

C 言語のサンプルプログラム (2)

計算機応用 5E 2004.04.15

1. 数学関数 -----	5
1.1 三角関数	
2. 演算子 (教科書 8 章) -----	7
2.1 算術演算子	
2.2 関係演算子	
2.3 論理演算子	
2.4 加算減算演算子	
2.5 代入演算子	
2.6 その他の演算子	
3. 制御文 (教科書 9 章) -----	11
3.1 if 文	
3.2 for 文	
3.3 while 文	
3.4 do while 文	
3.5 switch 文	
3.6 goto 文	

1. 数学関数

ここでは、数学関数を示します。プログラムの流れと、C 言語ではいろいろな数学の関数が使えることを理解してください。数学関数については、教科書の P.351 に簡単に書いてあります。

1.1 三角関数 (sin.c)

入力値の sin を計算するプログラムです。入力が -9999 以下だと、計算を打ち切ります。

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

int main(){
    double x, y;
    char c;

    while(1){
        printf("x=");
        scanf("%lf%c", &x, &c);

        if(x < -9998.9999) break;

        y = sin(x);
        printf("sin(%10.5f)=%10.5f\n", x, y);
    }

    return 0;
}
```

説明

- `stdio.h` は標準ライブラリー関数のヘッダーファイルです。標準ライブラリー関数とは、キーボードやディスプレイとの入出力に使われる関数です。(教科書 P.314)
- `math.h` は数学関数のヘッダーファイルです。ここでは、`sin()` を使うので必要です。数学関数を使う場合、このヘッダーファイルが必要です。
- ヘッダーファイルには、関数の定義やマクロ、プロトタイプなどが書かれています。
- `#include` 文(P.284～)により指定されたファイルがそこに、挿入されます。詳細は気にしないで、C 言語の関数を使うときには、その関数の仕様が書かれたヘッダーファイルを `#include` により、プログラムの先頭に書く必要があることを理解すればよいです。
- このプログラムは、`main` 関数しかありません。C 言語の `main` 関数とは FORTRAN のメインルーチンと同じです。FORTRAN のサブルーチンにあたるものは `main` 以外の名前を付けます。(P.197～)
- C 言語では `main` 関数が必ず 1 個必要で、そこから実行されます。

- main 関数を実行したときの戻り値の型は、int (整数) と宣言しています。戻り値は return 文 (P.153) で指定しています。ここでは、常に 0 です。
- 倍精度実数 (double) と文字 (char) の変数の宣言です。 (p.32)
- while () { } は繰り返し文 (P.141)。 () 内が真である限り、 { } を繰り返します。 break 文または偽、即ち (0) になれば、繰り返しから抜けます。 { } 内を実行前に、 () の真偽を確認します。 (1) と書けば、無限ループ。常套手段です。
- C 言語では、0 のときのみ偽 (false) です。それ以外は、すべて真 (true) となります。 (P.106)
- scanf で数値を読み込んでいる。数値は double と改行文字は char で読み込みます。scanf の引数は、ポインタ (アドレス) なので & をつけて、ポインタを渡しています。%lf は long float で倍精度実数 (double) を読み込むときの書式です。%c は character で 1 文字、\n を読み込むための書式です。
- C ではアドレスのことをポインタと言います (P.155)。厳密に言えば、少し異なりますが、今のところ、そう考えてください。変数 x が格納されているアドレスは、&x で取り出すことができます。これについては、後で詳しく説明します。
- printf 文 (P.320) で、ディスプレイに結果を表示します。%10.5f は、浮動小数点 (float) で小数点以下 5 桁、10 カラムで出力と言う意味。
- \n は改行 (new line) です。これはエスケープシーケンス (P.28) と呼ばれるものです。

コンパイル `cc -lm -o sin sin.c`

-lm は、数学関数のライブラリー /lib/libm.a を呼び出し、リンクするためにつけます。l はライブラリー (library) の l, m は libm.a の m です。math.h があるときは、必ずこのオプションが必要です。

実行結果 入力 x に対して、sin(x) の値を返す。
もし、-9999 以下の値を入力すれば、終了

練習問題 • 新たなディレクトリーを作成して、他の関数で同様なプログラムを作成せよ。全く新しくプログラムを書くのではなく、

`cp 元のファイル名 コピー先のファイル名`

または、ファイルマネージャーを使って、作成したファイルをコピーして、それを書き直すこと。

表1 数学関数一覧(P.351,P.410～)

名前	書式	戻り値	ヘッダーファイル	備考
abs	int abs(int n)	絶対値 $ n $	stdlib.h	
acos	double acos(double x)	arccos(x)	math.h	
asin	double asin(double x)	arcsin(x)	math.h	
atan	double atan(double x)	arctan(x)	math.h	$-\pi/2 < \text{戻り値} < \pi/2$
atan2	double atan(double y, double x)	arctan(y/x)	math.h	$-\pi < \text{戻り値} < \pi$
ceil	double ceil(double x)	小数点切り上げ	math.h	
cos	double cos(double x)	cos(x)	math.h	
cosh	double cosh(double x)	cosh(x)	math.h	
div	div_t div(int a, int b)	a/b の商と余り	stdlib.h	1.2 節を参照
exp	double exp(double x)	e^x	math.h	
fabs	double fabs(double x)	絶対値	math.h	
floor	double floor(double x)	小数点切り下げ	math.h	
fmod	double fmod(double x, double y)	x/y の余り	math.h	
frexp	double frexp(double value, int *p)	$x \cdot 2^p$ に分解	math.h	1.3 節を参照
labs	long labs(long n)	絶対値 $ n $	stdlib.h	
ldexp	double ldexp(double x, int n)	$x \times 2^n$	math.h	
ldiv	ldiv_t ldiv(long a, long b)	a/b の商と余り	stdlib.h	
log	double log(double x)	$\log_e(x)$	math.h	
log10	double log10(double x)	$\log_{10}(x)$	math.h	
modf	double modf(double x, double *ip)	整数と小数に分解	math.h	1.4 節を参照
pow	double pow(double x, double y)	x^y	math.h	
rand	int rand(void)	乱数の発生	stdlib.h	0~RAND_MAX (stdlib.h で定義)
sin	double sin(double x)	sin(x)	math.h	
sinh	double sinh(double x)	sinh(x)	math.h	
sqrt	double sqrt(double x)	x の平方根	math.h	
tan	double tan(double x)	tan(x)	math.h	
tanh	double tanh(double x)	tanh(x)	math.h	

2. 演算子

C言語には、いろいろな演算子があります。これらを上手に使うと、すっきりとした分かりやすいプログラムが書けます。演算子を上手に使う練習をしましょう。詳細については、教科書の8章(P.106～)に書かれています。

2.1 算術演算子 (math.c) (P.107)

おなじみの2項演算子と単項演算子です。説明は不要でしょう。

