C/C++言語プログラミング用語説明 - 基礎概念・実用技術・標準ライブラリ・記号・キーワード

C/C++言語の基本的な構文と概念を体系的に整理している． C/C++言語プログラミングの基礎知識として， 関数，変数，ポインタ，配列，構造体などの概念を，実践的な例を交えて説明している．さらに， ファイル操作，メモリ管理，入出力処理など，実用的なプログラミングに必要な技術も網羅している． それに加えて，エラー処理，デバッグ，コメント記述，数学関数，文字列処理，標準ライブラリなど， 実務でよく使用される機能，高品質なプログラム開発に不可欠な実践的内容を網羅している．

形式は，用語をあいうえお順に並べている． この文書は，C/C++言語プログラミングの包括的なリファレンスとして，学習と実践の両面に活用して欲しい．

ワードファイル版，PDF 版は，次のリンクでダウンロードできる．

- ワードファイル版
- PDF 版

【目次】

- 用語
- C プログラムの記号・キーワード等
- 目次

【サイト内の関連ページ】

- 種々のまとめページ: [人工知能，データサイエンス，データベース， 3 次元]，[Windows]，[Ubuntu]，[Python （Google Colaboratory を含む）]，[C/C++言語プログラミング用語説明]，[R システムの機能]，[Octave]

C/C++ の教材や説明など．

- C/C++言語プログラミング用語説明 - 基礎概念・実用技術・標準ライブラリ・記号・キーワード
  C/C++ の用語等の説明
- Windows で C プログラミング: [PDF]，[パワーポイント]
- 開発ツール，デバッグツールgcc，dbx，Makefile の使用法

- C プログラミング入門 （スライド資料とプログラム例）（全１５回）： リンク 》
- C/C++プログラミング講座：Visual Studioを用いた演習（基礎からアルゴリズムまで）（全14回）： リンク 》
- C++ オブジェクト指向プログラミング入門（授業資料）（全３回）： リンク 》
- C 言語によるアルゴリズムとデータ構造: リンク 》
- Visual Studio C++ 講座：基本操作から高度な機能まで学ぶ5回シリーズ

インストール

- Windows での Build Tools for Visual Studio 2022（ビルドツール for Visual Studio 2022）のインストール: 別ページ 》で説明
- Windows での Visual Studio Community 2022 のインストール: 別ページ 》で説明

用語

アクセス

データの読み出しや書き込みを行うこと．「メモリをアクセスする」，「ディスクをアクセスする」などのように使用する．

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」（コールバイバリュー）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．ただし，関数の引数と関数の仮引数とは本来別のものであるため，関数内で仮引数の値を変更しても，その影響が関数の引数に及ぶことはない．

関連項目：参照による呼び出し

アドレス

アドレスとは，メモリ内での位置を示す番地のこと．次のCプログラムは，データとそのアドレスを表示する例である．

```c
#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

同じ型のデータを複数個まとめて扱うためのデータ構造で，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（ネスト）構造になっている．

【プログラム説明】このプログラムは2次元配列xの全要素を0で初期化する．外側のループはiを0から9まで，内側のループはjを0から19まで繰り返す．

```c
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増やす操作のこと．C言語では「++」演算子を使用する．

関連項目：デクリメント

インデント

プログラムの可読性を高めるために行頭に空白を入れることで，字下げともいう．

エディタ

テキストやプログラムを編集する機能を持ったソフトウェア，あるいは編集を行うための画面のこと．

エラー

プログラムにおけるエラーには，次の3種類がある．

- 構文エラー（シンタックスエラー）
  プログラムの文法上の誤りにより，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
  定義域エラー（関数の入力が定義域外である場合．例えば，log関数の引数が負である場合など）や，範囲エラー（演算結果が許容範囲を超えて大きすぎたり，小さすぎたり，0に近すぎたりする場合）などがある．
- 意味的エラー
  プログラムは正常に動作するが，期待した結果が得られない論理的な誤り．

演算子

演算子（オペレータ）は，1個あるいは複数のオペランドを受け取り，特定の演算を行うための記号である．主な演算子は次の通りである．

論理演算子
- `!` 否定（NOT）
- `&&` 論理積（AND）
- `||` 論理和（OR）

比較演算子
- `!=` 等しくない
- `<` より小さい
- `<=` 以下（等しいか小さい）
- `==` 等しい
- `>` より大きい

>=　以上（等しいか大きい）
算術演算子
%　剰余（余り）
*　乗算
+　加算
++　インクリメント（1増加）
-　減算
--　デクリメント（1減少）
/　除算
その他
->　構造体へのポインタを使用したメンバアクセス
.　構造体のメンバ直接選択
=　代入
[]　配列要素の指定
など

