としては次のようなものがある．

- オープン／クローズ
  - fclose ファイルをクローズする
  - fopen ファイルをオープンする
- ファイル位置指定
  - fseek ファイル位置指示子をセットする
  - ftell ファイル位置指示子の値を得る
- 入出力
  - fgetc 1文字読み込む
  - fgets 1行読み込む
  - fputc 1文字書き込む
  - fputs 1行書き込む

- 書式文字列付き
  - fprintf 書式文字列を指定してファイルを書く
  - printf 書式文字列を指定して書く

FILE オブジェクト

C プログラムでは，FILE オブジェクトには，ファイル等を操作するためのすべての情報が収められている．例えば，ファイル位置指示子も FILE オブジェクト内に記憶されている． FILE オブジェクトへのポインタを，ファイルポインタ(file pointer)ともいう．

複文

複文は，ブロックともいう．

プログラム

コンピュータはプログラムで動く．プログラムには，オペレーティングシステムなどのシステムプログラムと，さまざまな用途ごとに使うアプリケーションプログラムの2種類がある．

プログラミング

プログラミングとは、プログラムを作ること．

プログラミング言語

プログラミング言語とは，プログラムを書くための言語のことである．プログラミング言語には，Python, JavaScript, C, C++, Java, bash, C#, R, MATLAB, Octave, SQL, Ruby, LISP, Prolog, Perl, Haskell, Scheme，アセンブリ，機械語（マシン語） などたくさんの種類がある．

プログラミング言語処理系

プログラミング言語処理系には，プログラムを解釈実行する処理系であるインタプリタ，プログラムをより解釈実行しやすい別のプログラムに変換するコンパイラがある．

プログラミングスタイル

C プログラムでは， if 文の後に出てくる「{」について，次のように書いてもよいし，

```
if (x < 0) {
```

```
    printf( "negative¥n" );
    return -1;
}
else {
    return sqrt(x);
}
```

あるいは次のように書いてもよい.

```
if(x < 0)
{
    printf( "negative¥n" );
    return -1;
}
else
{
    return sqrt(x);
}
```

次のように書くと(文の字下げがばらばら)読みづらいということが分かるはず. ブロックの中では, 式は同じ分だけ字下げするように心がけよう.

```
if(x < 0) {
printf( "negative¥n" );
  return -1;
}
else {
 return sqrt(x);
}
```

また, 次のように, カッコの字下げがばらばらだと, プログラムの意味が分かりづらくなる. カッコの対応が取りやすいように書くことも大事である.

```
if(x < 0) {
    printf( "negative¥n" );
    return -1;
    }
```

```
    else {
    return sqrt(x);
}
```

プロセッサ

プロセッサは，機械語（マシン語）のプログラムを解釈実行できる機能を持つ．

ブロック

C 言語でのブロックは，「{}と「」」で囲まれた文の並び（1つ以上の文）． ブロックは，関数の定義， if 文, switch 文, for 文, while 文 などで使われ る． 次のプログラムでは，main 関数の本体が 1 つのブロックになっている． ブロックは，入れ子になることがある．

```
int main()
{
  printf( "hello world¥n" );
  return 0;
}
```

文

C 言語では，文には次のような種類がある．

- ブロック(compound statement)
- 式文(expression statement)
- 繰り返し文(iteration statement)
    - for 文
    - while 文
      など
- ジャンプ文(jump statement)
    - break 文
    - continue 文
    - return 文
      など
- ラベル付き文(labeled statement)
    - case
    - default

- 空文(null statement)
- 選択文(selection statement)
    - if 文
    - switch 文

変数

変数は，名前の付いたオブジェクトのこと。 数学で 言う変数とは違う.

[変数宣言]

変数宣言とは，変数の名前，型などを書いて，プログラム中での変数の使用を宣言すること． C 言語での変数宣言は，最後をセミコロンで終える． 次の C プログラムでは，変数 x は int 型（整数データ）であり， 変数 rate は double 型（浮動小数点数データ）である.

```
int x;
double rate;
```

変数宣言と同時に，変数の値を初期化することもできる． 次の C プログラムでは，変数宣言と初期化を同時に行っている.

```
int i = 0;
```

変数宣言をブロック（「{}」で囲まれた文の並びのこと）」の中 で行ったとき，宣言された変数は，当該ブロックの中でのみ使えることに注意せよ.

