演算子（復習を兼ねて）

1. 算術演算子

+	加算	
-	減算	
*	掛算	
/	割算	例） 5.0/3.0 ---> 1.666667
%	剰余算	例） 5/3 ---> 1　　5%3 ---> 2

優先順位：　* / % のほうが、+ - より先に計算される。

例）

```
a=200+15*4;          a は 260 になる
a=200-15%6;          a は 197 になる
```

演算処理の順位を切り替えるには、( )を用いる。

```
a=(200+15)*4;        a は 860 になる
a=(200-15)%6;        a は  5 になる
```

2. 比較と論理演算

2.1 関係演算子（大小比較）

<	より小さい	
>	より大きい	
<=	以下	
>=	以上	
==	等しい	使用例）if(a==0)
!=	等しくない	使用例）if(a!=0)

2.2 論理演算子（論理演算）

&&	論理積（AND）	使用例）if(a==b && c>d)
||	論理和（OR）	使用例）if(a==b || c>=d)
!	否定（NOT）	使用例）if(!a)

優先順位：　! && || の順

例）

```
if(a==0 && b==0 || c==0) …… 「a が 0 かつ b が 0」または「c が 0」ならば、の意味
if(a==0 || b==0 && c==0) …… 「a が 0」または「b が 0 かつ c が 0」ならば、の意味
```

演算処理の順位を切り替えるには、( )を用いる。

```
if((a==0 || b==0) && c==0) …… 「a が 0 または b が 0」かつ「c が 0」ならば、の意味
```

またプログラムをわかりやすくするため、あえて( )を用いて

```
if((a==0 && b==0) || c==0) …… 「a が 0 かつ b が 0」または「c が 0」ならば、の意味
```

とすることがある（この場合は( )を取り去っても同じ処理をする）。

2.3 「真」と「偽」

条件判断では、「条件が真」「条件が偽」という表現をするが、これは条件部に書かれた式が満足されているか否かを表す。例えば

```
if(a<10)
```

と書く場合、条件式 a<10 を満たすとき、「真」となる。条件式を満たさないとき（a>=0 のとき)、「偽」となる。

条件式を満たすか否かを整数型の論理値で表し、0 を「偽」、0 以外を「真」とみなす。(関係演算・論理演算)
その結果、省略コーディングが可能となる。
例えば、if(a != 0) という条件判断は、if(a)と記述することができ、「a が真なら」と読みくだす。
また、 if(a == 0) という条件判断は、if(!a)と記述することができ、「a が偽なら」と読みくだす。

例）関係演算・論理演算の結果の論理値

```
#include <stdio.h>

void main(void)
{
    int a, b, c, d, bool;

    printf("Compare a, b ? ");
    scanf("%d %d", &a, &b);
    printf("a<b  ---> %d\n", a<b);          /* 関係・論理演算の結果は、真ならば 1、偽ならば 0 */

    printf("Compare c, d ? ");
    scanf("%d %d", &c, &d);
    printf("c<d  ---> %d\n", c<d);                          /* 上記のとおり */

    bool=a<b&&c<d;              /* 関係・論理演算の結果は整数型の変数に代入できるのです!!! */
    printf("a<b&&c<d  ---> %d\n", bool);
    bool=a<b||c<d;                                          /* 上記のとおり!!! */
    printf("a<b||c<d  ---> %d\n", bool);
}
```

```
------------------------実行開始----------------------
Compare a, b ? 0 3
a<b  ---> 1
Compare c, d ? 5 2
c<d  ---> 0
a<b&&c<d  ---> 0
a<b||c<d  ---> 1
------------------------おしまい----------------------
```