オーバーフロー

演算結果の値が，その型で表現可能な範囲を超えて大きくなること（または小さくなること）．このとき，正しい演算結果が得られない．

オープンモード

ファイルを開く際の方式を指定するもので，読み込み専用，書き込み専用，追加書き込みなどがある．C言語のfopen関数で指定する．

関連項目：ファイル

オペランド

演算子によって処理される対象となるデータや式．

改行

現在の表示位置を次の行の先頭に移動する制御動作のこと．C言語では，文字列内に「¥n」を記述することで改行を表現する．

関連項目：改行文字

改行文字

改行を実行するための特殊文字．C言語では文字列内で「¥n」と記述する．

型

データや関数の性質と取りうる値の範囲を規定するもの．C言語における代表的な型は次の通りである．

- 配列型（array types）：同じ型の要素を複数個まとめて扱うための型
- 文字型（character types）：1文字を表現するための型
- 浮動小数点数型（floating types）：小数を含む実数を表現するための型
- 関数型（function types）：関数の入出力の型を規定するもの
- 整数型（integral types）：整数値を表現するための型
- ポインタ型（pointer types）：メモリアドレスを保持するための型
- 構造体型（structure types）：異なる型のデータをまとめて扱うための型

空文字

文字列の終端を示す特殊文字で，ヌル文字ともいう．C言語では「'¥0'」と表記する．すべての文字列の末尾には自動的にこの文字が付加される．

仮引数

関数定義内で宣言される変数のこと．例えば，C言語の関数宣言「int foo(int x);」において，xが仮引数である．関数が呼び出されると，引数の値が対応する仮引数にコピーされる．

引数（argument）は関数呼び出し時に実際に渡される値であり，仮引数（parameter）は関数定義時に宣言される変数である．

関数

プログラムにおける関数は，特定の処理をまとめた再利用可能なコードブロックである．引数を受け取り，処理を実行し，結果を返す．

[関数呼び出しでの制御の流れ]

通常，プログラムは上から順に実行されるが，関数呼び出しが発生すると，その関数の処理が完了した後に呼び出し元の位置に戻る．C言語での関数呼び出しは，関数名の後にカッコ「()」を付け，その中に引数を記述する．

[関数宣言]

関数を使用する前に，その存在をコンパイラに知らせる宣言が必要である．C言語での関数宣言では，戻り値の型，関数名，そして仮引数の型と名前をカッコ「()」内に記述する．

[関数定義]

関数定義は，関数の具体的な処理内容を記述したもの．次のC言語の関数例は，整数値を受け取り，その値に1を加えて返す．

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

C言語の関数定義は「戻り値の型，関数名，(仮引数リスト) { ブロック }」という形式で記述する．return文で関数の戻り値を指定する．

関数には値を返すものと返さないものがある．C言語で値を返さない関数を定義する場合は，戻り値の型としてvoidを指定する．この場合のreturn文は単に「return;」と記述し，関数の実行を終了して呼び出し元に制御を戻す．

```
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

ポインタを使用してメモリ上のデータにアクセスすること．ポインタが指すアドレスに格納されている値を読み書きする操作である．

偽

C言語では，式の評価結果が0の場合を偽（false）と定義している．条件分岐や繰り返し制御で使用される．

関連項目：真

擬似乱数

C言語のrand関数によって生成される数値列．完全な乱数ではなく，一定のアルゴリズムで生成される数列である．srand関数を使用して初期値（シード値）を設定しない場合，プログラムを実行するたびに同じ数列が生成される．

クラス

フィールド（メンバ変数）とメソッド（メンバ関数）が同一の形式で定義されているオブジェクトの集まり．オブジェクト指向プログラミングにおける基本的な構成要素である．

繰り返し

ある条件が満たされるまで，同じ処理を繰り返し実行すること．ループ（loop）ともいう．C言語では，繰り返しを実現するための制御文として 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「：」が付いた識別子であり，switch 文の中でのみ使用できる．複数の条件分岐を実現するために用いられる．

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．プログラミングは，プログラムの設計から作成までの全工程を指すのに対し，コーディングは，既に決定された設計に基づいて，実際のプログラムを記述する作業を指すことが多い．

構造体

複数の異なるデータ型のメンバをまとめて一つの新しいデータ型として扱う仕組み．各メンバには名前が付けられ，個別にアクセスできる．

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．プログラムの起動と終了，プログラムの出力のファイルへのリダイレクト，コマンド履歴（ヒストリ）の管理，ワイルドカードを用いたパターンマッチ，ファイル名の補完などの機能を提供する．

コメント

プログラムの中に記述する注釈のこと．プログラムの動作説明，使用上の注意点，変数の役割などを記録する．コメントは，プログラム実行時には無視される．C言語では，複数行コメントの始まりを「/*」，終わりを「*/」で示す．また，「//」を使用して1行分のコメントを記述することもできる．コメントは入れ子構造にすることはできない．次のコメントは入れ子になっているため，誤りである．

```
/* 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，コンパイラを使用して，人間が記述したプログラムのファイルを，コンピュータが直接実行可能な実行形式ファイルや，より効率的に解釈実行できる中間形式のファイルに変換すること．