ポインタ

ポインタという言葉は，他のもの（整数，浮動小数点数，配列，構造体，ポインタ，関数など）を指し示すということに由来する． ポインタを使って，好きなデータの読み書きができる．但し，ポインタの中身が正しくないと，データを壊してしまうことになる． ポインタの中身は，メモリアドレスである.

[ポインタの宣言]

C プログラムでは，ポインタの宣言には，「*」を用いる． 次のように「int *p;」と書くと，整数データへのポインタが宣言される． p の中身は，整数データが入っているメモリアドレスである.

```
int *p;
```

C プログラムでは，ポインタ自身も1種の変数に格納されている(ポインタ変数という)に格納されているから，「int **table;」のように書いて，ポインタへのポインタを宣言することもできる．

[ポインタ型]

「int へのポインタ」を，C プログラムで書くと次の通り．

```
int *p;
```

このとき，p は，int 型へのポインタ型である．

メモリ

メモリは，データやプログラムを記憶するための装置のこと． メモリは，ある長さのメモリブロックが並んだ形になっていて，それぞれのメモリブロックにアドレスがつけられている．

メンバ

メンバとは，構造体のそれぞれの要素につけられた名前のこと． 次の C プログラムでは，age と height がメンバである．メンバの値を操作するには，ピリオド「．」あるいは，ー>を使う．

```
struct person {
  int age;
  double height;
};
```

文字型

文字データを扱うための型．次のプログラムでは，c は文字型．

```
char c;
```

文字列データ

文字列データを扱うための最も簡単な方法は，文字の配列を使う方法である． C プログラムでは，文字の配列に，文字を1文字ずつ入れ，最後に，ヌル文字を付ける． 最後のヌル文字は，「ここで文字列が終わること」を示すための記号である． 但し，「文字列の長さ」というときには，最後のヌル文字を数えない．

C プログラムでの，文字列を操作するライブラリ関数の代表的なものを次に挙げる．

- 文字列比較
  - strcmp 2つの文字列を比較する
- 文字列連結
  - strcat 2つの文字列を連結する
- 文字列コピー
  - strcpy 文字列をコピーする
- 文字列関数
  - strlen 文字列の長さを得る

など

予約語

予約語は，C プログラムでは，特別な意味があらかじめ定められているため，他の目的に使うことはできない．つまり，予約語を変数名や関数名として用いることはできない．

asm, auto, break , case , catch, char , const , continue , default , delete, do, double , else , enum, extern , float, for , goto, if , int , long, new, operator, private, protected, public, register, return , short, signed, sizeof, static, struct , switch , template, this, throw, try, typedef, union, unsigned, virtual, void , volatile, while

ループ

一連の文の実行を繰り返すこと．

関連項目：繰り返し，繰り返し

C プログラムの記号・キーワード等

!

！[オペランド]

！は否定演算子． オペランドが 0 なら偽，0 以外なら真である． つまり， オペランドが 0 でなければ，！[オペランド]は 0 になる． オペランドが 0 であれば，！[オペランド]の値は 0 でない値になる．

関連項目： !=

!=

[オペランド１]！=［オペランド２］

！= は，比較を行う演算子． オペランド1とオペランド2を比較し，両者が等しいときには真を， 等しくないときには偽を返す．

関連項目： !, <, <=, ==, >, >=,

″

二重引用符「″」は，文字列の始まりと終わりをあらわす．

#include

#include は，C コンパイラ（正確には C コンパイラの中のプリプロセッサ）に対して，プログラムのコンパイル時に#include の後で指定されたファイルを取り込むように指示する．

%

[オペランド１]%[オペランド２]

%は，整数の剰余を得る演算子． オペランド1を，オペランド2で割ったとき（整数の除算）の剰余を返す． 浮動小数点数の剰余を得るには， fmod 関数を使うこと．

関連項目： /

&

&[オペランド]

&はメモリアドレスを求めるための演算子.　オペランドのメモリアドレスを返す.

&&

[オペランド１]&&[オペランド２]

&&は論理積を求める演算子.　オペランド1とオペランド2の論理積を求める.　オペラント1とオペランド2のいずれか，あるいは両方が 0 のとき，［オペランド1]&&[オペランド2]の値は 0 になる.　オペラント1とオペランド2の両方が 0 でないとき，［オペランド1]&&[オペランド2]の値は 0 でない値になる.