```
#include <stdio.h>

int main(){
    int a, b, c;

    a = 100;
    b = 3;

    c = a%b;

    printf("%d/%d no amari wa %d\n", a, b, c);

    return 0;
}
```

説明 • printf 文中の%dは、10進数(decimal)出力ということです。(P.322)

コンパイル cc -o math math.c

練習問題 • 他の演算子を使って、実行させてみよう。
 • 整数の変数を1ケ追加して、加算と減算を行い。それぞれの結果を表示させてみよう。

演算子 算術演算子を、以下の表に示します(P.107)。

演算子	機能	例
+	(単項プラス)	+a
-	符号反転 (単項マイナス)	-a
*	乗算	c = a*b
/	除算	c = a/b
%	剰余(余り)	c = a%c
+	加算	c = a+c
-	減算	c = a-b

2.2 関係演算子 (relat.c) (P.108)

これも、おなじみの関係演算子です。説明は不要でしょう。

```
#include <stdio.h>

int main(){
    double x, y;
    int a, b;

    x = 1.0;
    y = 2.0;

    a = x < y;
    b = x > y;

    printf("x < y no enzan kekka wa %d desu\n", a);
    printf("x > y no enzan kekka wa %d desu\n", b);

    return 0;
}
```

演算子

- 関係演算子は、それを挟んでいる 2 個の数値の大小関係や、等値関係を判断します。判断の結果は、

正しいとき	: 1
間違っているとき	: 0

を返します。

- 関係演算子を、以下の表に示します(P.108)。
- 演算子の左右の値が等しいか否かを調べるときは、==を使います。=ではありません。

演算子	機能	例
<	大小関係 (等号含まず)	if (a<b)
<=	大小関係 (等号含む)	if (a<=b)
>	大小関係 (等号含まず)	if (a>b)
>=	大小関係 (等号含む)	if (a>=b)
==	等しい	if (a==b)
!=	等しくない	if (a!=b)

2.3 論理演算子 (logic.c) (P.109)

論理演算子です。c の場合、偽は 0 で、真は非 0 です。したがって、非 0 の整数の否定は 0 になります。