再帰的呼び出し

関数が自分自身を呼び出すような関数呼び出しのこと．再帰的呼び出しの代表的な例として，ハノイの塔の問題がある．ハノイの塔とは，3本の柱（X，Y，Z）があり，最初，複数枚の円盤が柱Xに大きいものが下になるように積まれている状態から，「大きな円盤を小さな円盤の上に置かない」というルールを守りながら，円盤を1枚ずつ移動させて，すべての円盤を柱Xから別の柱に移動させる問題である．

次に，ハノイの塔を解くC プログラムを示す．

【プログラム説明】このプログラムは，ハノイの塔の問題を再帰的アルゴリズムで解く関数hanoiを実装している．mainでは4枚の円盤を柱"X"から"Z"に移動させる手順を表示する．

```
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, x, z );  /* N-1 枚の円盤を Y から Z へ */
}

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

三角関数

プログラミングで使用できる代表的な三角関数は次の通りである．

- acos ： 逆コサイン（アークコサイン）関数
- asin ： 逆サイン（アークサイン）関数
- atan ： 逆タンジェント（アークタンジェント）関数
- cos ： コサイン関数
- sin ： サイン関数

- tan ： タンジェント関数

算術

正の整数，小数，分数を対象とした基本的な数値計算のこと．加減乗除などの基本演算を含む．

参照による呼び出し

関数の呼び出しにおいて，ポインタを介してデータにアクセスする方式．関数の引数としてポインタを渡すことで，関数内部でポインタが指し示すメモリ上のデータを直接読み書きできる．これにより，関数の処理結果を呼び出し側に反映させることが可能となる．関数に渡されたポインタ（メモリアドレス）は関数の仮引数にセットされ，そのポインタを通じてデータの読み書きが行われる．

関連項目：値による呼び出し

式

プログラミングにおける式とは，演算子とオペランドの組み合わせのこと．以下に，C プログラムにおける式の例を示す．

```
30                          /* 定数 */
x                           /* 変数 */
x[100]                      /* 配列要素 */
a+b                         /* 加算 */
a*b                         /* 乗算 */
a++                         /* インクリメント */
a<100                       /* 比較 */
c=a+b                       /* 代入 */
sqrt( (x*x) + (y*y) )       /* 関数呼び出し */
```

関連項目：式文

式文

式のみで構成される文．式の後ろにセミコロンを付けることで，式を文として使用できる．

識別子

C言語において識別子とは，関数，case ラベル，構造体，変数等に付ける名前のこと．識別子には，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（_），半角の数字を使用できる．ただし，識別子の先頭文字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言語の予約語は識別子として使用できない．

自己参照構造体

構造体の定義内に，自身の型へのポインタを持つような構造体のこと．リスト構造やツリー構造などのデータ構造を実現する際に使用される．

字下げ

プログラムにおいて，行の先頭に一定のルールでタブや半角空白文字を挿入し，プログラムの階層構造を視覚的に表現すること．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おり，プログラムの構造が明確に示されている．

```
#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，コンピュータが直接実行可能な形式と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，実行可能ファイルとも呼ばれる．コンパイラによってソースコードから生成される．

ジャンプ文

プログラムの実行位置を別の場所へ移動するための制御文．C プログラムにおけるジャンプ文には，continue 文（ループの次の繰り返しへ移動），break 文（ループや switch文から抜け出す），return 文（関数から戻る）などがある．

初期化

変数などのオブジェクトに最初の値をセットすること．初期化されていない変数は不定な値を持つため，プログラムの正しい動作を保証できない．C プログラムにおける主な初期化方法は次の通りである．

- 整数データの初期化

  等号（=）に続いて，数値や式を記述する．

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

- 浮動小数点数データの初期化

  等号（=）に続いて，数値や式を記述する．

  ```
  double rate = 124.10;
  ```

- 構造体データの初期化

  中括弧（{}）内に，メンバの初期値を順番に記述する．

  ```
  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（″）で囲む．配列サイズを指定しない場合（[]），文字列の終端を示すヌル文字も含めた必要最小限の大きさの配列が確保される．

  ```
  char message[] = "Please input data";  /* 文字列長+1のサイズ */
  ```

  配列サイズを明示的に指定した場合（例：[32]），指定されたサイズの配列が確保され，その先頭部分に文字列とヌル文字が格納される．

  ```
  char message[32] = "Please input data";  /* 32バイトの配列を確保 */
  ```

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 関数などで使用される入出力のための変換方式を指定する文字列である．

真

値が0以外のとき，真（true）となる．

関連項目：偽

数字

数字は，0から9までの以下の10個である．

0 1 2 3 4 5 6 7 8 9

制御文

プログラムの実行順序を制御するための文．C言語では，制御文がない場合，文は上から順に逐次的に実行される．制御文は以下の３種類に分類される．

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

宣言文では，識別子の名前と型を指定する．以下の C プログラムは，age という名前の整数型変数の宣言例である．

```
  int age;
```

以下は，FILE 型へのポインタの宣言例である．

```
FILE *fp;
```

以下は，整数型配列の宣言例である．

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

以下は，3つの整数の最大値を返す関数の宣言例である．