関係演算・論理演算の結果は整数型の数値（論理値）で表される。
C言語システムからは、真であれば 1、偽であれば 0 の値が設定される。

3. インクリメント・デクリメント演算子

++　インクリメント演算子（1 を加算する）
--　デクリメント演算子　（1 を減算する）

　++式、--式　　前置型：加算（減算）の後、その結果が次の演算に使われる。
　式++、式--　　後置型：現在の値が演算に使われた後、加算（減算）される。

例）前置型と後置型

```
#include <stdio.h>

void main(void)
{
    int i, a, b, dt[5]={10,20,30,40,50};
                                    /* 後置型の例 */
    i=0;
    ++i;                            /* i に 1 加算する。 i は 1  */
    a=++i;                          /* i に 1 加算してから i の値を a に代入。i は 2、a は 2  */
    b=dt[++i];                      /* i=i+1; b=dt[i]; と同じ。 i は 3、b は 40  */
    printf("i=%d a=%d b=%d\n",i,a,b);
                                    /* 前置型の例 */
    i=0;
    i++;                            /* i に 1 加算する。 i は 1  */
    a=i++;                          /* i の値を a に代入してから i に 1 加算。i は 2､a は 1  */
    b=dt[i++];                      /* b=dt[i]; i=i+1  と同じ。 i は 3、b は 30  */
    printf("i=%d a=%d b=%d\n",i,a,b);
}
```

```
-------------------------実行開始-----------------------
i=3 a=2 b=40
i=3 a=1 b=30
-------------------------おしまい-----------------------
```

4. アドレス

アドレスとは、変数が蓄えられているメモリー上の場所のことである。

アドレス演算子　&

　&変数名　　と書けばその変数が割り付けられているアドレスを求める。
　　求められたアドレス値はポインタ変数に代入して格納できる。

　ポインタ変数の宣言例）
　　int *point1;　char *point2;　float *point3;

間接演算子 *

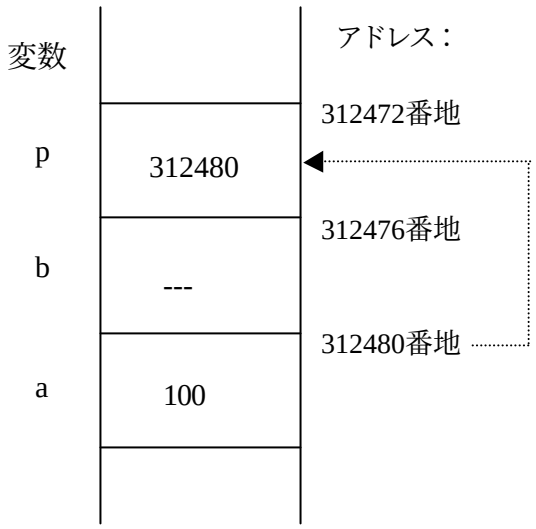
　*ポインタ変数名　と書けばポインタの示すアドレスに格納されているデータを（間接的に）表す。

例）アドレス演算子と間接演算子の働き

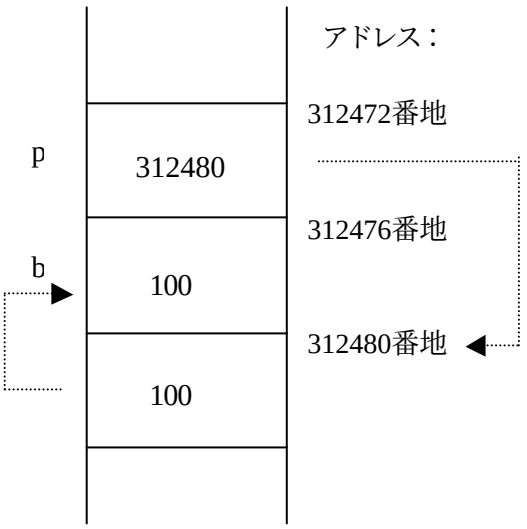
```
#include <stdio.h>
void main(void)
{
     int a=100, b;    int *p;    /* 型宣言により、変数を蓄える領域がメモリ上に確保される */

     printf("a,b,p のアドレスは、10 進数表示で %d %d %d\n", &a,&b,&p);
     p=&a;
     printf("a=%d p=%d\n", a,p);
     b=*p;
     printf("a=%d b=%d\n", a,b);
}
-------------------------実行開始-----------------------
a,b,p のアドレスは、10 進数表示で 312480 312476 312472
a=100 p=312480
a=100 b=100
-------------------------おしまい-----------------------
```

p = &a ; の働き



b = *p ; の働き



5. 代入演算子

=	n1 = n2		
+=	n1 += n2	は	n1 = n1+n2 と同じ
-=	n1 -= n2	は	n1 = n1-n2 と同じ
*=	n1 *= n2	は	n1 = n1*n2 と同じ
/=	n1 /= n2	は	n1 = n1/n2 と同じ
%=	n1 %= n2	は	n1 = n1%n2 と同じ

6. cast 演算子：データの型変換を明示的に行う。

例） int a; float b; char *d;
```
     b = (float) a;
     a = (int) b;
     d = (char *) a;
```