基礎からのプログラミング

福山平成大学　経営情報学科

福井正康

序

近年，C の地位は不動のものとなり，パソコンレベルにおいてもプログラム開発にかなり利用されています。また情報処理試験でも採用され，日本でも完全に標準言語になった感があります。さらに，C のスーパーセットともいうべき C++ も普及し、書店の言語関係のコーナーには Windows 用の Visual Basic の書籍と共に、C 及び C++ 関係の書籍が大半を占めるようになりました。このような状況でもプログラム言語を教育する側の人間には贅沢な悩みが１つあります。それは参考書が整備され過ぎていて、教材とする場合、講義の中に教員の自由な説明が取り入れにくいということです。講義では教員が学生に問題を投げかけ、学生はそれについて自分の頭で考えることが最も重要な気がします。この教材では知識は出来るだけコンパクトにまとめられ、例文には殆ど解説がありません。解説は教員が自分の判断で行います。学生は講義を聞き、行間を埋めることによって教科書を作り上げて行きます。

私は一部の例外を除けば、プログラムは「習うより慣れろ」であると信じています。そこでこの教材の別冊には演習書並の分量の問題が用意されており、解答は最後にまとめられています。演習は徹底的に基礎的な問題から出発して，だんだんと頭を慣らしてゆくような構成になっています。特に初心者の方は初めから自力で問題を解いて下さい。必ずや複雑なプログラムにも耐え得る基礎力が付いてくることでしょう。解答はスマートさよりも分かり易さを優先させていますが，コンパクトに出来る部分は注意書きのところに書いているつもりです。プログラムは LSIC86 で動作を確認しています。問題は基本的な問題と少し考える問題に分け，それぞれ問題番号の後ろに a と b を付けています。

この教材を利用して学生が講義中に自分の頭で考える訓練をし、後は厳密な参考書で勉強を進める。私にとってこれほどの喜びはありません。

福山平成大学経営学部経営情報学科　福井正康

1章　2進法/10進法/16進法

1.1 2進数・10進数・16進数の例

1_B	1	1_H
10_B	2	2_H
11_B	3	3_H
100_B	4	4_H
:	:	:
1000_B	8	8_H
:	:	:
1010_B	10	A_H
:	:	:
10000_B	16	10_H
:	:	:
100000_B	32	20_H
:	:	:
100000000_B	256	100_H

(Binary/Decimal/Hexadecimal)

1.2 相互の変換

2進数 → 10進数

$10101_B \rightarrow 2^4+2^2+2^0 = 16+4+1=21$

$100001_B \rightarrow 2^5+2^0 = 32+1=33$

10進数 → 2進数

		余り			なぜか	余り
2)	101	1	下位	10_B)	1100101_B	1
2)	50	0		10_B)	110010_B	0
2)	25	1		10_B)	11001_B	1
2)	12	0		10_B)	1100_B	0
2)	6	0		10_B)	110_B	0
2)	3	1		10_B)	11_B	1
2)	1	1	上位	10_B)	1_B	1

101 = 1100101_B

16進数 → 10進数

$2B_H \rightarrow 2\times16^1+11\times16^0 = 32+11 = 43$

$1AC_H \rightarrow 1\times16^2+10\times16^1+12\times16^0 = 256+160+12 = 428$

10 進数 → 16 進数

	余り			なぜか	余り
16)	428	12	下位	10_H) $1AC_H$	C_H
16)	26	10		10_H) $1A_H$	A_H
16)	1	1	上位	10_H) 1_H	1_H

2 進数 ↔ 16 進数

$110|1011_B \leftrightarrow 60_H+B_H=6B_H$

$1|0110|1001_B \leftrightarrow 100_H+60_H+9_H=169_H$

問題

1）2 進数に直せ

FF_H, $1A_H$, 27, 30

$FF_H=11111111_B$, $1A_H=11010_B$

$27=1B_H=11011_B$, $30=1E_H=11110_B$

2）16 進数に直せ

11000111_B, 101111_B, 1101100_B

$11000111_B(=C7_H)$, $101111_B(=2F_H)$, $1101100_B(=6C_H)$

1.3 ビットとバイト

単位

ビット(bit:binary digit):	2 進法の 1 桁（2 状態）
バイト(BYTE):	2 進法の 8 桁
	16 進法の 2 桁（256 状態）
ワード(WORD):	2 進法の 16 桁
	16 進法の 4 桁（65536 状態）
約 1,000 バイト(1024 BYTE)	1 KB
約 1,000,000 バイト(1024^2 BYTE)	1 MB
約 1,000,000,000 バイト(1024^3 BYTE)	1 GB

1.4 負数の２進数表現

基本原理

Q：計算機の中でいかに引き算を速く、簡単にするか

A：すべて足し算で行なう

1バイト　1-1=1+(-1)=0

1バイト数

```
    0000 0001B                  0000 1010B
+)  1111 1111B              +)  1111 0110B
  1|0000 0000B                1|0000 0000B
```

負数の表現

Nバイトで表わされる数なら、絶対値の2進表現の8Nビット分の0と1を逆転させ、最後の1ビットに1を加える。

-27=-1BH(1BYTE)　　-32=-20H(2BYTE)

```
 1BH  0001|1011B          20H  0000|0000|0010|0000B
      1110|0100B               1111|1111|1101|1111B
-1BH  1110|0101B         -20H  1111|1111|1110|0000B
```

2進数の正負は第1バイト目が0か1かによる

1バイト		2バイト	
最大	0111|1111B = 127	最大	0111|1111|1111|1111B
最小	1000|0000B =-128	最小	1000|0000|0000|0000B

1.5 小数の２進数表現

10進数 → 2進数

0.65625 → 2進数

```
 0.65625                                    0.
×      2
 1.31250 →  0.3125                          1
          ×      2
           0.6250                           0
         ×      2
           1.2500 →   0.25                  1
                    ×    2
                     0.50                   0
                    ×    2
                     1.00 →     0    1
                                    0.10101B
```

0.65625 = 0.10101B

なぜ2を掛けて行くか？

```
 0.10101B                                   0.
×      10B
```

$1.01010_B \rightarrow 0.0101_B$ 1

$\times \quad 10_B$

0.1010_B 0

$\times \quad 10_B$

$1.0100_B \rightarrow 0.01_B$ 1

$\times \quad 10_B$

0.10_B 0

$\times \quad 10_B$

$1.00_B \rightarrow 0_B$ 1

10 進数で有限な値も 2 進数では無限小数となる

0.1 0.

$\times \quad 2$

0.2 0

$\times \quad 2$

0.4 0

$\times \quad 2$

0.8 0

$\times \quad 2$

1.6 → 0.6 1

$\times \quad 2$

1.2 → 0.2 1

$0.0\underline{0011}|0011|0011|0011|0011\cdots_B$

2 進数 → 10 進数

$0.1011_B = 1011_B/10000_B$

$= 11/16$

$= 0.6875$

問題

1）2 進数に直せ

0.125, 0.3

$0.125 = 0.01_B$

$0.3 = 0.0\underline{1001}\cdots_B$

2）10 進数に直せ

0.1101_B, 0.0101_B

$0.1101_B = 1101_B/10000_B = 13/16 = 0.8125$

$0.0101_B = 101_B/10000_B = 5/16 = 0.3125$

1.6 数値データの内部表現

IEEE (Institute of Electrical and Electronic Engineering)
アメリカ電気電子学会の標準規格

文字　1 バイト（char 型）

```
  7| 6 5 4 3 2 1 0
  0| 0 1 0 1 1 0 1 B
 符号     7
```

最大　0111 1111B　　127

最小　1000 0000B　　-128

短い整数　2 バイト（short 型）

```
 151413121110 9 8    7 6 5 4 3 2 1 0
  0|0 0 0 0 0 1 1 | 0 1 1 0 1 0 1 1 B
 符号              15
```

最大　0111 1111 1111 1111B　　32767

最小　1000 0000 0000 0000B　　-32768

自然整数　2,4 バイト（int 型 CPU 依存）

同上または同下

倍長整数　4 バイト（long 型）

```
  0|000 1011 0111 0011 1111 0110 0001 1000B
符号
```

最大　0111 1111 1111 1111 1111 1111 1111 1111B　　2147483647

最小　1000 0000 0000 0000 0000 0000 0000 0000B　　-2147483648

符号なし整数

unsigned char 型, unsigned short 型, unsigned int 型, unsigned long 型

単精度実数　4 バイト（float 型）

```
    30         22                          0
  0|00000010|0100000 10010110 00110101B
   1       8                 23
符号   指数部             仮数部
```

仮数部の最大桁は常に１になるように桁取りし、最初の１は省略

$0.000101_B \rightarrow 1.01_B \times 2^{-3} \rightarrow 0100\cdots$　　Economized Form