```
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

条件式の値に基づいて，実行する文を選択するための制御文．C プログラムの選択文には，if 文，switch 文がある．

ソースファイル

プログラムのソースコードを保存したテキストファイルである．

ソースプログラム

プログラミング言語で記述されたプログラムの文字列，またはそれを保存したテキストファイルのこと．ソースコードともいい，プログラミング言語処理系によって実行可能な形式に変換される．

注釈

プログラム中に記述する説明文のことで，コメントともいう．

データ構造

データの集合をコンピュータで効率的に処理するための組織化方式．主なデータ構造として，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

人間が読める形式の文字のみで構成されたファイルである．

テスト

プログラムが仕様通りに動作するか検査し，プログラム中の不具合（バグ）を発見するための作業．

デバッガ

プログラム中の不具合（バグ）を発見し修正するための支援機能を提供するソフトウェアツール．

デバッグ

プログラム中の不具合（バグ）を発見し修正する作業.

デクリメント

変数の値を1減少させる操作.

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領域を必要に応じて確保・解放する機能. C++言語では new演算子と delete演算子を使用する.

NULL ポインタ

有効なメモリ位置を指していない特別なポインタ値.

ヌル文字

C言語の文字列において，文字列の終端を示す特殊な文字（値が0の文字）.

バイト

コンピュータにおけるデータの基本単位. 通常1バイトで基本文字（英数字，記号など）1文字を表現できる.

配列

同じデータ型の要素がメモリ上に連続して格納されているデータ構造. 例えば，Cプログラムの「int table[100];」では，整数型の要素100個が連続して配置され，各要素には0から99までの添字が割り当てられる.

```
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，要素数，および要素の型を指定する．例えば，C プログラムの「int table[100];」は，table という名前の，int型要素100個からなる配列を宣言している．

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

複数の[]を使用することで多次元配列を宣言できる．以下は2次元配列の例である．

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

関連項目： 配列

ハノイの塔

ハノイの塔の詳細については，「再帰的呼び出し」の項を参照のこと．

引数

関数呼び出し時に，関数に渡される具体的な値または式のこと．

関連項目：仮引数

評価

プログラム中の式から具体的な値を計算すること．

標準出力

プログラムの標準的な出力先であり，通常はディスプレイ画面に対応する．環境設定により，出力先をファイルなどに変更できる．C プログラムでは stdout として参照される．

標準入力

プログラムの標準的な入力元であり，通常はキーボードに対応する．環境設定により，入力元をファイルなどに変更できる．C プログラムでは stdin として参照される．

ファイル

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．ファイルの読み込みと書き出しを行う前と終了後には，オペレーティングシステムへの通知（オープン，クローズ）が必要である．

［ファイル位置指示子］

ファイル内で現在読み書きが行わ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ている．ファイルの読み込みや書き出しを行うと，ファイル位置指示子は移動していく．ファイル位置指示子を操作することで，ファイル内の読み書き位置を移動し，任意の位置での読み書きが可能となる．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．また，ファイル位置指示子の操作には，fseek関数，ftell関数等を使用する．

［ファイル名］

ファイルに付けられた名前のこと．C プログラムでは，fopen関数を使用してファイルをオープンする際に，ファイル名の指定が必要である．

［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必要な操作．C プログラムでは，ファイルのオープンにはfopen関数等を使用する．

［ファイルのクローズ］

ファイルの読み書きが終了した後に必要な操作．C プログラムでは，ファイルのクローズにはfclose関数等を使用する．

［ファイルの読み込み］

ファイルからプログラムへデータを読み込む操作のこと．

［ファイルの書き出し］

プログラムからファイルへデータを書き込む操作のこと．

［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に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 プログラムにおいて，FILE オブジェクトはファイル等を操作するためのすべての情報を保持している．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る．FILE オブジェクトへのポインタは，ファイルポインタ（file pointer）とも呼ばれる．

複文

ブロックの別名．

プログラム

コンピュータはプログラムによって動作する．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，様々な用途に応じて使用するアプリケーションプログラムがある．

プログラミング

プログラムを作成すること．

プログラミング言語

プログラムを記述するための言語．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，多様な種類が存在する．

プログラミング言語のシェル

Python言語のシェルとしてIPython，Ruby言語のシェルとしてirbが代表的である．

プログラミング言語処理系

プログラムを解釈実行するインタプリタと，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存在する．

プログラミングスタイル

C プログラムでは，if文の後に続く「｛」について，次のように記述できる．

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

あるいは次のような記述も可能である.

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

次のような記述では（文の字下げが不統一で），可読性が低下する．ブロック内では，式は一貫した字下げ幅を維持することが重要である.

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

また，次のように括弧の字下げが不統一だと，プログラムの構造が把握しづらくなる．括弧の対応関係が明確になるよう記述することが重要である.

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

プロセッサ

機械語（マシン語）のプログラムを解釈し実行する機能を持つ装置.

ブロック

C言語におけるブロックとは，「 {} 」で囲まれた1つ以上の文の集まり．ブロックは，関数の定義，if文，switch文，for文，while文などで使用される．以下のプログラムでは，main関数の本体が1つのブロックを形成している．また，ブロックは入れ子構造をとることができる．

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

文

C言語における文には，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

名前の付いたオブジェクトのこと．数学における変数の概念とは異なる．

［変数宣言］

変数の名前と型などを記述し，プログラム中での変数の使用を宣言すること．C言語における変数宣言は，末尾をセミコロンで終える．以下のCプログラムでは，変数xはint型（整数データ），変数rateはdouble型（浮動小数点数データ）として宣言されている．

