データ型　改訂版

1. 基本データ型

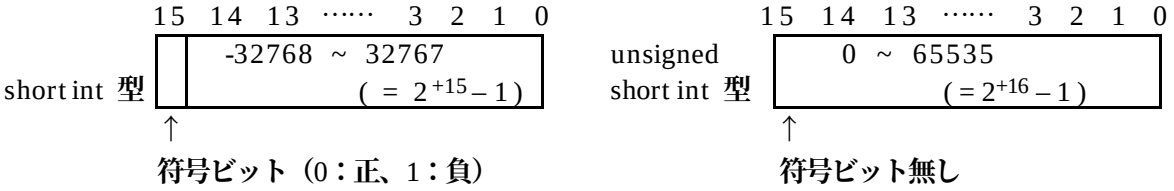
基本となるデータ型

・char　　：文字型　　（1バイト）　……　文字を表現するデータ型
・int　　：整数型　　（2バイトまたは 4バイト；C コンパイラによって異なる）
　　＊ CLEOS システムでは int 型は4バイト
・short int　：整数型　　（2バイト）
・long int　：倍長整数型（4バイト）
・float　　：実数型、　浮動小数点型とも呼ぶ　　（4バイト）
・double　：倍長実数型、倍精度浮動小数点型とも呼ぶ（8バイト）
・void　　：型を持たない型

int 型（short int と long int を含む）と char 型は、通常符号有りであるが、符号無しの指定ができる。

・unsigned int　：符号無し int 型
・unsigned char　：符号無し char 型

unsigned 型にすると、負の値は表現できないが、正の値の表現範囲が 2 倍になる。



符号ビット（0：正、1：負）　　符号ビット無し

int 型に short, long, unsigned などの修飾子が付くときは、int を省略できる。

例）short int　⟶　short
　　unsigned int　⟶　unsigned
　　unsigned long int　⟶　unsigned long

変数の表現範囲

＊ 注：1バイト＝8ビット

データ型	バイト幅 *	表 現 範 囲	
char	1	-128 ~ 127	（127 = $2^{+7}-1$ ）
short int	2	-32768 ~ 32767	（32767 = $2^{+15}-1$ ）
long int	4	-2147483648 ~ 2147483647	（2147483647 = $2^{+31}-1$ ）
int	4 (CLEOS)	-2147483648 ~ 2147483647	
float	4	3.4×10^{-38} ~ $3.4\times10^{+38}$,	有効桁数　約 7桁
double	8	1.7×10^{-308} ~ $1.7\times10^{+308}$,	有効桁数　約 15 桁

例１）変数の型宣言と定数

```
char   c1, c2;
int    i0, i1, i2;
float  pi1, eps;
double pi2, inf;

c1='A';       /* 文字型変数c1に文字定数'A'を代入 */
c2=65;        /* 文字型変数c2に'A'の文字コード65を代入、上式と同じこと */
i0=9876;      /* 10進数の9876をi0に代入 */
i1=045;             /* 8進数の45（先頭に0がつくと8進数）をi1に代入 */
i2=0x2d9f;      /* 16進数の2d9f（先頭に0xがつくと16進数）をi2に代入 */
pi1=3.141592;             /* 実数型     の有効桁数は約7桁 */
pi2=3.141592653589793;    /* 倍長実数型の有効桁数は約15桁 */
eps=1.0e-7;               /* 実数型     の表現範囲は3.4×10^-38  ~  3.4×10^+38 */
inf=-1.0e+300;            /* 倍長実数型の表現範囲は1.7×10^-308  ~  1.7×10^+308 */
```

例２）castによる型変換

```
#include <stdio.h>

void main(void)
{
     int i8=8, i5=5;                    /* i8 と i5の値の初期設定 */

①    printf("(float)(i8/i5)      = %f\n", (float)(i8/i5) );
②    printf("(float)i8/(float)i5= %f\n", (float)i8/(float)i5 );
③    printf("(float)i8/i5        = %f\n", (float)i8/i5 );
④    printf("i8/(float)i5        = %f\n", i8/(float)i5 );
}

------------------------実行開始----------------------
(float)(i8/i5)      = 1.000000
(float)i8/(float)i5= 1.600000
(float)i8/i5        = 1.600000
i8/(float)i5        = 1.600000
------------------------おしまい----------------------
```