```
#include <stdio.h>

int main(){
    double x, y, z;
    int a, b, c, d, e;

    x = 1;
    y = 0;
    z = -5;

    a = x < y && x > y;
    b = x < y || x > y;
    c = !x;
    d = !y;
    e = !z;

    printf("a = %d\n", a);
    printf("b = %d\n", b);
    printf("c = %d\n", c);
    printf("d = %d\n", d);
    printf("e = %d\n", e);

    return 0;
}
```

- 演算子
- 論理演算子は、ブール代数演算を行います。演算の結果、偽の場合は 0、真の場合は 1 を返します。
 - c の場合、偽は 0 で、真は非 0 です。したがって、非 0 の整数の否定は 0 になります。演
 - 論理演算子を、以下の表に示します。

演算子	機能	例
!	否定	if(!a)
&&	論理積 and	if(a && b)
	論理和 or	if(a b)

- 練習問題
- 作成したプログラムの結果が、なぜそのようになるか、考えよう。

2.4 インクリメント・デクリメント演算子 (**incdec.c**) (P.110)

数値を1加算、または1減算します。1加算をインクリメント、1減算をデクリメントと言います。

```
#include <stdio.h>

int main(){
    int a;

    a = 3;

    printf("a = %d\n", a);
    a++;
    printf("a = %d\n", a);
    a++;
    printf("a = %d\n", a);
    a++;
    printf("a = %d\n", a);

    a--;
    printf("a = %d\n", a);
    a--;
    printf("a = %d\n", a);
    a--;
    printf("a = %d\n", a);
    a--;
    printf("a = %d\n", a);

    return 0;
}
```

演算子

- ++演算子で1加算、--で1減算です。表中の FORTRAN と同じ記述は可能ですが、加算や減算演算子を使うほうが、C 言語風で良いです。
- ++a のは前置型、a++は後置型で、演算の順序が異なります。
(例1) a=1; x=++a; (例2) a=1; x=a++
例1の場合、①a=1 ②a=a+1 ③x=a の順序で実行され、x=2 となります。一方、例2の場合は、①a=1 ②x=a ③a=a+1 の順序で実行され、x=1 となります。気をつけてください。

演算子	書式	FORTRAN では(CでもOK)
++	a++ または ++a	a = a+1
--	a-- または --a	a = a-1

練習問題

- a++ が a=a+1、a-- が a=a-1 と同じであることを確認しよう。

2.5 代入演算子 (subst.c) (P.118)

演算の結果を代入します。

```
#include <stdio.h>

int main(){
    double x1, x2, x3, x4, y;

    x1 = 2.5;
    x2 = 2.5;
    x3 = 2.5;
    x4 = 2.5;

    y = 0.1;

    x1 += y;
    x2 -= y;
    x3 *= y;
    x4 /= y;

    printf(" x1 += y no kekka x1 = %f\n",x1);
    printf(" x2 -= y no kekka x2 = %f\n",x2);
    printf(" x3 *= y no kekka x2 = %f\n",x3);
    printf(" x4 /= y no kekka x2 = %f\n",x4);

    return 0;
}
```

コンパイル `cc -o subst subst.c`

演算子

- 代入演算子には、下表(P.119)のものがあります。表中の一般表記も使用できますが、代入演算子を使うほうが、c風でよろしいです。

代入演算子	一般表記(CでもOK)
a = b	
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a %= b	a = a % b
a &= b	a = a & b
a ^= b	a = a ^ b
a = b	a = a b
a <<= b	a = a << b
a >>= b	a = a >> b

2.6 その他の演算子

2.6.1 ビット演算子 (P.112)

ビット単位で演算を行います。

シフト演算子では、はみ出したビットは捨てられ、入ってくるビットは0が入れられます。また演算子の右の数字は、シフトの数を示します。

演算子	機能	使用例
&	ビットごとの AND	a = b & 0x7FFF
	ビットごとの OR	a = b 0x7FFF
^	ビットごとの XOR	a = b ^ 0x7FFF
~	ビットの反転 (1 の補数)	a = ~b
<<	左シフト	a = b << 2
>>	右シフト	a = b >> 2

2.6.1 ポインター演算子 (P.129)

ポインターと変数との間でデータを受け渡しするときに使います。

演算子	機能
*	ポインターが指しているアドレスの内容を取り出す
&	変数が格納されているアドレスを取り出す

3. 分岐と繰り返し(制御文)

3.1 if 文 (ifcont.c) (P.138)

数値を入力すると、それに応じた反応をします。-9999 以下の数字を入力すると、プログラムは停止します。