10 進数約 7 桁

指数部はゲタばき表現(Biased Exponent)、常に 01111111B(=127)をたして記録する

0 → 0000 0000B+0111 1111B=0+127
=0111 1111B
-126 → 1000 0010B+0111 1111B=-126+127=1　最小:2^{-126} ～ 10^{-38}
=0000 0001B
127 → 0111 1111B+0111 1111B=127+127=254　最大:2^{127} ～ 10^{38}
=1111 1110B
0000 0000B は－∞
1111 1111B は＋∞　と考える

倍精度実数　8 バイト（double 型）

6362　52 51　0
0|000 0000 0000|0000 0000 0000 ･･･ 0000B
1　11　52
符号　指数部　仮数部

仮数部の最大桁は常に 1 になるように桁取りし、最初の 1 は省略
10 進数約 16 桁

指数部はゲタばき表現、常に 011 1111 1111B(=1023)をたして記録する

0 → 000 0000 0000B+011 1111 1111B=0+1023
=0111 1111B
-1022 → 000 0000 0001B　最小:2^{-1022} ～ 10^{-307}
1023 → 111 1111 1110B　最大:2^{1023} ～ 10^{307}
000 0000 0000B は－∞
111 1111 1111B は＋∞　と考える

1.7 文字データの内部表現

JIS 8 単位コード

1 文字に 1 つずつ背番号を付ける
英字　52
数字　10　＋　カナ　64
記号　33
制御文字　33
128　＋　64　=　192
ASCII　7 ビット　JIS 8 ビット
漢字の表現
2 バイト表現　256×256＝65536 文字

区点コード

```
 1 2 3 4   .......   94 点
 2
 3
 4          第１水準  ... 4418 中 3418
 .
 .
47          .......
-------------------------
48          .......
 .
 .          第２水準  ... 4418 中 3384
 .
94          .......
区
```

JIS (Japanese Industrial Standard) コード

JIS コード＝区点コードの上下桁 16 進化＋2020H

空白 01 区 01 点

0101H + 2020H = 2121H

漢　　20 区 33 点

1421H + 2020H = 3441H

字　　27 区 90 点

1B5AH + 2020H = 3B7AH

KI コード(ESC K), KO コード(ESC H)で漢字と半角英数字との区別をつける。

シフト JIS コード

MS-DOS などで用いられる内部表現

第１バイト目のコードで漢字と半角英数字との区別をつける。

第１バイト目を JIS １バイトコードと違える。

81H～9FH, E0H～FCH（１バイトコードのグラフィック文字）

JIS コードとの計算による１対１の対応あり

EUC (Extended UNIX Code)

UNIX 上で広く使われている 8 ビット系漢字コード

JIS コードの第 1 バイト、第 2 バイトの両方に 0x80 を加算して表す。

半角カナ文字については、前に 1 バイト (0x8e) をつけ、2 バイトで表す。

1.8 論理演算

真を 1、偽を 0 とする

論理和と論理積

否定(NOT)

A	NOT A
0	1
1	0

A	B	A OR B	A AND B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

論理和　論理積

組合せ演算

A	B	NOR A NOR B	NAND A NAND B	排他的論理和(XOR) A XOR B
0	0	1	1	0
0	1	0	1	1
1	0	0	1	1
1	1	0	0	0

A XOR B=(A AND (NOT B)) OR ((NOT A) AND B)

２章　Ｃ言語の考え方

2.1 Ｃと BASIC の比較

例　肥満度の判定（ＢＡＳＩＣ）

```
'肥満度の判定
PRINT "身長？>>";
INPUT "",HEIGHT
PRINT "体重？>>";
INPUT "",WEIGHT
GOSUB HIMAN
END
'肥満度の計算
HIMAN:
HYOJUN=(HEIGHT-100.0)*0.9
HIMANDO=WEIGHT/HYOJUN
GOSUB HANTEI
RETURN
'判定
HANTEI:
PRINT "肥満度 =";HIMANDO
IF HIMANDO>=1.2 THEN
  PRINT "太り過ぎです。"
ELSEIF HIMANDO<=0.8 THEN
  PRINT "痩せ過ぎです。"
ELSE
  PRINT "標準的です。"
END IF
RETURN
```

例　肥満度の判定（Ｃ）

```
/* 肥満度の判定 */
#include <stdio.h>
void himan(float,float),hantei(float);
main()
```

```
{
    float height,weight;
    printf("身長？>>");
    scanf("%f",&height);
    printf("体重？>>");
    scanf("%f",&weight);
    himan(height,weight);
}
/* 肥満度の計算 */
void himan(float shincho,float taiju)
{
    float hyojun,himando;
    hyojun=(shincho-100.0)*0.9; /* 標準体重の計算 */
    himando=taiju/hyojun;       /* 肥満度の計算 */
    hantei(himando);
}
/* 判定 */
void hantei(float himando)
{
    printf("肥満度 = %f\n",himando);
    if(himando>=1.2) printf("太り過ぎです。\n");
    else if(himando<=0.8) printf("痩せ過ぎです。\n");
    else printf("標準的です。\n");
}
```

2.2 C 言語の特徴

特徴

1．変数型の定義が厳密
2．プログラムはすべて関数の集まり。
3．関数は戻り値を持ち、関数に型がある。
4．関数への変数の渡し方は、変数名には縛られない。

3章　変数とポインタ

3.1 ポインタの基礎

概略

Basic では、変数 X はその値だけが意味を持ちました。しかし、C ではその値だけでなく、その変数がメモリ上のどこに格納されているのかを利用する場合がよくあります。

int x; と定義すると、メモリ上のある 2 バイトがプログラムにより確保され、x=1 でその中に 1 という値が格納されます。この格納域の先頭の位置を示すのがポインタと呼ばれるもので、この場合 &x で表されます。このように変数 x の値からポインタを求めるには & を付けてやります。また逆に、ポインタから変数の値を求めるには * を付けてやります。則ち、*&x は x と同じです。

int *x; と定義すると、x は整数型の数値を格納する位置を指すポインタを格納する変数（2 または 4 バイト）になります。*x が値を表しまが、すぐに *x=1; などとすると整数を格納する領域を確保していませんので警告が発せられます。

int a[10]; で定義される変数は整数型の領域を 10 個連続して繋いだ配列変数で、先頭要素の値は a[0] です。また、前に述べた規則から、この要素の位置を表すポインタは &a[0] です。これは表現するのが厄介ですので a（または a+0）という配列名そのもので表すことも出来ます。また、同じく前に述べた規則から、先頭の a[0] という数値は *a で表すことも出来ます。次に配列の i 番目の要素の値は a[i]で、この要素へのポインタは &a[i]です。これはまた、a+i とも表わされます。そしてその要素の値 a[i]は *(a+i) と表すことも出来ます。

同様に char str1[20]; は char 型配列（通常の文字列）として 20 バイト確保する定義ですが、char *str2; は str2 が単に char 型の変数域を指しているポインタを格納する領域であるという定義です。str2="FUKUYAMA"; ではこのポインタが "FUKUYAMA"という文字列の格納してある領域の先頭を指すようにしています。この場合、*str2 則ち str2[0] は F という先頭の文字の ASCII コードを値として持っています。

文字列として扱われる "FUKUYAMA"のようなデータは、メモリ上にそれぞれ

の文字の ASCII コードが順番に並んでいますが、最後に 1 つ余分に ASCII コードの 0 (null 文字)が入っており、この場合 9 バイトの領域を使っていることに注意して下さい。

以上まとめると

	値	ポインタ
int x; の場合	x	&x
int *x; の場合	*x	x
int a[10](*a); の場合	a[i],*(a+i)	&a[i],a+i

例 3-1 変数の値とポインタ

```
#include <stdio.h>
main()
{
    int i,x1,*x2;
    int a[10],b[10];
    char str[10]="FUKUYAMA";
    x1=1;
    printf("%d\n",x1);
    x2=&x1;                /* いきなり *x2=3; のような代入はエラー */
    printf("%d\n",*x2);
    for(i=0;i<=9;i++) a[i]=i;     /* FOR I=0 TO 9: A(I)=I: NEXT I */
    for(i=0;i<=9;i++) *(b+i)=i;   /* 相当する BASIC の書式はない  */
    for(i=0;i<=9;i++) printf("%d ",a[i]);
    printf("\n");
    for(i=0;i<=9;i++) printf("%d ",b[i]);
    printf("\n");
    printf("%s\n",str);
}
```

4章　基本的な入出力

4.1 出力

printf 関数

関数　printf(format[,arg1,arg2,･･･])
機能　format の形式による引数 arg1,arg2,･･･ の出力
備考　%[-][[0]w][.n][l]変換文字
　　変換文字　d,u,o,x,X,c,s,e,E,f