① 整数型変数i8とi5による整数演算 i8/i5の結果は1であり、これを実数型に変換して出力すると1.000000となる。
② 実数型への変換した数値 (float)i8、(float)i5による演算 (float)i8/(float)i5の結果は1.600000となる。
③④ i8かi5のいずれか一方を実数型に変換しておけば、順位の高い型（この場合実数型）で演算が行われる。

型の異なる２数の演算においては、上位の型に自動的に変換されて演算が行われる。その順位は以下のとおり。
　　double > float > long (int) > short > char

この例のように変数の前に (型名) を付加して型変換を明示的に行うことを castという。

2. 配列

配列とは、同種のデータ型を集めた変数であり、添え字により順序づけられている。

配列の型宣言においては、データ型と要素数を指定する。

```
  1次元配列の型宣言は     データ型   配列名[要素数];
  2次元配列の型宣言は     データ型   配列名[要素数][要素数];
  3次元配列の型宣言は     データ型   配列名[要素数][要素数][要素数];
        :                    :          :
例）int dt[10], i;                    /* 整数型1次元配列 dt と実数型変数 i の宣言 */
    float  a[2][4], c[2][4][5];   /* 実数型2次元配列 a、実数型3次元配列 c の宣言 */
```

配列のメモリ上の配置

例１）int　dt[10];

は、整数型のデータが 10 個からなる変数 dt の型宣言文である。このとき確保される配列変数は、添え字が 0 から 9 までの、dt[0] ～ dt[9] である。

注意：BASIC では、dt[0] ～ dt[10] までの 11 個が確保される。混同しないように！

int dt[10]; のメモリ上の配置

dt[0]
dt[1]
dt[2]
dt[3]
dt[4]
dt[5]
dt[6]
dt[7]
dt[8]
dt[9]

例２）float a[2][4]; は、配列長 4 の配列を 2 組用意した、ということになる。

float a[2][4]; の行列イメージ

a[0][0]	A[0][1]	a[0][2]	a[0][3]
a[1][0]	A[1][1]	a[1][2]	a[1][3]

float a[2][4]; のメモリ上の配置

行ごとに配置（右側の括弧[ ]内数字が先に変化するように順番づけられる）

a[0][0]	↑
a[0][1]	配列長 4 の配列
a[0][2]	
a[0][3]	↓
a[1][0]	↑
a[1][1]	配列長 4 の配列
a[1][2]	
a[1][3]	↓

注意：FORTRAN では、2 次元配列 a(2,4) は列ごとに配置される（括弧内の左側の数字が先に変化するように順番づけられる）。混同しないように！

3. 文字と文字列

・文字には対応する番号（文字コード）が定まっている。
　文字コードに関しては、本節後部に載せる ASCII コード表を参照のこと。

・文字列は char 型の配列により表現される。

　文字列の最後は文字コード 0　（終端コードあるいはヌルコードと呼ばれ '\0' と文字定数表現される）。
　例えば文字列 ”abcdef” はメモリ上に次のように置かれている。

‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	0

　文字列は終端コード 0 を必要とするため、
　　char str[20];
　により宣言される文字列 str は、19 文字までしか格納できない。

例１）

```
#include <stdio.h>

void main(void)
{
      char ch, str[20];

      printf("input a capital letter (大文字) ==> ");
①     scanf("%c",&ch);
      printf("文字：%c       文字コード：%d\n",ch,ch);            /* 文字と文字コードの出力 */
      ch=ch+32;          /* char型は何と演算もできるのです！　大文字コードに32を足すと小文字コード */
      printf("文字：%c       文字コード：%d\n",ch,ch);            /* 文字と文字コードの出力 */

      printf("input string of 6 characters ==> ");
②     scanf("%s",str);            /* 読み込んだ文字列の直後に自動的に文字コード0が付加される */
      printf("1.str: %s\n",str);

      str[0]='X';                          /* 文字列strの１番目の文字をXにする */
      str[1]='Y';                          /* 文字列strの２番目の文字をYにする */
      str[2]='Z';                          /* 文字列strの３番目の文字をZにする */
      printf("2.str: %s\n",str);
      str[3]='\0';            /* str[3]=0; としてもよい。文字列strの４番目の要素に終端コードを与える */
      printf("3.str: %s\n",str);
}
-------------------------実行開始-----------------------
input a capital letter (大文字) ==> F
文字：F       文字コード：70
文字：f       文字コード：102
input string of 6 characters ==> abcdef
1.str: abcdef
2.str: XYZdef
3.str: XYZ
-------------------------おしまい-----------------------
```