関連項目：||

()

[関数名]（），あるいは（[式]）

()は，区切り記号.　関数名の後に()がくると、()の中には，引数のならびが入る.　式を()でくくったときには，その式を先に評価することを意味する.

*

[オペランド１]＊[オペランド２]
＊[ポインタ]
[タイプ名]＊[変数名（の並び）]；

「*」は乗算を行う演算子，あるいは間接参照演算子，あるいはポインタ区切り文字として使う.

- 2つのオペランドの間に＊がくると，乗算を行う演算子．オペランド1とオペランド2の積を返す.
- 「*」の次にポインタがくると，そのポインタがさす内容を取り出す（これを間接参照演算子という）.
- 宣言した変数をポインタであることを示すために，「*」を変数宣言で使う.

例：FILE* fp;

*/

＊／はコメント区切り．コメントの終わりを示す.　コメントは注釈（プログラムの説明や注意事項などを書く）のことで，プログラムの実行に影響を与えない.

関連項目： /*, //

+

［オペランド１］＋［オペランド２］

「+」は加算を行う演算子． オペランド1とオペランド2の和を返す．

関連項目： ++, −

++

［オペランド］++
++［オペランド］

「++」はインクリメントを行う演算子． インクリメントとは1足すこと． オペランドに1を足して，オペランドに格納する． 「［オペランド］ = ［オペランド］ + 1;」と同じ．

関連項目： −−

−

［オペランド１］－［オペランド２］

「−」は減算を行う演算子． オペランド1からオペランド2を引いて，その値を返す．

関連項目： +, −−

−−

［オペランド］ −−
−− ［オペランド］

「−−」はデクリメント演算子． デクリメントとは1引くこと． オペランドから1を引いて，オペランドに格納する． 「［オペランド］ = ［オペランド］−1;」と同じ．

関連項目： ++

->

［ポインタ］ー>［メンバ名］

「->」は，メンバを参照するための演算子． 構造体などへのポインタを使って，そのメンバを参照する．

例：

```
struct Person {
  int age;
  char name[32];
};
struct Person p;
struct Person ptr;
ptr->age = 32;
```

.

[名前].[メンバ名]

「.」は，メンバを参照するための演算子． 構造体名を使って，そのメンバを参照する．

例：

```
struct Person {
  int age;
  char name[32];
};
struct Person p;
p.age = 32;
```

関連項目： ->

/

[オペランド１] / [オペランド２]

「/」は除算を行う演算子． オペランド1をオペランド2で割って，その商を返す．

関連項目： %

/*

「/*」はコメント区切り．コメントの始まりを示す．「*/」までがコメント．

関連項目： */, //

//

「//」はコメント区切り. 「//」から行末までがコメントであることを示す.

関連項目： /*, */

;

「;」は区切り文字. 「;」は文と文の区切りを示す.

<

［オペランド１］<［オペランド２］

「<」は比較を行う演算子. オペランド1がオペランド2よりも小さい場合，真を返し， オペランド1がオペランド2と等しいか大きい場合，偽を返す.

関連項目： !=, <=, ==, >, >=

<=

［オペランド１］ <= ［オペランド２］

「<=」は比較を行う演算子. オペランド1がオペランド2と等しいか小さい場合，真を返し， オペランド1がオペランド2よりも大きい場合，偽を返す.

関連項目： !=, <, ==, >, >=

=

［オペランド１］= ［オペランド２］

< は代入演算子. オペランド2の値を，オペランド1に代入する.

==

［オペランド１］ == ［オペランド２］

「==」は比較を行う演算子. オペランド1がオペランド2と等しい場合に限り，真を返し， オペランド1がオペランド2と等しくない場合，偽を返す.