例 4-1 printf 関数の使用法

```
#include <stdio.h>
main()
{
    int age;
    long size;
    float weight;
    double pix;
    char *str;
    printf("%d\n",27);
    printf("%x\n",27);
    printf("%X\n",27);
    printf("%o\n",27);
    printf("%u\n",-1);
    age=15;
    printf("age[%3d]\n",age);  /* 右詰め３桁表示,\n は改行の印 */
    printf("age[%03d]\n",age); /* 右詰め３桁表示,空白は０でうめる */
    printf("age[%-3d]\n",age); /* 左詰め３桁表示 */
    size=100000L;
    printf("File size=%ld\n",size);
    weight=65.5;
    printf("weight=%f\n",weight);
    printf("weight=%5.1f\n",weight); /* 全体５桁小数点以下１桁 */
    pix=314.15926;
```

```
    printf("PIX = %10.6f\n",pix);
    printf("PIX = %15.8e\n",pix);
    printf("Fukuyama Heisei Univ.\n");
    printf("\x1b[31mFukuyama Heisei Univ.\x1b[0m\n");
    printf("%c%c%c\n",0x41,0x42,0x43);
    str="DORAEMON";
    printf("%s\n",str);
    printf("name=%s,age=%d\n",str,age);
}
```

例 4-2 データ型とキャスト演算子

```
#include <stdio.h>
main()
{
    int a=8,b=3,c,d;
    float x=8.0,y=3.0,z;
    c=a/b;
    d=a-a/b*b;
    z=(float)a/b;          /* z=a/b; 正しくない */
    printf("商 = %d  余り = %d\n",c,d);
    printf("小数解 = %f\n",z);
    c=x/y;                 /* c=(int)x/(int)y; 正しい */
    d=x-(int)(x/y)*y;      /* d=(int)x-(int)x/(int)y*(int)y 正しい */
    z=x/y;
    printf("商 = %d  余り = %d\n",c,d);
    printf("小数解 = %f\n",z);
}
```

4.2 入力

scanf 関数

関数　scanf(format,addr1[,addr2,･･･])
機能　format の形式による引数 arg1,arg2,･･･ の入力
備考　入力幅無指定の場合入力間は空白, タブ, リターン

例 4-3 scanf 関数の使用法

```
#include <stdio.h>
main()
{
    char str[21];
    int a,b;
    float x;
    double y;
    printf("input str >>");
    scanf("%s",str);
    printf("str=%s\n",str);
    printf("input int >> ");
    scanf("%d",&a);
    printf("a=%d\n",a);
    printf("input int float >> ");
    scanf("%d%f",&a,&x);  /* 入力区切りは空白、タブ、リターン */
    printf("a=%d x=%f\n",a,x);
    printf("input int,int >> ");
    scanf("%d,%d",&a,&b); /* 入力区切りは , */
    printf("a=%d b=%d\n",a,b);
    printf("6 桁の整数 >> ");
    scanf("%3d%3d",&a,&b);
    printf("a=%d b=%d\n",a,b);
    printf("input double >> ");
    scanf("%lf",&y);     /* double 型の入力には l を付ける */
    printf("y=%f\n",y);  /* しかし、出力には付けない */
}
```

4.3 1 文字の入出力

1 文字入出力関数

```
関数   putchar(文字)       /* 詳細は後に学ぶ */
機能   標準出力へ 1 文字出力
例     putchar('a'); putchar(0x71); putchar(c); /* int c; */
```

関数　文字=getchar()　　/* 詳細は後に学ぶ */
機能　標準入力から１文字入力
例　　c=getchar(); /* int c */
備考　プログラムの実行を１時中断する場合にも getchar(); と用いる

5章　条件判断

5.1 プログラムの構造化

構造化とは

GOTO 文を排除し、以下の制御構造だけでプログラミングを記述する。

連接(sequence)
判断(if else)
多方向分岐(switch case)
所定回反復(for)
前判断反復(while)
後判断反復(do while)

5.2 if 文

if 文の書式

```
書式  if(式){
          文並び;
      }
      if(式) 単一文;
```

例 5-1 if 文の利用法

```
#include <stdio.h>
main()
{
    int age;
    printf("何才ですか？ ");
    scanf("%d",&age);
    if(age<23){
        printf("若いですね。\n");
        printf("学校に通っているんですか。\n");
    }
    if(age>=23) printf("お元気そうですね。\n");
}
```

if else 文の書式

```
書式  if(式){                    if(式) 単一文;
          文並び１;              else if(式) 単一文;
      }                          ・・・・
      else if(式){               else 単一文;
          文並び２;
      }
      ・・・・・
      else {
          文並びｎ;
      }
```

例 5-2 if else 文の利用法

```
#include <stdio.h>
main()
{
    int age;
    printf("何才ですか？ ");
    scanf("%d",&age);
    if(age<23){
        printf("若いですね。\n");
        printf("学校に通っているんですか。\n");
    }
    else if(age<60) printf("元気に働いてますか？\n");
    else{
        printf("お元気そうですね。\n");
        printf("お体を大切に。\n");
    }
}
```

5.3 関係演算子と論理演算子

関係・論理演算子の種類

関係演算子	論理演算子
>	! NOT
>=	&& AND

```
<                                   ||   OR
<=
==          (= ではない)
!=          (NOT EQUAL)
```

条件式の例

```
if(height>=150&&height<=180)
if(weight<40||weight>90)
if(c!='a'&&c!='A')  ⇔  if(!(c=='a'||c=='A'))
if(c==0x1b)
```

5.4 真偽の値と代入式の値

例 5-3 真と偽の値

```
#include <stdio.h>
main(){
    int c;
    c=2;
    printf("%d\n",c==2);
}
```

注）条件式は真ならば 1、偽ならば 0 の値を返す。

例 5-4 代入式の値

```
#include <stdio.h>
main(){
    int c;
    c=0;
    if(c=0) printf("c は 0 です\n");
    else printf("c は 0 以外です\n");
}
```

注）c=0 は式として 0 の値を取り、条件式としては常に偽

5.5 if 文のネスト

例 5-5 if 文のネスト

```
#include <stdio.h>
main()
```

```
{
    int univ,ten;
    printf("A校: 0  B校: 1  どちらですか > ");
    scanf("%d",&univ);
    printf("点数 > ");
    scanf("%d",&ten);
    if(univ==0){
        if(ten>=80) printf("優\n");
        else if(ten>=70) printf("良\n");
        else if(ten>=60) printf("可\n");
        else printf("不可\n");
    }
    else{
        if(ten>=80) printf("A\n");
        else if(ten>=70) printf("B\n");
        else if(ten>=50) printf("C\n");
        else printf("D\n");
    }
}
```

5.6 switch ～ case 文

switch ～ case 文の書式

```
書式  switch(式){
        case 値1: 処理1; 処理2; …; break;
        case 値2: 処理1; 処理2; …; break;
          ……
        case 値n: 処理1; 処理2; …; break;
        default:  処理1; 処理2; …;
      }
機能  式の値に応じた値ｉの後ろの処理が実行される
      該当する値ｉがない場合は default の後ろの処理が実行される
備考  式と値は整数値でなければならない
```

例 5-6 色を指定した表示

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

```
main()
{
    int c;
    printf("color no.(1-7)>>");
    scanf("%d",&c);
    switch(c){
        case 1 :printf("\x1b[34m"); break;
        case 2 :printf("\x1b[32m"); break;
        case 3 :printf("\x1b[36m"); break;
        case 4 :printf("\x1b[31m"); break;
        case 5 :printf("\x1b[35m"); break;
        case 6 :printf("\x1b[33m"); break;
        case 7 :printf("\x1b[37m"); break;
        default:printf("\x1b[0;1m");
    }
    printf("C programming\n");
    printf("\x1b[0;1m");
}
```

6章　反復

6.1 for 文

for 文の書式

```
書式　for(初期設定 ; 条件判断 ; 繰り返し時の処理){
          文並び; ･･･
      }
機能　繰り返し・特に所定回の繰り返しに用いられる
例　　for(i=1;i<=10;i++){
          printf("fukuyama\n");
      }           /* 文が１つの場合は { } は省略可 */
      for(j=2;j<=max;j=j+2) printf("No.%d\n",j);
備考　i++ (i=i+1 のこと) i-- (i=i-1 のこと)
      i+=a (i=i+a のこと) i-=a (i=i-a のこと)
      無限ループ　for(;;){ ･･･ }
```

例 6-1 繰り返し回数が決まっている場合

```
#include <stdio.h>
main()
{
    int i;
    for(i=1;i<=10;i++)
        printf("%4d",i);
    printf("\n");
}
```

6.2 while 文

while 文の書式

```
書式　while(条件判断){
          文並び; ･･･
      }           /* 文が１つの場合は { } は省略可 */
機能　繰り返し・条件を参照しながらの繰り返し
例　　while(c!='a') c=getchar();
備考　無限ループ　while(1){ ･･･ }
```

例 6-2 繰り返し回数が決まっている場合

```
#include <stdio.h>
main()
{
    int i;
    i=1;
    while(i<=10){
        printf("%4d",i);
        i++;
    }
    printf("\n");
}
```