```
#include <stdio.h>

int main()
{
    double a;
    char c;

    printf("type number ? ");
    scanf("%lf%c", &a, &c);

    if(a < -9998.999){
        printf("bye bye !! \n");
        exit();
    }

    if(a < 0){
        printf(" a wa zero ika desu\n");
    }else{
        printf(" a wa zero ijou desu\n");
    };

    if(a < -5){
        printf("a < -5\n");
    }else if(-5 <= a && a < 0){
        printf("-5 <= a < 0\n");
    }else if(0 <= a && a < 5){
        printf("0 <= a < 5\n");
    }else{
        printf("5 <= a\n");
    }

    return 0;
}
```

- 説明
- if 文の真偽は、()の中が 0 の場合が偽で、非 0 の場合が真である。真の場合、続く{}を実行する。{}内の文はセミコロン;で区切ることにより、複文も可能である。
 - exit() は処理を打ち切る関数である。(P.373)
 - else if 文がある場合は、最初に真となったところで続く{}内を実行して、そのブロックからぬける。

コンパイル `cc -o ifcont ifcont.c`

3.2 for 文 (forcont.c) (P.142)

for 文で 100 回ループをまわして、1~100 までの和を求めるプログラムです。

```
#include <stdio.h>

int main()
{
    int a, b;

    a = b = 0;

    for(a=1; a<=100; a++){
        b += a;
    }

    printf("b = %d\n",b);

    return 0;
}
```

説明 • 指定の回数ループを回す場合、よく使われる for 文の文法は、

```
for( 初期設定式 ; 継続条件式 ; 再設定式) {
    文;
    文;
}
```

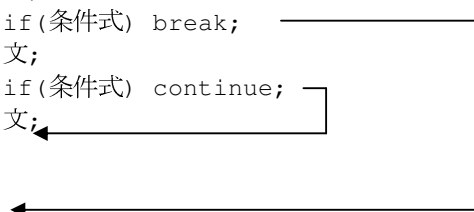
です。

• break 文や continue 文を使って、処理を打ち切ることも可能です。
break はループから抜けますが、continue は再設定式を実行して次のループに進みます。

```
文;
文;

for( 初期設定式 ; 継続条件式 ; 再設定式) {
    文;
    文;
    if(条件式) break;
    文;
    if(条件式) continue;
    文;
}

文;
文;
```



3.3 while 文 (whilecont.c) (P.141)

while 文で 100 回ループをまわして、1~100 までの和を求めるプログラムです。

```
#include <stdio.h>

int main()
{
    int a, b;

    a = 1;
    b = 0;

    while(a<=100){
        b += a;
        a++;
    }

    printf("b = %d\n",b);

    return 0;
}
```

説明

- while 文は、不定回数のループを回す場合によく使われます。継続条件がループの入口で判断されるため、最初からそれが偽の場合、ループ内は一度も実行されません。継続条件が真(非 0)の間、{}の中の文が繰り返されます。文法は、以下の通りです。

```
while(継続条件式){
    文;
    文;
}
```

- if 文同様に break 文や continue 文が使えます。

3.4 do while 文 (dowhilecont.c) (P.145)

do while 文で 100 回ループをまわして、1~100 までの和を求めるプログラムです。

```
#include <stdio.h>

int main()
{
    int a, b;

    a = 1;
    b = 0;

    do{
        b += a;
        a++;
    }while(a<=100);

    printf("b = %d\n",b);

    return 0;
}
```

説明

- do while 文も不定回数のループを回す場合によく使われます。ただし、while 文とは異なり、継続条件がループの出口で判断されるため、最低 1 回はループが回ります。継続条件が真 (非 0) の間、{} の中の文が繰り返されます。文法は以下の通りです。

```
do{
    文;
    文;
}while(継続条件式)
```

- if 文同様に break 文や continue 文が使えます。

3.6 goto 文 (goto.c) (P.152)

goto 文で 100 回ループをまわして、1~100 までの和を求めるプログラムです。

```
#include <stdio.h>

int main()
{
    int a, b;

    a = b = 0;

    start: a++;
    b += a;

    if(a >= 100){
        goto finish;
    }else{
        goto start;
    }

    finish: printf("b = %d\n",b);

    return 0;
}
```

説明

- goto 文で指定のラベルの文に飛びます。文法は、以下の通りです。

```
goto ラベル;
    文;
    文;
```

```
ラベル: 文;
```

- ラベルの後は文が続く必要があります。文との区切りは、コロンです。
- 実行文が無いときには、空文に飛ぶようにします。空文は、次のように書きます。

```
ラベル: ;
```

- goto 文を使うとプログラムの流れが分かりにくくなります。極力、goto 文は使わないようにしましょう。