```
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可能である．以下の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（「{}」で囲まれた文の集まり）内で行った場合，宣言された変数はそのブロック内でのみ有効である．

ポインタ

ポインタという用語は，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）を指し示すという機能に由来する．ポインタを使用することで，任意のデータの読み書きが可能となる．ただし，ポインタの内容が不正確な場合，データを破壊する可能性がある．ポインタの内容は，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「*」を使用する．「int *p;」のような記述で，整数データへのポインタが宣言される．pの内容は，整数データが格納されているメモリアドレスとなる．

```
int *p;
```

C プログラムでは，ポインタ自身も1種の変数として格納される（ポインタ変数という）ため，「int **table;」のような記述でポインタへのポインタを宣言することも可能である．

[ポインタ型]

「int へのポインタ」は，C プログラムでは次のように記述する．

```
int *p;
```

この場合，p は int 型へのポインタ型である．

メモリ

データやプログラムを記憶するための装置．メモリは一定の長さを持つメモリブロックが連続して配置された構造を持ち，各メモリブロックには固有のアドレスが割り当てられている．

メンバ

構造体の各要素に付けられた名前のこと．以下の 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際は，ピリオド「.」あるいは，->を使用する．

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

文字型

文字データを扱うための型．以下のプログラムでは，c は文字型である．

```
char c;
```

文字列データ

文字列データを扱うための最も基本的な方法は，文字の配列を使用すること．C プログラムでは，文字の配列に文字を1文字ずつ格納し，最後にヌル文字を付加する．このヌル文字は，文字列の終端を示す記号である．ただし，「文字列の長さ」を計算する際には，最後のヌル文字は含まない．

C プログラムにおける文字列操作の代表的なライブラリ関数を以下に示す.

- 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

C プログラムにおいてあらかじめ特別な意味が定められている単語. そのため, 予約語を変数名や関数名として使用することはできない.

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]

！は論理否定を行う演算子．オペランドが0の場合は真（0以外の値）を返し，0以外の場合は偽（0）を返す．すなわち，オペランドの真偽値を反転させる働きがある．

関連項目： !=

!=

[オペランド１]!=[オペランド２]

！= は不等価比較を行う演算子．オペランド１とオペランド２を比較し，両者が異なる値のときには真（1）を，等しい値のときには偽（0）を返す．

関連項目： !，<，<=，==，>，>=，

″

二重引用符「″」は，文字列リテラルの開始と終了を示す区切り記号．

#include

#include は，Cコンパイラ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指定されたファイルの内容をその位置に取り込むように指示するプリプロセッサ命令．

%

[オペランド１]%[オペランド２]

%は，整数の除算における剰余を求める演算子．オペランド１をオペランド２で割った際の余りを返す．浮動小数点数の剰余を求める場合は，fmod関数を使用する必要がある．

関連項目： /

&

&[オペランド]

&はアドレス演算子であり，オペランドが格納されているメモリ上の位置（アドレス）を返す演算子．

&&

[オペランド１]&&[オペランド２]

&&は論理積（AND）を求める演算子．オペランド１とオペランド２の両方が真（0以外の値）の場合に真を返し，それ以外の場合は偽（0）を返す．

関連項目： ||

()

[関数名]()，あるいは([式])

()は，関数呼び出しや式の優先順位を示す区切り記号．関数名の後に()を付けると引数のリストを指定できる．また，式を()で囲むと，その式を優先的に評価することを示す．

*

[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，以下の3つの用途で使用される演算子または記号：

- 乗算演算子：２つのオペランド間で使用され，それらの積を計算する．
- 間接参照演算子：ポインタが指すメモリ位置の内容を取得する．
- ポインタ宣言：変数宣言時に使用し，その変数がポインタ型であることを示す．

例： FILE* fp;

*/

＊／はコメントの終了を示す区切り記号．コメントはプログラムの説明や注意事項を記述するための注釈であり，プログラムの実行には影響を与えない．

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であり，オペランド１とオペランド２の和を計算して返す．

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．オペランドの値に1を加えて，その結果をオペランドに格納する．「[オペランド] = [オペランド] + 1;」と同等の処理を行う．

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であり，オペランド１からオペランド２を減じた結果を返す．

関連項目： +, --

--

[オペランド] --
-- [オペランド]

「--」はデクリメントを行う演算子．オペランドの値から1を減じて，その結果をオペランドに格納する．「[オペランド] = [オペランド] - 1;」と同等の処理を行う．

関連項目： ++

->

[ポインタ]->[メンバ名]

「->」は，構造体ポインタを通じてメンバを参照するための演算子．構造体へのポインタからそのメンバにアクセスする際に使用する．

例：

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

.