例 6-3 繰り返し回数が決まっていない場合

```
#include <stdio.h>
main()
{
    int x,total;
    total=0;
    scanf("%d",&x);
    while(x!=0){
        total=total+x;
        scanf("%d",&x);
    }
    printf("合計 = %d\n",total);
}
```

例 6-4 合計と平均１

```
#include <stdio.h>
main()
{
    int c;
    float dat,sum,n;
```

```
    sum=n=0; /* sum=0; n=0; に同じ */
    c=scanf("%f",&dat);
    while(c!=EOF){
        sum=sum+dat;
        n++;
        c=scanf("%f",&dat);
    }
    printf("合計 = %f\n",sum);
    if(n>0) printf("平均 = %f\n",sum/n);
}
/* EOF は stdio.h の中で #define EOF -1 と定義されている */
```

例 6-5 合計と平均２

```
#include <stdio.h>
main()
{
    float dat,sum,n;
    sum=n=0;
    while(scanf("%f",&dat)!=EOF){
        sum=sum+dat;
        n++;
    }
    printf("合計 = %f\n",sum);
    if(n>0) printf("平均 = %f\n",sum/n);
}
```

例 6-6 キーの読み取り

```
#include <stdio.h>
#include <conio.h>
main()
{
    int c;
    while((c=getch())!=0x1b)
        printf("%x ",c);
```

```
}
```

6.3 do ～ while 文

do ～ while 文の書式

書式　do{
　　文並び; ･･･
　}while(条件判断);

機能　繰り返し・ループの最後に条件を参照しながらの繰り返し

例 6-7 キーの読み取り

```
#include <stdio.h>
#include <conio.h>
main()
{
    int c;
    do {
        c=getch();
        printf("%x ",c);
    } while(c!=0x1b);
}
```

6.4 break 文による抜け出し

例 6-8 無限ループからの抜け出し

```
#include <stdio.h>
#include <conio.h>
main()
{
    int c;
    while(1){
        c=getch();
        if(c==0x1b){
            printf("\nESC が入力されました。\n");
            break;
```

```
        }
        else if(c==0xd){
            printf("\nEnter が入力されました。\n");
            break;
        }
        else printf("%xH  ",c);
    }
}   /* 抜け出し方によって処理の変るときに便利 */
```

例 6-9 抜け出しの段数

```
#include <stdio.h>
main()
{
    int i,j;
    i=1;
    while(1){
        if(i>10) break;
        printf("loop = %d\n",i);
        j=1;
        while(1){
            if(j>10) break;
            printf("  %d",j);
            j++;
        }
        printf("\n");
        i++;
    }
}   /* 抜け出しは１段ループだけ */
```

6.5 goto 文

goto 文の書式

書式　goto label
ラベル　　label:

備考　なるべく使用しない。１つの関数内の小規模なジャンプにとどめる。

7章　配列

7.1 配列と表式

例 7-1　配列への値の代入と参照

```
#include <stdio.h>
main()
{
    int i,a[20];
    float mat[10];
    char c[21];

    for(i=0;i<=19;i++) a[i]=i;
    for(i=0;i<=19;i++) printf("%d ",a[i]);
    printf("\n");
    for(i=0;i<=19;i++) printf("%d ",*(a+i));
    printf("\n");
    for(i=0;i<=9;i++) mat[i]=(float)(i*i);
    for(i=0;i<=9;i++) printf("%5.1f",mat[i]);
    printf("\n");
    for(i=0;i<=9;i++) printf("%5.1f",*(mat+i));
    printf("\n");
    for(i=0;i<=19;i++) c[i]=0x61+i;
    c[20]=0x0;
    for(i=0;i<=19;i++) printf("%c",c[i]);
    printf("\n");
    printf("%s\n",c);
}
```

7.2 2次元配列

例 7-2　２次元配列への値の代入と参照

```
#include <stdio.h>
main()
```

```
{
    int i,j,no[5][10];

    for(i=0;i<=4;i++){
        for(j=0;j<=9;j++) no[i][j]=i*10+j;
    }
    for(i=0;i<=4;i++){
        for(j=0;j<=9;j++) printf("%5d",no[i][j]);
        printf("\n");
    }
    printf("\n");
    for(i=0;i<=4;i++){
        for(j=0;j<=9;j++) printf("%5d",*(no[i]+j));
        printf("\n");
    }
    printf("\n");
    /* 次の意味が分かれば偉い！ */
    for(i=0;i<=4;i++){
        for(j=0;j<=9;j++) printf("%5d",*(*(no+i)+j));
        printf("\n");
    }
    printf("\n");
}
```

7.3 文字配列

例 7-3　文字列としての文字配列

```
#include <stdio.h>
main()
{
    int i;
    char a[10],b[2][10];
```

```
    a[0]='H';  a[1]='e';  a[2]='l';  a[3]='l';  a[4]='o'; a[5]='\0';
    b[0][0]='M';  b[0][1]='y';  b[0][2]=0x0;
    b[1][0]='f';  b[1][1]='r';  b[1][2]='i'; b[1][3]='e';
    b[1][4]='n';  b[1][5]='d';  b[1][6]=0;

    printf("%s\n",a);
    for(i=0;i<2;i++) printf("%s ",b[i]);
    printf("\n");
}
```

7.4 配列の初期化

例 7-4　配列の初期化の方法

```
#include <stdio.h>
main()
{
    int i,j;
    int x1[5]={1,2,3,4,5};
    int x2[]={1,2,3,4,5};
    int y1[2][5]={{1,2,3,4,5},
                  {6,7,8,9,10}};
    int y2[][5]={{1,2,3,4,5},{6,7,8,9,10}};
/*  int y2[][]={{1,2,3,4,5},{6,7,8,9,10}}; とは出来ない */

    for(i=0;i<=4;i++) printf("%d ",x1[i]);
    printf("\n");
    for(i=0;i<=4;i++) printf("%d ",x2[i]);
    printf("\n");
    for(i=0;i<=1;i++) for(j=0;j<=4;j++) printf("%d ",y1[i][j]);
    printf("\n");
    for(i=0;i<=1;i++) for(j=0;j<=4;j++) printf("%d ",y2[i][j]);
    printf("\n");
}
```

7.5 文字配列の初期化

例 7-5　文字配列の初期化の方法

```
#include <stdio.h>
main()
{
    int i;
    char a[10]={'F','u','k','u','y','a','m','a',0x0};
    char b[]={'H','i','r','o','s','h','i','m','a',0x0};
    char c[20]="Okayama";
    char d[]="Onomichi";
    char e[2][10]={{'Y','A','M','A',0x0},{'K','A','W','A',0x0}};
    char f[3][10]={"UMI","SHIMA","NAMI"};

    printf("%s\n",a);
    printf("%s\n",b);
    printf("%s\n",c);
    printf("%s\n",d);
    for(i=0;i<2;i++) printf("%s ",e[i]);
    printf("\n");
    for(i=0;i<3;i++) printf("%s ",f[i]);
    printf("\n");
}
```

7.6 ポインタ配列

例 7-6　ポインタ配列の利用

```
#include <stdio.h>
main()
{
    int i;
    char *univ[3];
    char *city[3]={"fukuyama","hiroshima","okayama"};
```

```
    univ[0]="fukuyama";
    univ[1]="hiroshima";
    univ[2]="okayama";
    for(i=0;i<=2;i++) printf("%s\n",univ[i]);
    for(i=0;i<=2;i++) printf("%s\n",city[i]);
}
```

7.7 注意事項

Basic との相違点

配列の表示は Basic ではその成分値だけしか表せなかったが、C では様々な表示法が可能である。その分、習得が難しいが、慣れるとこれほど柔軟で便利なものはない。また、これを自分のものにすると、他の言語の変数使用法が簡単に理解出来る。

7.8 配列のポインタ

例 7-7　ポインタの表示

```
#include <stdio.h>
main()
{
    int i;
    int a[5];
    double x[5];
    for(i=0;i<5;i++) a[i]=i;
    for(i=0;i<5;i++) x[i]=(double)i;
    for(i=0;i<5;i++) printf("%pH ",a+i);
    printf("\n");
    for(i=0;i<5;i++) printf("%pH ",x+i);
    printf("\n");
}
```

8章　関数

8.1 関数の基本形

関数の書式

```
型 関数名([引数並びと型宣言])
{
    ....
    本文;
    ....
    [return 式;]
}
```

例：

```
float absol(float x)
{
    if(x<0) return -x;
    else return x;
}
```

注意　return 式 の式の型は関数の型と一致させる
　　関数の型を省略すると int 型になる

8.2 関数とは

例 8-1　関数の使用例

```
#include <stdio.h>
double fact(int); /* 関数のプロトタイプ宣言 */
main()
{
    int k;
    double x;
    printf("n! の計算 n >> ");
    scanf("%d",&k);
    x=fact(k);
    if(x>0) printf("%d!=%.0f\n",k,x);
    else printf("error!!\n");
}
double fact(int no)
```

```
{
    int k;
    double s=1.0;
    if(no<0) return 0;
    for(k=1;k<=no;k++) s=s*(double)k;
    return s;
}
```