・scanf による文字と文字列の読み込みかたの相違に注意

①１文字の読み込み

scanf("%c", &変数名); …… 変数名の前に & をつける

②文字列の読み込み

scanf("%s", 文字配列名); …… 配列名の前に & をつけない

scanf で指定する変数名には、正確には変数のアドレスを書く。

&変数名

と書けば、その変数のアドレスを取り出すことができる。

要素指定のない配列名はその配列の先頭アドレスを表すので、& をつける必要はない。

例２）文字コードと文字の出力

```
void main(void)
{
     char i;                                    /* char型の変数の表現範囲は -128 ~ 127 */

     for (i=32; i<127; i++){
        printf("%d %c  ",i,i);
        if( i%10==0 ) printf("\n");
     }
     printf("\n");
}
```

```
-------------------------実行開始-----------------------
32    33 !  34 "  35 #  36 $  37 %  38 &  39 '  40 (
41 )  42 *  43 +  44 ,  45 -  46 .  47 /  48 0  49 1  50 2
51 3  52 4  53 5  54 6  55 7  56 8  57 9  58 :  59 ;  60 <
61 =  62 >  63 ?  64 @  65 A  66 B  67 C  68 D  69 E  70 F
71 G  72 H  73 I  74 J  75 K  76 L  77 M  78 N  79 O  80 P
81 Q  82 R  83 S  84 T  85 U  86 V  87 W  88 X  89 Y  90 Z
91 [  92 \  93 ]  94 ^  95 _  96 `  97 a  98 b  99 c  100 d
101 e  102 f  103 g  104 h  105 i  106 j  107 k  108 l  109 m  110 n
111 o  112 p  113 q  114 r  115 s  116 t  117 u  118 v  119 w  120 x
121 y  122 z  123 {  124 |  125 }  126 ~
-------------------------おしまい----------------------
```

例３）文字コード表を見やすく表示（ 拡張ASCII コード表を書く）

ASCII: American Standard Code for Information Interchange

```
#include <stdio.h>

void main(void)
{                    /* 例２においては0 ～ 127 のコード範囲を扱ったので、変数宣言は char i; */
     int i;               /* 本例では 0 ～ 255 のコード範囲を表示するので、*/
                          /*   変数宣言は int i; または unsigned char i;   */

     printf("Table for ASCII Code\n");

     printf("\n   |");
```

```
    for(i=0x0; i<=0xf; i++) printf("  %x",i);     /* 先頭に 0x がつく定数は16進数 */
    printf("\n---+");
    for(i=0x0; i<=0xf; i++) printf("---");

    for(i=0x0; i<0xff; i++){
      if( i%0x10==0 )
        printf("\n  %x|",i/0x10);
      if( (i>0x20 && i<0x7f)||(i>0xa0 && i<0xe0) )
        printf("  %c",i);
      else
        printf("   ");
    }
    printf("\n");
}
```

```
-------------------------実行開始-----------------------
Table for ASCII Code

   |  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
---+-------------------------------------------------
  0|
  1|
  2|     !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
  3|  0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
  4|  @  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
  5|  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
  6|  `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
  7|  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~
  8|
  9|
  a|     。 「 」 、 ・ ヲ ァ ィ ゥ ェ ォ ャ ュ ョ ッ
  b|  ー ア イ ウ エ オ カ キ ク ケ コ サ シ ス セ ソ
  c|  タ チ ツ テ ト ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ
  d|  ミ ム メ モ ヤ ユ ヨ ラ リ ル レ ロ ワ ン ゛ ゜
  e|
  f|
-------------------------おしまい-----------------------
```

・文字の表現方法

+) ASCII コードでは、アルファベット・数字は char 型 1 バイトで表され、その文字コードは上の表の 0 から 127（16 進数で 0 から 7f；表の上 8 段）までのとおり。
例えば、大文字の A の文字コードは、16 進数で 41、10 進数で 65。

+) ASCII コードの 0 から 31（16 進数で 0 から 1f；表の上 2 段）と 127（ 16 進数で 7f ）は制御コード。

+) 半角カナ文字は 符号なし char 型 1 バイト（0 から 255）で表され、その文字コードは上の表の 161 から 223（16 進数で a1 から df）までのとおり。
日本ではこれも含めて ASCII コードと呼ぶ場合がある。