[名前].[メンバ名]

「.」は，構造体変数のメンバを直接参照するための演算子．構造体変数からそのメンバにアクセスする際に使用する．

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で割った商を返す．

関連項目： %

/*

「/*」はコメントの開始を示す区切り記号．「*/」までの範囲がコメントとして扱われる．

関連項目： */, //

//

「//」は行コメントを示すコメント区切り記号．「//」から行末までがコメントとして扱われる．

関連項目： /*, */

;

「;」は文の終端を示す区切り記号．プログラム内で文と文の区切りを明示する．

<

[オペランド１]<[オペランド２]

「<」は大小比較を行う演算子．オペランド１がオペランド２より小さい場合に真を返し，それ以外の場合は偽を返す．

関連項目： !=, <=, ==, >, >=

<=

[オペランド１] <= [オペランド２]

「<=」は大小比較を行う演算子．オペランド１がオペランド２以下の場合に真を返し，それ以外の場合は偽を返す．

関連項目： !=，<，==，>，>=

=

［オペランド１］=［オペランド２］

「=」は代入を行う演算子．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］

「==」は等価比較を行う演算子．オペランド１とオペランド２が等しい場合に限り真を返し，それ以外の場合は偽を返す．

関連項目： !=，<，<=，>，>=

>

［オペランド１］>［オペランド２］

「>」は大小比較を行う演算子．オペランド１がオペランド２より大きい場合に真を返し，それ以外の場合は偽を返す．

関連項目： !=，<，<=，==，>=

>=

［オペランド１］>=［オペランド２］

「>=」は大小比較を行う演算子．オペランド１がオペランド２以上の場合に真を返し，それ以外の場合は偽を返す．

関連項目： !=，<，<=，==，>

||

［オペランド１］||［オペランド２］

「||」は論理和（OR）を求める演算子．オペランド１とオペランド２の論理和を計算する．各オペランドは0を偽，0以外を真として扱い，いずれかのオペランドが真（0以外）の場合に真を返す．

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ヘッダファイル．指数関数，対数関数，双曲線関数，整数剰余関数，三角関数などが含まれている．

stdio.h

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

stdio.h は，入出力に関する定義と宣言が収められたヘッダファイル．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<stdlib.h>
```

stdlib.h は，一般操作に関する定義と宣言が収められたヘッダファイル．メモリブロックの確保と解放，擬似乱数の生成，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，文字列操作に関する定義と宣言が収められたヘッダファイル．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．

英字

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である．

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である．

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である．

関連項目： acos , asin , cos , sin , tan

break

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

関連項目： continue , return

case

```
case [式];
```

caseは，switch文とともに使用する制御文．switch文で評価された式の値は，各caseラベルの値と比較され，一致したcaseラベルに制御が移り，break文まで実行される．

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（8ビット）で表現され，256通りの文字をコード化できる．

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などの型名と併用する．関数がデータ値を変数で受け取る際，関数内でその変数の値を変更しない場合によく使用される．Cのライブラリ関数では，関数内で変数の値を変更しない場合，関数の引数宣言にconstを付加することが一般的である．

continue

continueは，次のループ実行を行うための文．continue文を含む最も内側のループ文（while文，for文など）について，ループ本体の残りを省略して次の繰り返しを開始する．

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

defaultは，switch文とともに使用する．switch文で評価された式の値が，どのcaseラベルの値とも一致しない場合，default部分に制御が移り，break文まで実行される．

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．精度は10桁で，10の-37乗から10の+37乗までの正負の範囲を扱える．コンピュータの機種により，より高い精度や広い範囲を扱える場合もある．

else