8.3 グローバル変数とローカル変数

例 8-2　変数のスコープ

```
#include <stdio.h>
void sub1(void),sub2(void);
int n0;       /* グローバル変数 */
main()
{
    int n1;
    n0=0;
    n1=0;
    printf("main: n0=%d  n1=%d\n",n0,n1);
    sub1();
    printf("main: n0=%d  n1=%d\n",n0,n1);
    sub2();
    printf("main: n0=%d  n1=%d\n",n0,n1);
}
void sub1(void)
{
    int n1;
    n0=1;
    n1=1;
    printf("sub1: n0=%d  n1=%d\n",n0,n1);
}
void sub2(void)
```

```
{
    int n0;
    n0=2;
    printf("sub2: n0=%d\n",n0);
}
```

注意

１）局所変数は各関数（ブロック）内で値が完全に独立。

２）大域変数はどの関数内でも値を変化出来る。

３）大域変数と局所変数が同じ名前なら局所変数として使用される。

8.4 引数の値渡しとアドレス渡し

例 8-3　値渡しとアドレス渡しの違い

```
#include <stdio.h>
void sub1(int),sub2(int *);
main()
{
    int n;
    n=0;
    printf("main  n=%d\n",n);
    sub1(n);
    printf("main  n=%d\n",n);
    sub2(&n);
    printf("main  n=%d\n",n);
}
void sub1(int no)    /* 値渡し */
{
    printf("sub1 hikisu = %d\n",no);
    no=1;
    printf("sub1 hikisu = %d\n",no);
}
void sub2(int *no)   /* アドレス（ポインタ）渡し */
{
```

```
    printf("sub2 hikisu = %d\n",*no);
    *no=2;
    printf("sub2 hikisu = %d\n",*no);
}
```

注意

１）値渡しは変数を変化させても元の値には影響しない。

２）アドレス渡しは変数を変化させると元の値も変る。

例 8-4　文字配列のアドレス渡しとコメントの例

```
#include <stdio.h>
#include <string.h>
int print(int,int,char *);
main()
{
    char *data;
    int c;
    data="福山大学";
    c=print(20,5,data);
    printf("  バイト数 = %d\n",c);
}
/*
関数  print(x,y,str)
機能  x 桁 y 行 に文字列 str を出力 画面左上は (0,0)
戻値  出力した文字数
例    print(30,5,"福山");
*/
int print(int x,int y,char *str)
{
    int len;
    printf("\x1b[%d;%dH",y+1,x+1);
    printf("%s",str);
    len=strlen(str);
    return len;
```

```
}
```

8.5 関数プロトタイプ宣言

関数プロトタイプ宣言とは

関数型の定義で引数の型を付けて定義することが出来ます。その場合にはコンパイラは引数の数のチェックと関数呼び出しの際の強制的な型変換を行ってくれます。

例 8-5　関数プロトタイプ宣言の効果

```
#include <stdio.h>
void pr1(int),pr2();
/* pr1(int n) のように変数名を付けても同じ */
/* int pr3(int); は省略されている */
main()
{
    int a=5;
    float x=2.6789;
    pr1(a);
    pr1(x);  /* 自動的に型変換が起る */
    pr2(a);
    pr2(x);  /* 実行結果エラー */
    pr2((int)x);
    pr3(a);
    pr3(x);  /* 実行結果エラー */
    pr3((int)x);
}
void pr1(int n)
{
    printf("pr1=%d\n",n);
}
void pr2(int n)
{
    printf("pr2=%d\n",n);
```

```
}
int pr3(int n)
{
    printf("pr3=%d\n",n);
    return n;
}
```

8.6 main 関数の引数

例 8-6　コマンドラインの引数

```
/* ファイル名は sample.c */
#include <stdio.h>
main(int argc, char *argv[])
{
    int i;
    if(argc==1){
        printf("main 関数の引数\n");
        printf("sample 引数１ 引数２ ･･･ として\n");
        printf("引数を適当に付けて動かすと引数がどのように\n");
        printf("読み込まれているか分かる。\n");
    }
    else{
        printf("argc=%d\n",argc);
        for(i=0;i<argc;i++){
            printf("argv[%d]: %s\n",i,argv[i]);
        }
    }
}
```

例 8-7　コマンドラインの引数を利用したプログラム

```
/* ファイル名は color.c */
#include <stdio.h>
main(int argc, char *argv[])
{
```

```
    if(argc==1){
        printf("テキスト出力の色を変えるプログラムです。\n");
        printf("color 色番号\n");
        printf("色番号 1:青 2:赤 3:紫 4:緑 5:水 6:黄 7:白\n");
    }
    else if(argc==2){
        if(*argv[1]=='1') printf("\x1b[34mchanged!\n");
        else if(*argv[1]=='2') printf("\x1b[31mchanged!\n");
        else if(*argv[1]=='3') printf("\x1b[35mchanged!\n");
        else if(*argv[1]=='4') printf("\x1b[32mchanged!\n");
        else if(*argv[1]=='5') printf("\x1b[36mchanged!\n");
        else if(*argv[1]=='6') printf("\x1b[33mchanged!\n");
        else if(*argv[1]=='7') printf("\x1b[37mchanged!\n");
        else printf("引数の色番号のエラーです。\n");
    }
    else printf("引数の指定のエラーです。\n");
}
```

注）int argc と char *argv[] の名前は慣例的に用いられる。

8.7 変数の記憶クラス

静的変数

関数内での static 宣言

```
  char str1[40];          ：関数が始まってからメモリ上に取られ、
                            関数の終了で消滅（余分なメモリを取らない）
  static char str2[40]; ：常にメモリ上に存在
                            他の関数からは呼び出せないが値は常に保持
```

関数外での static 宣言

　初期化するとコンパイル時に値が取られること以外に、あまり実用的な差はない

例 8-8　動的変数の破壊と静的変数の生存

```
#include <stdio.h>
void setp(void);
char *p1,*p2,*p3;
main()
```

```
{
    setp();
    printf("main: %s\n",p1);
    printf("main: %s\n",p2);
    printf("main: %s\n",p3);
    printf("\n");
}
void setp(void)
{
    char str1[10]="fukuyama";
    static char str2[10]="okayama";
    p1=str1;  p2=str2;
    p3="hiroshima";
    printf("setp: %s\n",p1);
    printf("setp: %s\n",p2);
    printf("setp: %s\n",p3);
    printf("\n");
}
```

例 8-9　計算への利用

```
#include <stdio.h>
void sum(int);
main()
{
    int i;
    for(i=1;i<=5;i++) sum(i);
}
void sum(int n)
{
    static int total=0;
    total=total+n;
    printf("sum = %d\n",total);
}
```

９章　標準ライブラリ関数

9.1 標準ライブラリの使用法

例 9-1　時刻の取得

```
#include <stdio.h>
#include <time.h>
main()
{
    long t;  /* time_t 型としてもよい */
    time(&t);
    printf("1970/1/1 から%ld 秒経過\n",t);
}
```

例 9-2　乱数の発生

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i;
    unsigned int seed;
    double rnd;
    seed=1;
    srand(seed);
    for(i=1;i<=100;i++){
        rnd=(double)rand()/32768.0;
        printf("%8.4f",rnd);
    }
    printf("\n");
}
```

例 9-3　簡単な算術関数

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

```
main()
{
    double x;
    printf("%10s%10s%10s\n","x","sqrt(x)","pow(2,x)");
    for(x=1;x<=10;x++)
        printf("%10.3f%10.3f%10.3f\n",x,sqrt(x),pow(2.0,x));
}
```

例 9-4　文字列処理

```
#include <stdio.h>
#include <string.h>
main()
{
    int a;
    char str1[41],str2[41];
    strcpy(str1,"Fukuyama");
    strcpy(str2," city");
    a=strlen(str1);
    printf("%s は%d 文字\n",str1,a);
    strcat(str1,str2);
    a=strlen(str1);
    printf("%s は%d 文字\n",str1,a);
    a=strcmp(str1,"Fukuyama city");
    printf("strcmp=%d\n",a);
}
```

例 9-5　文字数字変換

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int a;
    double x;
    char str[20];
```

```
    a=atoi("123");
    printf("a=%d\n",a);
    x=atof("1.23");
    printf("x=%f\n",x);
    x=atof("1.2abc");    /* 1.2 のみ変換  "abc" では 0 */
    printf("x=%f\n",x);

    sprintf(str,"%d",123);
    printf("str=%s\n",str);
    sprintf(str,"%8.4f",1.23);
    printf("str=%s\n",str);
}
```