関連項目： !=, <, <=, >, >=

>

［オペランド１］ > ［オペランド２］

「>」は比較を行う演算子． オペランド1がオペランド2よりも大きい場合，真を返し， オペランド1がオペランド2と等しいか小さい場合，偽を返す．

関連項目： !=, <, <=, ==, >=

>=

［オペランド１］ >= ［オペランド２］

「>=」は比較を行う演算子． オペランド1がオペランド2と等しいか大きい場合，真を返し， オペランド1がオペランド2よりも小さい場合，偽を返す．

関連項目： !=, <, <=, ==, >

||

［オペランド１］ || ［オペランド２］

「||」は論理和を求める演算子． オペランド1とオペランド2の論理和を求める． 各オペランドが0なら偽，0以外なら真であって，オペラント1とオペランド2のいずれかが0以外のときに，[オペランド1] || [オペランド2]は真を返す．

関連項目： &&

math.h

#include <math.h>

math.h は数学関係の定義と宣言が収められたファイルのこと． 指数対数関数，双曲線関数，整数剰余関数，指数／対数関数，三角関数などが含まれている．

stdio.h

#include <stdio.h>

stdio.h は入出力等に関する定義と宣言が収められたファイルのこと． エラー処理，ファイルのオープン／クローズ／各種の操作，ファイルの位置指定，文字入出力，書式文字列付き入出力などが含まれている．

stdlib.h

```
#include <stlib.h>
```

stdlib.h は一般操作に関する定義と宣言が収められたファイルのこと． メモリブロックの確保と解放，擬似乱数，プログラムの終了処理，文字列から数値への変換，マルチバイト文字関係の処理などが含まれている．

string.h

```
#include <string.h>
```

string.h は文字列操作等に関する定義と宣言が収められたファイルのこと． 文字列の連結，文字列の比較，文字列のコピー，文字列の検索などが含まれている．

abs()

```
#include <stdlib.h>
int abs(int n);
```

abs 関数は，整数の絶対値を求めるライブラリ関数．

acos()

```
#include <math.h>
double acos(double arg);
```

acos 関数は，逆コサインを計算するライブラリ関数． 返される値の単位はラジアン．

関連項目： asin , atan , cos , sin , tan

asin ()

```
#include <math.h>
double asin(double arg);
```

asin 関数は，逆サインを計算するライブラリ関数． 返される値の単位はラジアン．

関連項目： acos , atan , cos , sin , tan

atan ()

```
#include <math.h>
double atan(double arg);
```

atan 関数は，逆タンジェントを計算するライブラリ関数． 返される値の単位はラジアン．

関連項目： acos , asin , cos , sin , tan

break

ループを中断するための文． break を含む最も内側の switch 文あるいはループ文(while , for 文など)から抜け出す．

関連項目： continue , return

case

```
case [式];
```

switch 文とともに使う． switch で書かれた式の値は，各 case ラベルの値を比較され，一致した case ラベルに，プログラムの制御が飛び，break 文まで実行が行われる．

関連項目： case ラベル, break , default , switch

char

char は，文字データを扱うためのデータ型． 文字(数値，英字，英記号)は 1 バイトで表現できる(1 バイトは 8 ビットあって，256 通りの文字をコード化可能である)．

const

const は，値の変更が行われないことを示す予約語． const は， int , double などの型名と一緒に使う． 関数は，データ値を変数で受け取るが，関数の中でその変数の値を変更しないことが多い． そういうときに const がよく用いられる． C のライブラリ関数などでは，関数の中で変数の値を変更しない場合，関数での引数宣言に const が付いていることが多い．

continue

次のループ実行を行うための文． continue を含む最も内側のループ文(while 文, for 文など)について，ループの本体の残りを飛ばして，次の繰り返しを始める．

関連項目： break , return

cos ()

```
#include <math.h>
double cos(double arg);
```

cos 関数は，コサインを計算するライブラリ関数．引数の単位はラジアンである．

関連項目： acos , asin , atan , sin , tan

default

switch とともに使う． switch で書かれた式の値は，各 case ラベルの値を比較されるが，どの case ラベルの値とも一致しない場合には，default 部分に，プログラムの制御が飛び， break 文まで実行が行われる．

関連項目： break , switch

double

double は，浮動小数点数データを扱うためのデータ型． 浮動小数点数のデータとは，精度が 10 桁で，10 の －37 乗から 10 の +37 乗までの正と負の範囲数のこと． コンピュータの種類によっては，精度が 10 桁以上であったり，範囲が 10 の －37 乗から 10 の +37 乗よりも広い範囲を扱える場合もある．

else

```
else [文];
```

if 文は，else と組合わせることができる． 次の例では，x が負のとき，メッセージ「x is negative」が表示され， x が正またはOのとき，x の平方根が計算されます．

```
if ( x < 0 ) {
    printf( "x is negative¥n" );
}
else {
    y = sqrt( x );
```

