C 言語のサンプルプログラム (2)

計算機応用　5E 2003.04.24

1. 数学関数

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>

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);
      }
}
```

説明

- math.h は数学関数のヘッダーファイルです。sin() を使うので必要です。ヘッダーファイルには、関数の定義やマクロなどがかかれています。
- while(){}は繰り返し分。() 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。
- 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、ポインター(アドレス)なので&をつけて、ポインターを渡しています。%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ーと言う。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。

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

表 1 数学関数一覧

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)	arcos(x)	math.h	
asin	double asin(double x)	arcsin(x)	math.h	
atan	double atan(double x)	arctan(x)	math.h	$-\pi/2$ < 戻り値 < $\pi/2$
atan2	double atan(double y, double x)	arctan(y/x)	math.h	$-\pi$ < 戻り値 < π
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*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)	整数と少数に分解	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	

1.2 商と剰余を求める関数 (div.c)

入力値の 20/3 の商と余りを計算するプログラムです。

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

main(){
   div_t c;
   int a, b;
   int shou, amari;

   a = 20;
   b = 3;

   c = div(a, b);

   shou = c.quot;
   amari = c.rem;

   printf("%d/%d = %d amari %d\n", a, b, shou, amari);

}
```

説明

- div_t は構造体で、メンバーとして quot と rem がある。関数 div(a, b) は、a/b の商(quotient)を quot に、余り(remainder)を rem に入れる。構造体は、数学のベクトルに似ており、複数のデータを格納できる。

コンパイル　cc -o div div.c

1.3 指数分解関数 (frexp.c)

-2〜2 の間を 0.1 刻みで、$\alpha \times 2^{\beta}$の形に分解する。

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

main()
{
   double value;
   int p;
   double a;

   for(value = -2; value <= 2 ; value += 0.1){
      a = frexp(value, &p);
      printf("%5.2f = %7.4f*2^%d\n",value, a, p);
   }
}
```

説明

- for(){}は繰り返し文である。これは、初期値 value=2 として、value を 0.1 刻みで増加させて、value <= 2 の間、{ }を繰り返す。
- value += 0.1 を FORTRAN 流に書くと、value = value + 0.1 である。もちろん C でも、この FORTRAN 流の書き方は許されるが、サンプルのように書くのが C らしくて良い。
- frexp()の第一引数が分解する数値、第二引数が分解した結果の指数部、この関数の戻り値が仮数部である。第二引数は、数値を戻すためにポインター(アドレス)を渡している。

コンパイル　cc -lm -o frexp frexp.c

1.4 小数分離関数 (modf.c)

-2〜2 の間を 0.1 刻みで、整数部 + 小数部の形に分解する。

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

main()
{
   double x, ip, a;

   for(x = -2; x <= 2 ; x += 0.1){
      a = modf(x, &ip);
      printf("%5.2f = %2.0f + %5.2f\n", x, ip, a);
   }
}
```

説明

- modf()関数で、浮動小数値を整数部と小数部に分解する。この関数の戻り値が、小数部である。整数部は、第二引数で示されたポインターに格納される。

コンパイル　cc -lm -o modf modf.c

2. 演算子

2.1 算術演算子 (math.c)

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>

main(){
   int a, b, c;

   a = 100;
   b = 3;

   c = a%b;

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

演算子　　算術演算子を、以下の表に示します。

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)

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>

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);
}
```

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。
- 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)

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>

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);
}
```

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)

2.4 加算減算演算子 **(incdec.c)**

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>

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);

}
```

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;　　(例 2) a=1;
  　　x=++a;　　　　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
++	a++　または　++a	a = a+1
--	a--　または　--a	a = a-1

2.5 代入演算子 (subst.c)

演算の結果を代入します。

```
#include <stdio.h>

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);

}
```

コンパイル　　cc -o subst subst.c

演算子　　・代入演算子には、下表のものがあります。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
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

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

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)

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>

main()
{
   double a;
   char c;

   printf("type any 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");
   }
}
```

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。
- 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める。

コンパイル　cc -o ifcont ifcont.c

3.2 for 文 (forcont.c)

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>

main()
{
   int a, b;

   a = b = 0;

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

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

}
```

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;
}
```

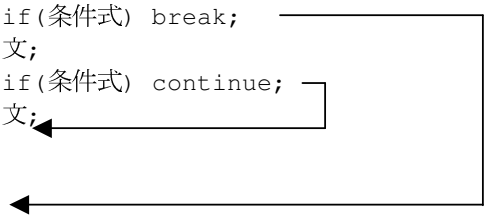
です。

• 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)

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>

main()
{
   int a, b;

   a = 1;
   b = 0;

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

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

}
```

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)

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>

main()
{
   int a, b;

   a = 1;
   b = 0;

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

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

}
```

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。

3.5 switch 文 (switchcont.c)

switch 文で 1～30 間での和を、途中経過を含めて出力します。

```
#include <stdio.h>

main()
{
   int a, b;

   a = b = 0;

   for(a=0; a<=30; a++){
       b += a;

       switch(a){
           case 0:
               printf("--------  0 --------\n");
               printf("   %d   %d\n", a, b);
               break;
           case 10:
               printf("-------- 10 --------\n");
               printf("   %d   %d\n", a, b);
               break;
           case 20:
               printf("-------- 20 --------\n");
               printf("   %d   %d\n", a, b);
               break;
           case 30:
               printf("-------- 30 --------\n");
               printf("   %d   %d\n", a, b);
               break;
           default:
               printf("   %d   %d\n", a, b);
       }
   }

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

}
```

説明

- switch case default の文法は、以下の通りです。式と定数式は、整数である必要があります。

```
switch(式){
   case 定数式1:
      文;
      文;
      break;
```

```
    case 定数式2:
        文;
        文;
        break;
    case 定数式3:
        文;
        文;
        break;
    default:
        文;
        文;
}
```

- break 文は省略可能ですが、マッチしたケース以外に default が実行されます。そのため、省略はしないほうが良いです。

- case 定数式:はラベルです。文の先頭に書く必要があります。そして、セミコロンではなく、コロンです。

3.6 goto 文 (goto.c)

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>

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);

}
```

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。