+) 漢字・ひらがななどの全角文字は char 型 2 バイトで表され、先頭の第 1 バイトは上表のあいている部分のコードを用いる（シフト JIS コード）。

例４）おまけ：シフト JIS コード（漢字、ひらがな、カタカナなどの全角文字）から

```
#include <stdio.h>

void main(void)
{
    int i;
    unsigned char s[3]={0,0,0};     /* s[0]、s[1]は全角文字１個を表すため、s[2]=0 は終端コード */

    printf("おまけ１：シフト JIS コードから（ひらがななど）\n");
    for (i=0x00; i<=0xff; i++){                          /* 先頭に 0x がつく定数は16進数 */
        s[0]=0x82;          /* 16進数の82は例３の ASCII コード表であいてるコードでしょ！ */
        s[1]=i;             /* s[0]、s[1]の２バイトで全角文字１つを表す */
      if( 0x4e<i&&i<0x59 || 0x5f<i&&i<0x7a || 0x80<i&&i<0x9b || 0x9e<i&&i<0xf2 )
        printf("%2x%02x %s  ",s[0],s[1],s);
      else
        printf("%2x%02x      ",s[0],s[1]);
      if( (i+1)%8==0 ) printf("\n");              /* コード番号が８進む毎に改行する */
    }

    printf("\nおまけ２：シフト JIS コードから（カタカナなど）\n");
    for (i=0x00; i<=0xff; i++){
        s[0]=0x83;          /* 16進数の83も例３の ASCII コード表であいてるコードでしょ！ */
        s[1]=i;             /* s[0]、s[1]の２バイトで全角文字１つを表す */
      if( 0x3f<i&&i<0x7f || 0x7f<i&&i<0x97 || 0x9e<i&&i<0xb7 || 0xbe<i&&i<0xd7 )
        printf("%2x%02x %s  ",s[0],s[1],s);
      else
        printf("%2x%02x      ",s[0],s[1]);
      if( (i+1)%8==0 ) printf("\n");              /* コード番号が８進む毎に改行する */
    }
}
```

----------実行開始----------

おまけ１：シフト JIS コードから（ひらがななど）

```
8200      8201      8202      8203      8204      8205      8206      8207
8208      8209      820a      820b      820c      820d      820e      820f
8210      8211      8212      8213      8214      8215      8216      8217
8218      8219      821a      821b      821c      821d      821e      821f
8220      8221      8222      8223      8224      8225      8226      8227
8228      8229      822a      822b      822c      822d      822e      822f
8230      8231      8232      8233      8234      8235      8236      8237
8238      8239      823a      823b      823c      823d      823e      823f
8240      8241      8242      8243      8244      8245      8246      8247
8248      8249      824a      824b      824c      824d      824e      824f ０
8250 １   8251 ２   8252 ３   8253 ４   8254 ５   8255 ６   8256 ７   8257 ８
8258 ９   8259      825a      825b      825c      825d      825e      825f
8260 Ａ   8261 Ｂ   8262 Ｃ   8263 Ｄ   8264 Ｅ   8265 Ｆ   8266 Ｇ   8267 Ｈ
8268 Ｉ   8269 Ｊ   826a Ｋ   826b Ｌ   826c Ｍ   826d Ｎ   826e Ｏ   826f Ｐ
8270 Ｑ   8271 Ｒ   8272 Ｓ   8273 Ｔ   8274 Ｕ   8275 Ｖ   8276 Ｗ   8277 Ｘ
8278 Ｙ   8279 Ｚ   827a      827b      827c      827d      827e      827f
8280      8281 ａ   8282 ｂ   8283 ｃ   8284 ｄ   8285 ｅ   8286 ｆ   8287 ｇ
8288 ｈ   8289 ｉ   828a ｊ   828b ｋ   828c ｌ   828d ｍ   828e ｎ   828f ｏ
8290 ｐ   8291 ｑ   8292 ｒ   8293 ｓ   8294 ｔ   8295 ｕ   8296 ｖ   8297 ｗ
8298 ｘ   8299 ｙ   829a ｚ   829b      829c      829d      829e      829f ぁ
```