}

if 文で指定された条件が 0 なら偽，0 以外なら真であって， if 文での条件が 0 のとき，else 以下の文が実行される． if 文での条件が 0 以外のときは，else 以下の文が実行されない．

関連項目： if , switch

EOF

#include <stdio.h>

EOF は、ファイルの終わりを示す． ファイルの入力において，これ以上の入力の無いことを示したり， 入出力操作においてエラーが発生したことを示す．

関連項目： stdio.h

exit()

```
#include <stlib.h>
void exit(int status);
```

exit 関数は，プログラムを終了させるためライブラリ関数． exit 関数を実行すると，プログラムは終了する．

exp()

```
#include <math.h>
double exp(double z);
```

exp 関数は，指数関数の計算を行うライブラリ関数． e を底とする指数 z の累乗(e の z 乗)を計算する． 対数関数 log の逆関数である．

関連項目： log

extern

extern [タイプ名] *[識別子];

extern は，指定された識別子の定義が外部で行われていることを示す．

fclose()

```
#include <stdio.h>
int fclose(FILE *fp);
```

fclose 関数は，ファイル等のクローズを行うためのライブラリ関数．

関連項目： fopen

fgetc()

```
#include <stdio.h>
int fgetc(FILE *fp);
```

fgetc 関数は，ファイル等から1文字読み込むためのライブラリ関数． fgetc 関数を実行すると，入力された文字が返される． 読み込み時点で，ファイルの終わりに達していたら EOF が返される． また、もし，ファイルからの読み込み時にエラーが発生したときには， EOF が返される．

関連項目： fgets , fputc , fputs

fgets()

```
#include <stdio.h>
int fgets(char *string, int n, FILE *fp);
```

fgets 関数は，ファイル等から文字列を読み込むためのライブラリ関数． ファイル等から，引数 n 文字未満の文字を読み込んで，引数 string に格納される． fgets 関数は，基本的には1行分だけ文字を読み込み，最後に文字列の末尾を示すヌル文字('¥0') を付け加える． 読み込むときは改行文字までを含めて読み込む． 但し，1行が長くて，読み込む文字が n-1 文字になったり，あるいは行の途中でファイルの終わりに達した場合には，読み込みを止めて， 最後に文字列の末尾を示すヌル文字('¥0')を付け加える． もし，ファイルからの読み込み時にエラーが発生したときには， NULL が返される．

関連項目： fgetc , fputc , fputs

FILE

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

FILE オブジェクトは，ファイル等を扱うために必要なもの． 現在の読み書き位置，ファイルの状態を示す情報を含む． fopen 関数でファイルをオープンすると，FILE オブジェクトへのポインタが得られる． こうして得られた FILE オブジェクトへのポインタは，後のファイル読み書き，ファイルのクローズのために使用する．

fmod()

```
#include <math.h>
double fmod(double x, double y);
```

fmod 関数は，剰余を得る演算子． 引数 x を引数 y で割ったときの剰余を返す．

関連項目： %

fopen()

```
#include <stdio.h>
int fopen(const char *file, const char *mode);
```

fopen 関数は，ファイルをオープンするためのライブラリ関数． ファイル名とモードを指定してオープンを行う． 引数 file には，オープンすべきファイル名を指定する． 引数 mode には，"r"，"w" などのオープンモードを指定する．

- "r" モード
  読み込みモード． 引数 file で指定したファイルが存在しないか，読み込み不可能な場合には，オープンすることができない．
- "w" モード
  書き出しモード． 引数 file で指定したファイルが存在しない場合には，ファイルが新たに作成される． ファイルがすでに存在した場合，ファイル中のデータはすべて捨てられる（ファイルの長さは0になる）．

関連項目： fclose

for

for（ 初期式； 条件式； 再設定式） 文

for は，何かの処理の繰り返しのために使う． for は，例えば for(i = 0; i < 100; i++) のように，3 つの式を書く． 最初の式は，初期式で，繰り返しの最初に1回だけ実行される． 2 番目の式は，条件式で，繰り返しのたびに，真偽が判定される（偽ならば繰り返しが終わる）． 3 番目の式は，繰り返しのたびに実行される． for での繰り返しから抜け出すには break 文を使用する．繰り返し途中で， 残りの処理を飛ばすには continue 文を使う．

「for(初期化; 条件; 繰り返し） 文」は，次と等価である．

初期式

```
while ( 条件式 ) {
    文
    再設定式
}
```

関連項目: break , continue , while

fprintf

```
#include <stdio.h>
int fprintf( FILE* fp, const char *format, …);
```

fprintf 関数は, 指定された書式文字列(2番目の引数の format のこと)で, ファイル等への出力を行う. fprintf 関数を使って, メッセージ及び整数, 浮動小数点数, 文字, 文字列などのデータの出力を行うことができる. fprintf 関数の引数である書式文字列には, データを出力を行うために, %付きの文字を書く(これについては, printf の項を参照). fprintf 関数を使って, 「%」を出力したいときには, 書式文字列中で「%%」のように書くこと.