```
else [文];
```

elseは，if文と組み合わせて使用する制御文．次の例では，xが負の場合にメッセージ「x is negative」を表示し，xが0以上の場合にxの平方根を計算する．

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

if文の条件式が0の場合は偽，0以外の場合は真となる．条件式が0のとき，else以下の文が実行され，0以外のときは実行されない．

関連項目： if , switch

EOF

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

EOFは，ファイルの終端を示す定数．ファイル入力において，これ以上の入力がないことを示す場合や，入出力操作でエラーが発生したことを示す場合に使用する．

関連項目： stdio.h

exit()

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

exit関数は，プログラムを終了させるためのライブラリ関数．exit関数を実行すると，プログラムは直ちに終了する．

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．自然対数の底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キーワード．

fclose()

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

fclose関数は，ファイル等のクローズ（終了処理）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．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場合，または読み込み時にエラーが発生した場合には，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に格納する．基本的には1行分の文字を読み込み，最後に文字列の末尾を示すヌル文字（'¥0'）を付加する．読み込み時は改行文字までを含める．ただし，1行が長く読み込む文字がn-1文字に達した場合，または行の途中でファイルの終わりに達した場合には，読み込みを終了し，最後にヌル文字を付加する．ファイルからの読み込み時にエラーが発生した場合は，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構造体．現在の読み書き位置や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．このポインタは，その後のファイル読み書きやファイルのクローズの際に使用する．

fmod()

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

fmod関数は，浮動小数点数の剰余を計算するライブラリ関数．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，新規にファイルが作成される．ファイルがすでに存在する場合，ファイル中のデータはすべて削除される（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，処理の繰り返しを制御する制御文．for文では，例えば for(i = 0; i < 100; i++)のように，3つの式を記述する．最初の式（初期式）は，繰り返しの開始時に1回だけ実行される．2番目の式（条件式）は，繰り返しのたびに真偽が判定され，偽の場合は繰り返しが終了する．3番目の式（再設定式）は，繰り返しのたびに実行される．for文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文を使用する．

「for(初期化; 条件; 繰り返し) 文」は，次のwhile文と等価である．

```
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（第2引数のformat）に従って，ファイル等への出力を行うライブラリ関数．メッセージおよび整数，浮動小数点数，文字，文字列などのデータの出力が可能である．書式文字列には，データ出力のために%付きの指定子を記述する（詳細はprintfの項を参照）．「%」文字自体を出力する場合は，書式文字列中で「%%」と記述する．

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文字を書き込むライブラリ関数．ファイルへの書き出し時にエラーが発生した場合は，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ライブラリ関数．文字列末尾のヌル文字は出力されない．ファイルへの書き出し時にエラーが発生した場合は，EOFが返される．

改行文字の扱いは次の通りである．

- gets：改行文字までを読み込むが，改行文字は破棄する．
- fgets：改行文字までを読み込み，改行文字を含めて出力する．
- puts：文字列末尾に改行文字を自動的に付加して書き込む．
- fputs：文字列末尾に改行文字を付加しない．

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を解放する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．第2引数のoffsetには，移動量をバイト単位で指定する．前方に進める場合は正の数を，後方に戻す場合は負の数を指定する．第3引数のwhereには，移動の基準位置を次の3種類から指定する．

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取得するためのライブラリ関数．実行すると，通常，ファイル先頭からの文字数（位置）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用する制御構文．次の例では，変数 x が負の値のときにのみ，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 の場合は偽，0 以外の場合は真となり，if 文の条件が真のときは，if 以下の文が実行される．if 文の条件が偽のときは，if 以下の文は実行されない．

if 文の後に else を記述し，else の後にさらに文を記述できる．このような場合，if 文の条件が 0 以外のときは if 以下の文のみが実行され，if 文の条件が 0 のときは else 以下の文のみが実行される．

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 までの範囲の数値のことで，数学での整数とは異なる．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．対数関数は 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．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 argc, char *argv[]」（整数データ argc と，文字列の配列 argv）である．main 関数の戻り値は，「int main(...」とあるように int 型である．main 関数の最後の「return 0;」で，戻り値として 0 を返している．

```
#include<stdio.h>

int main( int argc, 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，ヌルポインタを表す定数（ヌルポインタとは，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である）．NULL は，「(void *)0」と同等である．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（第 1 引数の format）に基づき，標準出力への出力を行う（標準出力は端末に結びついており，Microsoft Windows では画面に表示されるウインドウのことである）．printf 関数を使用して，メッセージおよび整数，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 % 付きの文字を記述する．printf 関数を使用して「%」を出力したい場合には，書式文字列中で「%%」と記述する．

```
%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
%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 のように数字を記述することができ，表示のための幅を指定する．このように表示幅を指定した場合，データ（数値や文字）が指定した幅より大きいときは，表示が崩れる（ただし，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記述することができ，表示のための幅（この場合は 10）と，小数点以下の表示桁数（この場合は 4）を指定する．

関連項目： fprintf , sscanf

rand()

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

rand 関数は擬似乱数 (pseudo-random number) を生成するためのライブラリ関数. 生成される数値は, 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場合, 同じ系列の擬似乱数を返す.

関連項目： RAND_MAX , srand

RAND_MAX

```
#include<stdlib.h>
```

RAND_MAX は, rand 関数で生成される擬似乱数 (pseudo-random number) の最大値を表す定数である.

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用する制御文. return を実行すると, return を含む関数の実行を直ちに終了し, 呼び出し側の関数に戻る. return の後に式を記述することができ (そのためには関数の宣言が int foo(…)や double foo2(…) のように, 型を指定して行われている必要がある) , 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示す定数. fseek などで使用する.