82a0 あ	82a1 ぃ	82a2 い	82a3 ぅ	82a4 う	82a5 ぇ	82a6 え	82a7 ぉ
82a8 お	82a9 か	82aa が	82ab き	82ac ぎ	82ad く	82ae ぐ	82af け
82b0 げ	82b1 こ	82b2 ご	82b3 さ	82b4 ざ	82b5 し	82b6 じ	82b7 す
82b8 ず	82b9 せ	82ba ぜ	82bb そ	82bc ぞ	82bd た	82be だ	82bf ち
82c0 ぢ	82c1 っ	82c2 つ	82c3 づ	82c4 て	82c5 で	82c6 と	82c7 ど
82c8 な	82c9 に	82ca ぬ	82cb ね	82cc の	82cd は	82ce ば	82cf ぱ
82d0 ひ	82d1 び	82d2 ぴ	82d3 ふ	82d4 ぶ	82d5 ぷ	82d6 へ	82d7 べ
82d8 ぺ	82d9 ほ	82da ぼ	82db ぽ	82dc ま	82dd み	82de む	82df め
82e0 も	82e1 ゃ	82e2 や	82e3 ゅ	82e4 ゆ	82e5 ょ	82e6 よ	82e7 ら
82e8 り	82e9 る	82ea れ	82eb ろ	82ec ゎ	82ed わ	82ee ゐ	82ef ゑ
82f0 を	82f1 ん	82f2	82f3	82f4	82f5	82f6	82f7
82f8	82f9	82fa	82fb	82fc	82fd	82fe	82ff

おまけ2：シフト JIS コードから（カタカナなど）

8300	8301	8302	8303	8304	8305	8306	8307
8308	8309	830a	830b	830c	830d	830e	830f
8310	8311	8312	8313	8314	8315	8316	8317
8318	8319	831a	831b	831c	831d	831e	831f
8320	8321	8322	8323	8324	8325	8326	8327
8328	8329	832a	832b	832c	832d	832e	832f
8330	8331	8332	8333	8334	8335	8336	8337
8338	8339	833a	833b	833c	833d	833e	833f
8340 ァ	8341 ア	8342 ィ	8343 イ	8344 ゥ	8345 ウ	8346 ェ	8347 エ
8348 ォ	8349 オ	834a カ	834b ガ	834c キ	834d ギ	834e ク	834f グ
8350 ケ	8351 ゲ	8352 コ	8353 ゴ	8354 サ	8355 ザ	8356 シ	8357 ジ
8358 ス	8359 ズ	835a セ	835b ゼ	835c ソ	835d ゾ	835e タ	835f ダ
8360 チ	8361 ヂ	8362 ッ	8363 ツ	8364 ヅ	8365 テ	8366 デ	8367 ト
8368 ド	8369 ナ	836a ニ	836b ヌ	836c ネ	836d ノ	836e ハ	836f バ
8370 パ	8371 ヒ	8372 ビ	8373 ピ	8374 フ	8375 ブ	8376 プ	8377 ヘ
8378 ベ	8379 ペ	837a ホ	837b ボ	837c ポ	837d マ	837e ミ	837f
8380 ム	8381 メ	8382 モ	8383 ャ	8384 ヤ	8385 ュ	8386 ユ	8387 ョ
8388 ヨ	8389 ラ	838a リ	838b ル	838c レ	838d ロ	838e ヮ	838f ワ
8390 ヰ	8391 ヱ	8392 ヲ	8393 ン	8394 ヴ	8395 ヵ	8396 ヶ	8397
8398	8399	839a	839b	839c	839d	839e	839f Α
83a0 Β	83a1 Γ	83a2 Δ	83a3 Ε	83a4 Ζ	83a5 Η	83a6 Θ	83a7 Ι
83a8 Κ	83a9 Λ	83aa Μ	83ab Ν	83ac Ξ	83ad Ο	83ae Π	83af Ρ
83b0 Σ	83b1 Τ	83b2 Υ	83b3 Φ	83b4 Χ	83b5 Ψ	83b6 Ω	83b7
83b8	83b9	83ba	83bb	83bc	83bd	83be	83bf α
83c0 β	83c1 γ	83c2 δ	83c3 ε	83c4 ζ	83c5 η	83c6 θ	83c7 ι
83c8 κ	83c9 λ	83ca μ	83cb ν	83cc ξ	83cd ο	83ce π	83cf ρ
83d0 σ	83d1 τ	83d2 υ	83d3 φ	83d4 χ	83d5 ψ	83d6 ω	83d7
83d8	83d9	83da	83db	83dc	83dd	83de	83df
83e0	83e1	83e2	83e3	83e4	83e5	83e6	83e7
83e8	83e9	83ea	83eb	83ec	83ed	83ee	83ef
83f0	83f1	83f2	83f3	83f4	83f5	83f6	83f7
83f8	83f9	83fa	83fb	83fc	83fd	83fe	83ff

----------おしまい----------