関連項目: printf , sscanf

fputc()

```
#include <stdio.h>
int fputc(int c, FILE *fp);
```

fputc 関数は, ファイル等に, 1文字書き込む. もし, ファイルへの書き出し時にエラーが発生したときには, EOF が返される.

関連項目: fgetc , fgets , fputs

fputs()

```
#include <stdio.h>
int fputs(const char *string, FILE *fp);
```

fputs 関数は, ファイル等に文字列を書き込む. 文字列の末端に付いた, ヌル文字 は出力しない. もし, ファイルへの書き出し時にエラーが発生したときには, EOF が返される.

改行文字の扱いは次の通り.

- gets: 改行文字までを読み込むが, 改行文字は捨てる.

- fgets : 改行文字までを読み込み，改行文字までを含めて出力する.
- puts: 末尾に改行文字を自動的に付けて書き込む.
- fputs: 末尾に改行文字を付けない.

関連項目： fgetc , fgets , fputs

free()

```
#include <stdlib.h>
void free(void *ptr);
```

free 関数は malloc 関数等で動的に確保したメモリブロックの解放を行うためのライブラリ関数.

関連項目： malloc

fseek()

```
#include <stdlib.h>
void fseek(FILE *fp, long int offset, int where);
```

fseek 関数は，ファイルの読み書き位置を変更するためのライブラリ関数. 2 番目の引数の offset には，移動させる量をバイト単位で示す. 進めるときは正の数を，戻すときは負の数を指定する. 3 番目の引数の where には，移動すべき原点を指定する. 次の 3 通りがある.

1. SEEK_CUR 現在の読み書き位置
2. SEEK_END ファイルの終端
3. SEEK_SET ファイルの先頭

関連項目： ftell

ftell()

```
#include <stdlib.h>
long int ftell(FILE *fp);
```

ftell 関数は，現在のファイル読み書き位置を知るためのライブラリ関数. ftell 関数を実行すると，普通，先頭から何文字目かの数値が返される.

関連項目： fseek

if

if (条件式) 文

if は，条件分岐のために使う． 次の例では，x が負のときに限り，メッセージ「x is negative」が表示され る．

```
if ( x < 0 ) {
    printf( "x is negative¥n" );
}
```

if 文で指定された条件式が 0 なら偽，0 以外なら真であって， if 文での条件が真のときは，if 以下の文が実行される． if 文での条件が偽のとき，if 以下の文は実行されない．

if 文の後に else を書いて， else の後にさらに文を書くことができる． このような場合には， if 文での条件が 0 以外のときは，if 以下の文だけが実行され， if 文での条件が 0 のとき，else 以下の文だけが実行される．

関連項目： switch

int

int は，整数データを扱うためのデータ型． 整数データとは，-32768 から +32767 までの範囲の数のことで， 数学での整数では無い． コンピュータの種類によっては，-32768 から +32767 よりも広い範囲を扱える場合もある．

log()

```
#include <math.h>
double log(double z);
```

log 関数は，底を e とする自然対数の計算を行うライブラリ関数． 対数関数 exp 関数の逆関数である．

関連項目： exp

main

main 関数は，プログラム実行の開始時に自動的に呼び出される関数のこと． プログラムには，必ず１つの main 関数が含まれていなければならない． 下記の例では，main 関数の仮引数は「int argv, char *argv[]」（整数データ argv と，文字列の配列 argv）． main 関数の戻り値は，「int main(...」とあるように int 型． main 関数の最後の方の「return 0;」で，戻り値として0を返している．

```
#include <stdio.h>

int main( int argv, char *argv[] )
{
    printf("Hello, world¥n");

    return 0;
}
```