１０章　種々の演算子

10.1 ビット演算子とシフト演算子

演算子　~

機能　　ビットごとの否定
使用例　char c; p=~c;　int a; if(~a==b){‥‥‥};
数値例　a=0xff1a -> ~a=00e5　/* 足すと 0xffff になる */
　　a=0x0123 -> ~a=fedc

演算子　&

機能　　ビットごとの論理積
使用例　if(c&0x08==0x08) /* 下位から４ビット目が１か否か */
　　int a; a=a&0xff00　/* 下位８ビットを０にマスクする */
数値例　0xa523&0xff00=0xa500

演算子　|

機能　　ビットごとの論理和
使用例　char c; c=c|0x08;　　/* 下位から４ビット目を立てる */
数値例　0xff44&0x0008=0xffc4

演算子　^

機能　　ビット毎の排他的論理和 00->0,01->1,11->0
使用例　int a; a=a^0x00ff;　　/* 下位８ビットのビット反転 */
数値例　0x88ff^0x00ff=0x8800

演算子　<<　/* 左シフト */

機能　　a<<n　a の各ビットを左へ n ビットシフトする
　　空いた右 n ビットには 0 が入れられる
使用例　int a,n; a=a<<n;
数値例　0x1234<<4=0x2340　0x1234<<8=0x3400

演算子　>>　/* 右シフト */

機能　　a>>n　a の各ビットを右へ n ビットシフトする
　　空いた左 n ビットには 0 が入れられる
使用例　int a,n; a=a>>n;

数値例　0x1234>>4=0x0123　0x1234>>8=0x0012

例 10-1　数値の２進数表示

```
#include <stdio.h>
void bitpat(int n)
{
    int i,j;
    for(i=12;i>=0;i=i-4){
        for(j=i+3;j>=i;j--) printf("%d",(n>>j)&0x01);
        if(i!=0) printf(" ");
    }
    printf("B\n");
}
main()
{
    int a1;
    printf("int >> ");
    scanf("%d",&a1);
    printf("%d = %04xH = ",a1,a1);
    bitpat(a1);
}
```

10.2 インクリメント・デクリメント演算子

演算子　++　/* インクリメント演算子 */

機能　　a++;（後置）：a の値を評価して a=a+1
　　++a;（前置）：a=a+1 として a の値を評価する
使用例　printf("%d",a++); は printf("%d",a); a=a+1;
　　printf("%d",++a); は a=a+1; printf("%d",a);　に同じ

演算子　--　/* デクリメント演算子 */

機能　　a--;（後置）：a の値を評価して a=a-1
　　--a;（前置）：a=a-1 として a の値を評価する
使用例　printf("%d",a--); は printf("%d",a); a=a-1;

printf("%d",--a); は a=a-1; printf("%d",a); に同じ

例 10-2　前置型と後置型

```
#include <stdio.h>
main()
{
    int a;
    a=0;
    while(a<10) printf("%3d",a++);
    printf("\n");
    a=0;
    while(a<10) printf("%3d",++a);
    printf("\n");
}
```

例 10-3　* 演算子との混合

```
#include <stdio.h>
main()
{
    char *p;
    p="fukuyma";
    while(*p!=0x0) putchar(*p++);  /* *(p++)と同じ */
    printf("\n");
}
```

10.3 キャスト演算子

演算子　(double)x, (int)y 等

機能　　強制的な型変換

10.4 代入演算子

演算子　+=　-=　*=　/=　%=　　<<=　>>=　&=　|=　^=

機能　　a+=b(a=a+b) etc.　a<<=n(a=a<<n) etc.

10.5 条件演算子

演算子　e1 ? e2 : e3

機能　　もし e1 が真なら値は e2 偽なら値は e3

使用例　c=(x>=0)? x:-x; /* 絶対値 */

10.6 カンマ演算子

演算子　,

機能　　単純に文をつなぐ

使用例　a=1,b=2;　while(scanf("%d",&a),a!=-999){ ...}

補足 enum 型データ

例 10-4　enum 型データの利用

```
#include <stdio.h>
main()
{
    enum color {BLACK,BLUE,GREEN,CYAN,RED,MAGENTA,YELLOW,WHITE} col;
    printf("color no.(0-7)>>");
    scanf("%d",&col);
    switch(col){
        case BLACK   :printf("\x1b[30m"); break;
        case BLUE    :printf("\x1b[34m"); break;
        case GREEN   :printf("\x1b[32m"); break;
        case CYAN    :printf("\x1b[36m"); break;
        case RED     :printf("\x1b[31m"); break;
        case MAGENTA :printf("\x1b[35m"); break;
        case YELLOW  :printf("\x1b[33m"); break;
        case WHITE   :printf("\x1b[0m");  break;
        default      :printf("\x1b[0m");
    }
    printf("C programming\n");
    printf("\x1b[0m");
}
```

１１章　構造体と共用体

11.1 構造体

構造体とは

例えば個人のプロフィールを考えるとき、各人の氏名、住所、電話番号、身長、体重等がデータとなります。これを 100 人分集めるとすると、今までのデータ形式なら各項目毎に配列を用いることになると思われます。しかし、これでは１つ１つのデータはばらばらで、データベースで使われるレコードの概念はありません。そこで、各人のすべてのデータを１つのレコードとして、まとめるデータ構造があれば便利ではないかと思われます。これを実現するのが構造体です。

11.2 構造体の宣言

構造体宣言の書式

書式　struct 構造体名（タグ名） {
　　メンバ型指定;
　　・・・・・・
　} [構造体変数名];

構造体変数名は分離して後に struct タグ名 構造体変数名; で定義出来る。
struct タグ名 構造体配列名;　とすることも出来る。
struct タグ名 構造体ポインタ; とすることも出来る。
struct タグ名 構造体ポインタ配列; も出来る。

例　
```
struct profile {
    char name[21],address[61],tel[13];
    int age;
    float height,weight;
};
struct profile prof1;
struct profile prof2[100];
struct profile *prof3;
struct profile *prof4[100];
```

11.3 構造体データの参照

構造体データの参照の書式

```
x=prof1.height;
x=prof2[20].height;
  /* 構造体変数.メンバ変数 */
x=prof3->height;
x=prof4[20]->height;
  /* 構造体ポインタ->メンバ変数 */
```

11.4 構造体データの代入と初期化

例 11-1　構造体データの代入

```
#include <stdio.h>
#include <string.h>
main()
{
    int i;
    struct profile {
        char name[21];
        int age;
    };
    struct profile prof1,prof2[5],*prof3,*prof4[5];
    strcpy(prof1.name,"中村泰三");
    prof1.age=21;
    printf("%s  ",prof1.name);
    printf("%d 才\n\n",prof1.age);
    for(i=0;i<=4;i++){
        strcpy(prof2[i].name,"中村泰三");
        prof2[i].age=21;
        printf("%s  ",prof2[i].name);
        printf("%d 才\n",prof2[i].age);
    }
    printf("\n");
```

```
    prof3=&prof1;
    printf("%s  ",prof3->name);
    printf("%d 才\n\n",prof3->age);
    for(i=0;i<=4;i++){
        prof4[i]=&prof2[i];
        printf("%s  ",prof4[i]->name);
        printf("%d 才\n",prof4[i]->age);
    }
    printf("\n");
}
```

例 11-2　構造体データの初期化１

```
#include <stdio.h>
main()
{
    int i;
    struct profile {
        char name[21];
        int age;
    };
    struct profile prof1={"関優勝",40};
    struct profile prof2[2]={{"売　二",18},{"常荷金作",20}};
    printf("%s  ",prof1.name);
    printf("%d 才\n",prof1.age);
    for(i=0;i<=1;i++){
        printf("%s  ",prof2[i].name);
        printf("%d 才\n",prof2[i].age);
    }
}
```

例 11-3　構造体データの初期化２

```
#include <stdio.h>
main(){
    int i;
```

```
    struct shain {
        char *name;
        int age,nenshu;
    };
    struct shain meibo[2]={{"福井正康",41,650},
                           {"浅田精吾,40,1200}};
    struct shain *p;
    p=meibo;
    for(i=0;i<=1;i++){
        printf("%s  %d  %d\n",p->name,p->age,p->nenshu);
        p++;
    }
}
```

11.5 共用体

変数の型が指定されるとそれ以外の型はその変数（メモリ領域）の中に入れられませんでしたが、共用体変数を用いると様々な型のデータが同一の変数（メモリ領域）の中に入れられるようになります。

例えば数字か文字列かどちらかのデータの場合など、共用体を使うとメモリの無駄がなくなります。

11.6 共用体の宣言

共用体宣言の書式

```
union 共用体名（タグ名） {
    メンバ型指定;
    ・・・・・・
} [共用体変数名];
```

共用体のメモリ確保は最長のメンバ型に合わせられる。

共用体変数名は分離して後から

union タグ名 共用体変数名;　で定義出来る。

union タグ名 共用体配列名;　とすることも出来る。

union タグ名 共用体ポインタ; とすることも出来る。

union タグ名 共用体ポインタ配列; も出来る。

共用体データの宣言例

```
union sdata {
    char str[6];
    float no;
};
union sdata data1;
union sdata data2[100];
union sdata *data3;
union sdata *data4[100];
```