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示す定数．fseek などで使用する．

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示す定数．fseek などで使用する．

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（正弦）を計算するためのライブラリ関数である．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ための型である．

sqrt ()

```
#include<math.h>
double sqrt(double x);
```

sqrt 関数は，平方根を計算するためのライブラリ関数である．

srand()

```
#include<stdlib.h>
void srand(unsigned int seed);
```

srand 関数は，rand 関数で生成する擬似乱数（pseudo-random number）の系列を設定するためのライブラリ関数である．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に従ってデータを読み込む関数である．sscanf 関数を使用して，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 % 付きの文字を記述する．format の後には，読み取ったデータを格納する変数の並びを記述するが，必要に応じて，変数の前に & を付ける．

```
%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標準入力のことで，Microsoft Windows ではキーボードを指す．

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である．strcat 関数を実行すると，引数の string2（string2 の末尾のヌル文字までを含めた全ての文字）が，string1 の末尾に連結されてコピーされる．コピーの前にあった string1 の末尾のヌル文字は，string2 により上書きされる．

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である．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 が string2 より辞書順で前の場合： strcmp 関数は負の値を返す
- string1 と string2 が一致する場合： strcmp 関数は 0 を返す
- string1 が string2 より辞書順で後の場合： 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ライブラリ関数である．この関数を実行すると，引数string2の文字列（末尾のヌル文字を含むすべての文字）が，string1にコピーされる．コピー先のstring1の内容は上書きされるため，コピー前の内容は失われる．

strlen()

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

strlen関数は，文字列の長さを取得するためのライブラリ関数である．この関数を実行すると，引数stringの文字数が返される．ただし，末尾のヌル文字は文字数に含まれない．

struct

structは，構造体を定義するためのキーワードである．構造体とは，1つ以上のメンバを含むデータ型のことであり，各メンバは異なる名前と型を持つことができる．

switch

switch ([式]) [文]

switch文は，条件分岐を実現するための制御構文である．switch文で指定された式の値が，各caseラベルの値と順次比較される．一致したcaseラベルに制御が移り，対応する処理がbreak文まで実行される．

```
#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（正接）を計算するライブラリ関数である．引数はラジアン単位で指定する．

関連項目： acos , asin , atan , sin , cos

void

voidは，関数の戻り値の型，引数の型，またはポインタの型を指定するために使用されるキーワードである．

- 関数の型
  関数が値を返さないことを示すためにvoidを使用する．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を使用する．例えば，
  int foo2(void);
  なお，引数がない場合のvoidは省略可能であり，int foo2();と記述することもできる．
- void*
  void*は，任意の型を指すことができる汎用ポインタである．

関連項目： NULL

while

while(条件) 文

whileは，処理を繰り返し実行するための反復制御構文である．while文では，条件式の評価が真である限り，続く文が繰り返し実行される．繰り返しを途中で終了するにはbreak文を，現在の繰り返しをスキップして次の繰り返しに進むにはcontinue文を使用する．

関連項目： break ， continue ， for

目次

あ -- ん

- 値による呼び出し
- アドレス
- アレイ
- 入れ子
- インクリメント
- インデント
- エディタ
- エラー
- 演算子
- オーバーフロー
- オープンモード

- オペランド
- 改行
- 改行文字
- 型
- 空文字
- 仮引数
- 関数
- 間接参照
- 偽
- 擬似乱数
- クラス
- 繰り返し
- case ラベル
- コーディング
- 構造体
- コマンドラインシェル
- コメント
- コンパイル

- 再帰的呼び出し
- 三角関数
- 算術
- 参照による呼び出し
- 式
- 式文
- 識別子
- 自己参照構造体
- 字下げ
- 実行形式ファイル
- ジャンプ文
- 初期化
- 書式文字列
- 真
- 数字
- 制御文
- 宣言
- 選択文

- ソースファイル
- ソースプログラム
- 注釈
- データ構造
- テキストファイル
- デクリメント
- テスト
- デバッガ
- デバッグ
- 動的メモリ管理
- NULL ポインタ
- ヌル文字
- バイト
- 配列
- 配列の宣言
- ハノイの塔
- 引数
- 評価

- 標準出力
- 標準入力
- ファイル
- FILE オブジェクト
- 複文
- プログラム
- プログラミング
- プログラミング言語
- プログラミング言語のシェル
- プログラミング言語処理系
- プログラミングスタイル
- プロセッサ
- ブロック
- 文
- 変数
- ポインタ
- メモリ
- メンバ

- 文字型
- 文字列データ
- 予約語
- ループ

a -- z

- abs
- acos
- asin
- atan
- break
- case
- char
- const
- continue
- cos
- default
- double
- else
- EOF
- exit
- exp
- extern
- fclose
- fgetc
- fgets
- FILE
- fmod
- fopen
- for
- fprintf
- fputc

- fputs
- free
- fseek
- ftell
- if
- int
- log
- main
- malloc
- NULL
- printf
- rand
- RAND_MAX
- return
- SEEK_CUR
- SEEK_END
- SEEK_SET
- sin
- size_t
- sqrt
- srand
- sscanf
- stdin
- strcat
- strcmp
- strcpy
- strlen
- struct
- switch
- tan
- void
- while