malloc()

```
#include <stdlib.h>
void *malloc(size_t size);
```

malloc 関数は，メモリブロックの確保を行うためのライブラリ関数． 引数の size（単位はバイト）で指定された大きさのメモリブロックが割り当てられる．

関連項目： free

NULL

NULL は，ヌルポインタ（値として，特別な値0を持つポインタのことで，どこも指し示していないポインタのことである）． NULL は，「(void *)0」と同じ． fgets 関数，fopen 関数，malloc 関数などでは，操作に失敗したときに NULL が返される．

printf

```
#include <stdio.h>
int printf( const char *format, …);
```

printf 関数は，指定された書式文字列（1番目の引数の format のこと）で，標準出力（標準出力は端末に結びついている．Microsoft Windows では，画面に表示されるウインドのこと）への出力を行う． printf 関数を使って，メッセージ及び整数，浮動小数点数，文字，文字列などのデータの出力を行うことができる． printf 関数の引数である書式文字列には，データの出力を行うために，%付きの文字を書く． printf 関数を使って，「%」を出力したいときには，書式文字列中で「%%」のように書くこと．

%c　　文字の表示
%d　　int 型のデータを，10 進数で表示
%f　　double 型のデータの表示（参考：「%f」で float 型のデータの表示も可能）

%p　　ポインタの表示
%s　　文字列の表示
%x　　int 型のデータを，１６進数で表示

%10d, %10f, %10s のように数字を書くことができ，表示のための幅を示す． このようにして，表示幅を指定したとき，もし，データ(数値や文字)が指定した幅より大きければ，表示が崩れる(データはすべて表示される)．

%10.4f のように，% と f の間に小数付きの数を書くことができ， 表示のための幅(ここでは10)と，小数点以下何桁まで表示すべきか(ここでは4)を示す．

関連項目： fprintf , sscanf

rand()

```
#include <stlib.h>
int rand(void)
```

rand 関数は擬似乱数(pseudo-random number)を発生させるためのライブラリ関数． 発生される数は，0から RAND_MAX の間の値をとる． rand 関数は，シードの設定を行わないと，同じ系列の擬似乱数を返す．

関連項目： RAND_MAX , srand

RAND_MAX

```
#include <stlib.h>
```

RAND_MAX は， rand 関数で発生する擬似乱数(pseudo-random number)の最大値を表す．

関連項目： rand , srand

return

```
return ;
または
return [式] ;
```

return は，呼び出し側の関数に戻るために使う． return を実行すると，return を含む関数の実行を直ちに終えて，呼び出し側の関数に戻る． return の後に式を書くことができて(そのために

は関数の宣言が int foo(…)や double foo2(…) のように，型を付けて行われていなければならない)，その場合には，式の値が，呼び出し側の関数に渡される．

関連項目： break , continue

SEEK_CUR

SEEK_CUR は，ファイル位置指示子の移動を，現在の読み書き位置を基準にして行うことを意味する． fseek などで用いる．

関連項目： ファイル, fseek , SEEK_END , SEEK_SET , stdio.h

SEEK_END

SEEK_END は，ファイル位置指示子の移動を，ファイルの終端を基準にして行うことを意味する．fseek などで用いる．

関連項目： ファイル, fseek, SEEK_CUR , SEEK_SET , stdio.h

SEEK_SET

SEEK_SET は，ファイル位置指示子の移動を，ファイルの先頭を基準にして行うことを意味する．fseek などで用いる．

関連項目： ファイル, fseek, SEEK_CUR , SEEK_END , stdio.h

sin ()

```
#include <math.h>
double sin(double arg);
```

sin 関数は，サインを計算するライブラリ関数．引数の単位はラジアンである．

関連項目： acos , asin , atan , cos , tan

size_t

size_t は，各種のデータのサイズを表す型である．

sqrt ()

```
#include <math.h>
```

```
double sqrt(double x);
```

sqrt 関数は，平方根を計算するライブラリ関数.

srand()

```
#include
void srand(unsigned int seed);
```

srand 関数は， rand 関数で発生させる擬似乱数(pseudo-random number)の系列を設定するためのライブラリ関数． 擬似乱数の系列は， srand 関数の引数 seed によって変化する.