11.7 共用体データの参照

共用体データの参照例

```
x=data1.no;
strcpy(data,data1.str);
/* 共用体変数.メンバ変数 */
```

データは新しく代入すると前のデータは壊れます。

例 11-4　共用体の内容表示

```
#include <stdio.h>
#include <string.h>
main()
{
    union srep{
        char str[4];
        unsigned long a;
    }z;
    strcpy(z.str,"ABC");
    printf("z.str=%s\n",z.str);
    printf("z.a  =%08lxH\n",z.a);
}
```

例 11-5　構造体と共用体の併用

```
#include <stdio.h>
main()
```

```
{
    struct crep{
        unsigned char a,b;
    };
    union irep{
        int x;
        struct crep y;
    }z;
    z.x=0x1234;
    printf("z.x=%xH\n",z.x);
    printf("z.y.a=%xH z.y.b=%xH\n",z.y.a,z.y.b);
}
```

11.8 ビットフィールド

例 11-6　フラグデータの保存

```
#include <stdio.h>
main()
{
    int n;
    struct bitset{
        unsigned int ans1  : 1;
        unsigned int ans2  : 1;
        unsigned int ans3  : 1;
        unsigned int ans4  : 1;
        unsigned int       : 4;
    };
    union ansdata{
        struct bitset x;
        unsigned char y;
    }data;

    printf("以下の質問に Yes(1)/No(0) で答えて下さい。\n");
```

```
    printf("田舎が好き [1/0] > ");
    scanf("%d",&n);
    data.x.ans1=n;
    printf("体を動かすのが好き [1/0] > ");
    scanf("%d",&n);
    data.x.ans2=n;
    printf("動物が好き [1/0] > ");
    scanf("%d",&n);
    data.x.ans3=n;
    printf("体力に自信あり [1/0] > ");
    scanf("%d",&n);
    data.x.ans4=n;
    printf("\n あなたのデータは以下の形で貯えられます。\n");
    printf("DATA = %u\n",data.y);
}
```

１２章　プリプロセッサ

12.1 プリプロセッサとは

コンパイルに先立ってプログラムを書き換える操作を指示する部分をプリプロセッサと言います。例えば #include <ファイル名> はコンパイルの前に <ファイル名> のファイルを #include を書いた位置に読み込みます。また、#define EOF -1 は EOF の文字列が現れたらそれを -1 に置き換えます。

12.2 プリプロセッサの例

#include

ファイルを読み込む

ファイルを標準ディレクトリから読み込む

標準ディレクトリ：Turbo C の場合は tc\include\

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

ファイルをカレントディレクトリから読み込む

```
#include "graphics.h"   #include "data.h"
```

ファイルを指定ドライブ、指定ディレクトリから読み込む

```
#include "b:\data\graphics.h"
```

#define

マクロの定義

```
#define マクロ名 [マクロ定義]
#define PC98
#define EOF -1    #define RED printf("\x1b[31m")
```

引数を持つマクロの定義

```
#define マクロ名(引数並び) 引数を含むマクロ定義
#define abs(x) (x>=0 ? x:-x)
#define c_jump(x,y) printf("\x1b[%d;%dH",y+1,x+1); /*左上は(0,0)*/
```

#undef

マクロ定義の除去

```
#undef PC98
#undef EOF
```

#if ～ #elif ～ #else ～ #endif

条件付きコンパイル

```
#define DSIZE 100
#if DSIZE<=100
    char page[10];
#elif DSIZE<=200
    char page[20];
#else
    char page[30]
#endif
```

#if defined

条件付きコンパイル

#if defined 識別子 ～ #endif

#if !defined 識別子 ～ #endif　　/* if not defined */

#if defined 識別子 ～ #elif defined 識別子 ～ #else ～ #endif

識別子で与えられるマクロが定義されているかどうかで ～ の部分を
コンパイルするかどうか決める。

```
#define PC98
#if defined PC98
・・・・・・・・
#else
＊＊＊＊＊＊＊＊
#endif
```

PC98 が定義(#define)されていれば・・・の部分がコンパイルされ、
定義されていなければ＊＊＊の部分がコンパイルされる。

ＰＣ９８とＦＭＲ等のようにと同時に２つの機種で利用出来るソース等を
作る場合に有効。

例 12-1　条件付きコンパイル

```
#include <stdio.h>
#define DOS
#define PI 3.14159
```

```
#if defined DOS
    #define RED printf("\x1b[31m")
    #define YEL printf("\x1b[33m")
    #define NOR printf("\x1b[0;1m")
#else
    #define RED
    #define YEL
    #define NOR
#end if
#define c_jump(x,y) printf("\x1b[%d;%dH",(y)+1,(x)+1)

main()
{
    float r=5.0;
    c_jump(20,10);
    RED; printf("半径 %.1f の円の面積 = ",r);
    YEL; printf("%.3f\n",PI*r*r);
    NOR;
}
```

１３章　ファイル入出力

13.1 高水準ファイル入出力関数

関数　FILE *fopen(char *filename, char *mode)

#include <stdio.h>
機能　ファイルの mode によるオープン
戻値　成功：新たなストリームへのポインタ
　　失敗：NULL
例　　FILE *fp;　fp=fopen("test.dat","r");
備考　mode　　"r":読み出し専用(R)　"w":書き込み専用(W/C)
　　"a":追加(C)　"r+":既存ファイルの更新(R/W)
　　"w+":新規ファイルの更新(R/W/C)
　　"a+":追加・更新・ﾎﾟｲﾝﾀは文末(R/W/C)
　　ｵﾌﾟｼｮﾝ　"t":ﾃｷｽﾄﾌｧｲﾙ　"b":ﾊﾞｲﾅﾘﾌｧｲﾙ　使用法 "wt","rt+" etc.

関数　int fclose(FILE *fp)

#include <stdio.h>
機能　ファイルのクローズ
戻値　成功：0　失敗：-1

関数　int fprintf(FILE *stream, char *str, 引数１, 引き数２, ･･･)

#include <stdio.h>
機能　stream への出力
例　　fprintf(fp,"データの出力\n");
備考　標準のストリーム　stdin,stdout,stderr,stdaux,stdprn(プリンタ)

関数　int fscanf(FILE *stream, char *str, 引数１, 引数２, ･･･)

#include <stdio.h>
機能　stream からの入力
戻値　成功:フィールド数　フィールドなし:0　ファイルの終り:EOF

関数　int fseek(FILE *stream, long offset, int mode)

#include <stdio.h>
機能　ファイルの定まった位置(offset)へのポインタの移動
戻値　成功:0　失敗:0 以外
備考　mode SEEK_SET (0):ファイルの始めから
　　SEEK_CUR (1):現在位置から
　　SEEK_END (2):ファイルの終りから

関数　int putc(int c, FILE *stream)

#include <stdio.h>

機能　stream からの１文字入力

関数　int getc(FILE *stream)

#include <stdio.h>
機能　stream から１文字読んでアスキーコードを返す
戻値　成功:アスキーコード　ファイルの終り:EOF

関数　int ungetc(int c, FILE *stream)

#include <stdio.h>
機能　stream への１文字のプッシュバック

関数　size_t fread(void *buf,size_t size,size_t n,FILE *fp)

#include <stdio.h>
機能　ファイルから size バイトのデータを n 個、バッファ buf に読み込む
戻値　正常に読み込まれたブロック数

関数　size_t fwrite(void *buf,size_t size,size_t n,FILE *fp)

#include <stdio.h>
機能　バッファ buf からファイルに size バイトのデータを n 個書き込む
戻値　正常に書き込まれたブロック数

13.2 ファイル入出力

例 13-1　TYPE 命令

```
/* ファイル名は ty.c */
#include <stdio.h>
main()
{
    int c;
    char st[20];
    FILE *fp;
    printf("ファイル名 >> ");
    scanf("%s",st);
    fp=fopen(st,"r");
    if(fp==NULL){
        printf("ファイルが見つかりません。\n");
        exit(1);   /*失敗*/
    }
    while((c=getc(fp))!=EOF) putchar(c);
```

```
    fclose(fp);
    exit(0);   /*成功*/
}
```

例 13-2　ファイルからの数値の読み込み

```
#include <stdio.h>
main()
{
    char fname[31];
    int n=0;
    float x,total=0,mean;
    FILE *fp;
    printf("file name >> ");
    scanf("%s",fname);
    fp=fopen(fname,"rt");
    if(fp==NULL){
        printf("file not found\n");
        return;
    }
    while(fscanf(fp,"%f",&x)!=-1){
        n++;
        total=total+x;
    }
    fclose(fp);
    mean=total/(float)n;
    printf("total=%f\n",total);
    printf("mean =%f\n", mean);
}
```

例 13-3　テキストファイルにおけるランダムアクセス

```
#include <stdio.h>
#include <stdlib.h>
main()
{
```

```
    char fname[21];
    int i,n,dat;
    FILE *ff;
/* データの保存 */
    printf("何個のデータを発生させますか？ >> ");
    scanf("%d",&n);
    printf("保存するファイル名 >> ");
    scanf("%s",fname);
    fp=fopen(fname,"w+t");
    if(fp==NULL){
        printf("ファイルへの書き込みが出来ません。\n");
        exit(1);
    }
    for(i=0;i<n;i++){
        dat=i+1;
        fprintf(fp,"%5d",dat);
        printf("%5d",dat);
    }
    printf("\n");
/* データの読み出し */
    printf("奇数番目のデータのみ表示\n");
    for(i=0;i<n;i=i+2){
        fseek(fp,(long)i*5,SEEK_SET);
        fscanf(fp,"%5d",&dat);
        printf("%5d",dat);
    }
    printf("\n");
    fclose(fp);
    exit(0);
}
```

例 13-4　ファイルへのブロックライト

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

```
main()
{
    char fname[]="file15.dat";
    int i,a[100],b[100];
    FILE *fp;
    for(i=0;i<100;i++) a[i]=i+1;
    fp=fopen(fname,"wb");
    if(fp==NULL){
        printf("ファイルがオープン出来ません。\n");
        return;
    }
    fwrite((int *)a,sizeof(int),100,fp);
    fclose(fp);
    fp=fopen(fname,"rb");
    if(fp==NULL){
        printf("ファイルがオープン出来ません。\n");
        return;
    }
    fread((int *)b,sizeof(int),100,fp);
    fclose(fp);
    for(i=0;i<100;i++) printf("%4d",b[i]);
    printf("\n");
}
```

１４章　再帰呼出し

14.1 再帰呼出しの例

例 14-1 factorial

```
#include <stdio.h>
double fact(int);
main()
{
    int n;
    double p;
    printf("n! の計算 n >> ");
    scanf("%d",&n);
    p=fact(n);
    printf("n! = %.0f\n",p);
}
double fact(int n)
{
    double x;
    if(n==0) return 1;
    x=(double)n*fact(n-1);
    return x;
}
```

例 14-2　数列　$a_n=2a_{n-1}+3$　$a_1=1$

```
#include <stdio.h>
double a(int);
main()
{
    int n;
    double x;
    printf("n >> ");
    scanf("%d",&n);
```

```
    x=a(n);
    printf("an = %.0f\n",x);
}
double a(int n)
{
    double x;
    if(n==1) return 1;
    x=2*a(n-1)+3;
    return x;
}
```

例 14-3　ハノイの塔

```
#include <stdio.h>
void hanoi(int,char,char,char);
main()
{
    int n;
    printf("A の円盤を C を経由して B に移します。\n");
    printf("円盤の数 >> ");
    scanf("%d",&n);
    hanoi(n,'A','B','C');
}
void hanoi(int n,char a,char b,char c)
{
    if(n==1) printf("円盤 %d を %c から %c へ移す\n",n,a,b);
    else{
        hanoi(n-1,a,c,b);
        printf("円盤 %d を %c から %c へ移す\n",n,a,b);
        hanoi(n-1,c,b,a);
    }
}
```

１５章　メモリ管理（メモリの動的確保）

15.1 メモリの動的確保

始めからデータの個数が分かっていない場合、データを記憶するメモリの大きさをあらかじめ決めておくことは出来ません。取り敢えず十分な大きさのメモリを確保しておくことは出来ますが、小さなデータの場合はメモリの無駄使いになりますし、大きなデータではオーバーフローするかも知れません。そこで、メモリを必要なときに必要な量だけ切り取って利用出来る機能がＣには含まれています。その代表的な関数が malloc と free 関数です。

15.2 代表的関数

関数　void *malloc(size_t size)

#include <stdlib.h> or #include <alloc.h>
機能　メモリの動的確保
戻値　確保された領域の先頭ポインタ　どんな型のポインタでも可能
　　メモリ確保に失敗したら NULL
例　　float *block; block=(float *)malloc(sizeof(float)*2000);

関数　void free(void *block)

#include <stdlib.h> or #include <alloc.h>
機能　確保されたメモリの解放　*block の型は何でもよい
戻値　なし
例　　free(block);

15.3 メモリ管理の例文

例 15-1　int 型メモリ確保

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i,n,*a;
    printf("使用する配列の個数の入力\n");
    printf("  個数(n) >> ");
```

```
    scanf("%d",&n);
    a=(int *)malloc(sizeof(int)*n);
    for(i=0;i<n;i++) a[i]=random(100)+1;
    for(i=0;i<n;i++) printf("%4d",a[i]);
    free(a);
}
```

例 15-2　構造体メモリ確保

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i,n;
    struct profile {
        char name[21];
        int age;
        char tel[18];
    } *rec;
    printf("使用するデータの個数\n");
    printf("  個数(n) >> ");
    scanf("%d",&n);
    rec=(struct profile *)malloc(sizeof(struct profile)*n);
    /* 初期化 */
    for(i=0;i<n;i++){
        *(rec[i].name)=0x0;
        *(rec[i].tel)=0x0;
        rec[i].age=0;
        printf("%d ",i);
    }
    free(rec);
}
```

例 15-3　メモリ確保による２次元配列

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

```
#include <stdlib.h>
void setno(int **,int,int);
main()
{
    int i,j;
    int *a[10];
    for(i=0;i<10;i++) a[i]=malloc(5*sizeof(int *));
    setno(a,10,5);
    for(i=0;i<10;i++){
        for(j=0;j<5;j++) printf("%5d",a[i][j]);
        printf("\n");
    }
}
void setno(int **b,int m,int n)
{
    int i,j;
    for(i=0;i<10;i++)
        for(j=0;j<5;j++) b[i][j]=i*10+j;
}
```

例 15-4　リスト型データ構造

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

struct list {
    int data;
    struct list *next;
} *first;

main()
{
    makelist();
    replist();
```

```
    printf("\n");
    dellist(50);
    replist();
    freelist();
}

void makelist(void)
{
    int i;
    struct list *p,*q;
    q=NULL;
    for(i=0;i<100;i++){
        p=(struct list *)malloc(sizeof(struct list));
        p->data=100-i;
        if(i==0) p->next=NULL;
        else p->next=q;
        q=p;
    }
    first=p;
}
void replist(void)
{
    struct list *p;
    p=first;
    while(p!=NULL){
        printf("%4d",p->data);
        p=p->next;
    }
    printf("\n");
}
void freelist(void)
{
```

```
    struct list *p,*q;
    p=first;
    while(p!=NULL){
        q=p->next;
        free(p);
        p=q;
    }
}
void dellist(int n)
{
    int i;
    struct list *p,*q,*r;
    p=first;
    for(i=1;i<=n-2;i++) p=p->next;
    q=p->next;
    r=q->next;
    free(q);
    p->next=r;
}
```

１６章　割り込み

16.1 セグメントの概念

16.2 レジスタ構成

16.3 システムコールとは

16.4 システムコールの例文

例　キーボードバッファのクリア

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
void keycheck(void)
{
    union REGS inregs,outregs;
    inregs.h.ah=0x0b;
    intdos(&inregs,&outregs); /* int86(0x21,&inregs,&outregs); */
    if(outregs.h.al==0) return 0;
    else return 1;
}
void keycls(void)
{
    union REGS inregs,outregs;
    inregs.h.ah=0x0c;
    inregs.h.al=0x00;
    intdos(&inregs,&outregs); /* int86(0x21,&inregs,&outregs); */
}
void keyclear(void)
{
    int chk;
    chk=keycheck();
    if(chk==1) keycls();
```

```
}
main()
{
    int c;
    while(1){
        keyclear();
        c=getch();
        if(c==0x1b) break;
        printf("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n");
    }
}
```

16.5 インクルードファイル DOS.H 中のレジスタ構造体

この構造体の要素に値を代入して intdos(), int86(), int86x() 関数により割込みを発生させる。結果は同じくこの形の構造体の中に返ってくる。

```
構造体  WORDREGS
struct WORDREGS {
    unsigned int    ax, bx, cx, dx, si, di, cflag, flags;
};
構造体  BYTEREGS
struct BYTEREGS {
    unsigned char   al, ah, bl, bh, cl, ch, dl, dh;
};
共用体  REGS
union   REGS    {
    struct  WORDREGS x;
    struct  BYTEREGS h;
};
構造体  SREGS
struct  SREGS   {
    unsigned int    es;
    unsigned int    cs;
```

```
    unsigned int    ss;
    unsigned int    ds;
};
```

注）int86x で使用する