関連項目： rand , RAND_MAX

sscanf

```
#include <stdio.h>
int sscanf( const char *string, const char *format, …);
```

sscanf 関数は，引数 string で指定された文字列から， 引数 format で指定された書式文字列で，データを読み込む． sscanf 関数を使って，整数，浮動小数点数，文字，文字列などのデータの読み込みを行うことができる． sscanf 関数の引数である書式文字列には，データの読み込みを行うために，%付きの文字を書く． format の後には，読み取ったデータを格納すべき変数の並びを書くが，必要に応じて，変数の頭には&を付けること.

%c　　文字の読み込み
%d　　int 型のデータ（10 進数）の読み込み
%lf　　double 型のデータの読み込み(参考：「%f」は float 型のデータの読み込み)
%s　　文字列の読み込み
%x　　int 型のデータ（16 進数）の読み込み

関連項目： printf , fprintf

stdin

stdin は、プログラムが入力を行う先のことで，Microsoft Window では，キーボードのこと.

関連項目： stdio.h

strcat()

```
#include <string.h>
```

```
char *strcat(char *string1, const char *string2);
```

strcat 関数は，文字列の連結を行うためのライブラリ関数． strcat 関数を実行すると，引数の string2 が(string2 の末尾のヌル文字までを含めた全ての文字)が，srting1 の末尾につながってコピーされる． コピーの前にあった string1 の末尾のヌル文字は，string2 で上書きされる．

strcmp()

```
#include <string.h>
char *strcmp(const char *string1, const char *string2);
```

strcmp 関数は，文字列の比較を行うためのライブラリ関数． strcmp 関数を実行すると，引数 string1 と引数 string2 との比較が辞書順で行われる．

- string1 の方が string2 より辞書順で前： strcmp 関数は負の値を返す
- string1 と string2 が一致： strcmp 関数は 0 を返す
- string1 の方が string2 より辞書順で後： strcmp 関数は正の値を返す

strcpy()

```
#include <string.h>
char *strcpy(const char *string1, const char *string2);
```

strcpy 関数は，文字列のコピーを行うためのライブラリ関数． strcpy 関数を実行すると，引数 string2 が(末尾のヌル文字まで含めたすべての文字)，srting1 にコピーされる．string2 が上書きされるので，コピー前の string1 は消える．

strlen()

```
#include <string.h>
char *strlen(const char *string);
```

strlen 関数は，文字列の長さを得るためのライブラリ関数． strlen 関数を実行すると，引数 string の文字数を数えるが，末尾のヌル文字は含めない．

struct

struct は，構造体を表す．構造体とは，1つ以上のメンバを含むデータのことで，各メンバは異なった名前と型をもつ．

switch

switch ([式]) [文]

switch 文は，条件分岐のために使う． switch 文で指定された式の値が，switch 文に続く 各 case ラベルの値と比較され，一致した case ラベルに，プログラムの制御が飛び， break 文まで実行が行われる．

```
#include <stdio.h>

int main ()
{
    int month;
    char buf[256];

    gets( buf );
    sscanf( buf, "%d", &month );

    if ( ( month < 1 ) || ( month > 12 ) ) {
        printf( "month is invalid¥n" );
    }

    switch ( month ) {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            printf( "31¥n" );
            break;
        case 2:
            printf( "28 or 29¥n" );
            break;
        case 4:
        case 6:
        case 9:
        case 11:
```

```
            printf( "30¥n" );
            break;
    }
}
```

関連項目： break , case , default , if

tan()

```
#include <math.h>
double tan(double arg);
```

tan 関数は，タンジェントを計算するライブラリ関数．引数の単位はラジアンである．

関連項目： acos , asin , atan , sin , cos

void

void は，関数が値を返さないこと，関数が引数を持たないこと，あるいは汎用のポインタなどのために用いられる．

- 関数の型
  関数が値を返さないことを示すために void を用いる．例えば，
  void foo(int age);
- 関数の引数
  関数が引数を持たないことを示すために void を用いる．例えば，
  int foo2(void);
  関数が引数を持たないことを示すとき， void を省略して， int foo2();
  と書くこともある．
- void*
  void* は，汎用のポインタである．

関連項目： NULL

while

while(条件) 文

while は，何かの処理の繰り返しのために使う． while では，繰り返しのたびに while 文で書かれた条件の真偽が判定され， 真である限り，while のあとに続く文が実行され続ける． while

での繰り返しから抜け出すには break 文を使用する. 繰り返し途中で, 残りの処理を飛ばすには continue 文を使う.

関連項目: break , continue , for