３回目　文字列の取扱い

Ｃ言語での文字列の扱い方

Windowsでの文字列

TCHAR型

mlibでの文字入出力関数

C言語での文字変数の一般的な取扱い

キーボードから入力される英数字、記号等の半角文字はASCIIコードと呼ばれる0～255（8bit、1バイト）までの番号が割り振られている。

char 型　→　8bit の整数型

C言語等で文字変数を扱うときには
char型が使われる。

```
char moji;
```
変数 moji をchar型（1バイト整数、8bit、0～255）で定義

変数moji は半角1文字を表現できる。ASCIIコードで表される整数値が入る

データの種類	printf関数での変換文字
1文字	%c

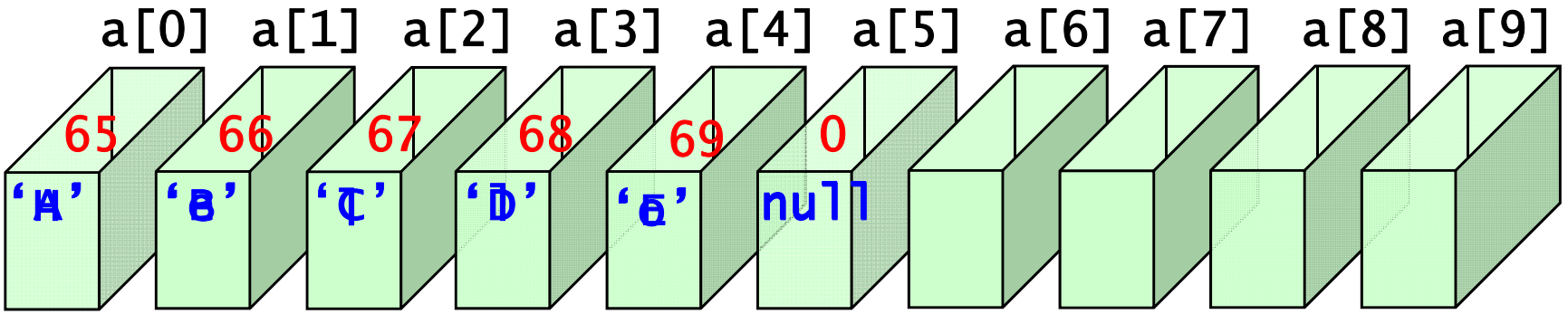
```
moji=65;
Printf("%c",moji);
```
A　ASCIIコード65に対応する文字’A’が出力

文字列の取扱い

複数の文字で表される文字列は変数でどう表すか　　char型の**配列変数**を使う

文字列の定義

```
char a[10];
```
変数aをchar型配列として10文字確保




```
char a[10]="ABCDE";
```
文字配列は宣言時に“　”でくくって初期化可能

注意）文字に対応する数値（8bit）が各配列要素に入る
　　最後に文字列の終わりをあらわすヌル文字（0）が必ず入る.

```
char a[]="Hello";
```
変数aをchar型配列として6文字（5文字+null1文字）定義

文字配列の出力

```
char a[10]="ABCDE";
```

データの種類	printf関数変換文字
文字列	%s

```
Printf("%s",a);
```

```
ABCDE
```

配列変数名のみの記述では、その配列変数が確保されたメモリの先頭アドレス番号（正数値）を示す.(Cプログラム内で a は&a[0]と同じ)

変換文字%sはアドレス&a[0]から順番にnull が見つかるまで数値をASCIIコードとして表示し続ける

```
Printf("%s",&a[0]);
```

```
ABCDE
```

```
Printf("%s",&a[2]);
```

```
CDE
```

```
sprintf(b,"%s FG",a);
```

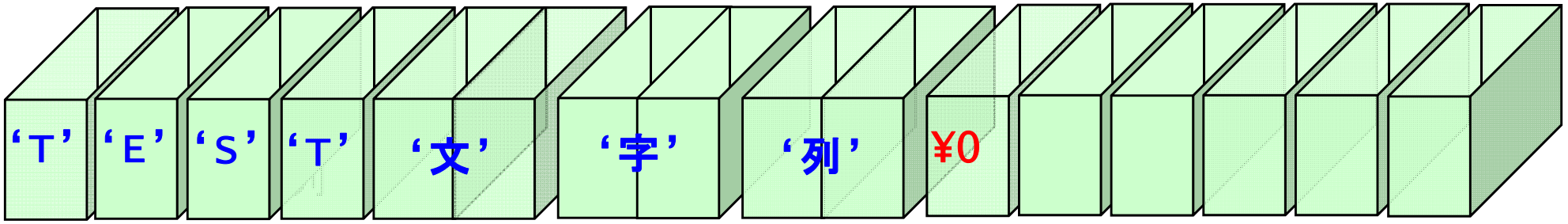
b ← "ABCDE FG" が代入される

Visual Cでの文字列表現

Windowsでは互換性の問題で複数の文字コードが混在している

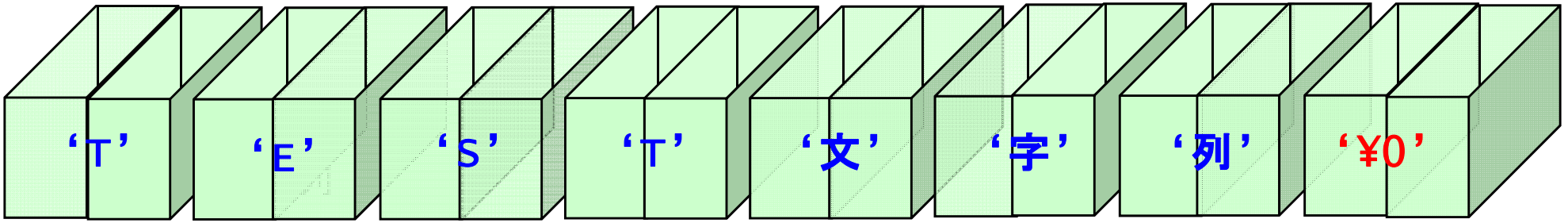
マルチバイト文字列(char型)

主に英字を1バイト（8ビット）、日本語文字を2バイト（16ビット）で表現する。
日本語Windows上で現在我々の利用する文字コードであるShift-JISに準拠



ワイド文字列(wchar_t型, LPTSTR型, LPCTSTR型)

英字・日本語文字・その他言語文字の種別にかかわらず、原則すべて
1字2バイトで表現。世界標準文字コードでUnicodeを採用



ワイド文字列の使用法

マルチバイト文字列->今までと同じ

```
char a[]="test文字列";
printf("マルチバイト文字 %s",a);
```

```
マルチバイト文字 test文字列
```

取扱いはlinuxでやってきたやり方と一緒、特別なことは考えない

ワイド文字列(UNICIDE)の利用

```
wchar_t b[]=L"test文字列";
wprintf(L"ワイド文字 %s",b);
```

```
ワイド文字 test文字列
```

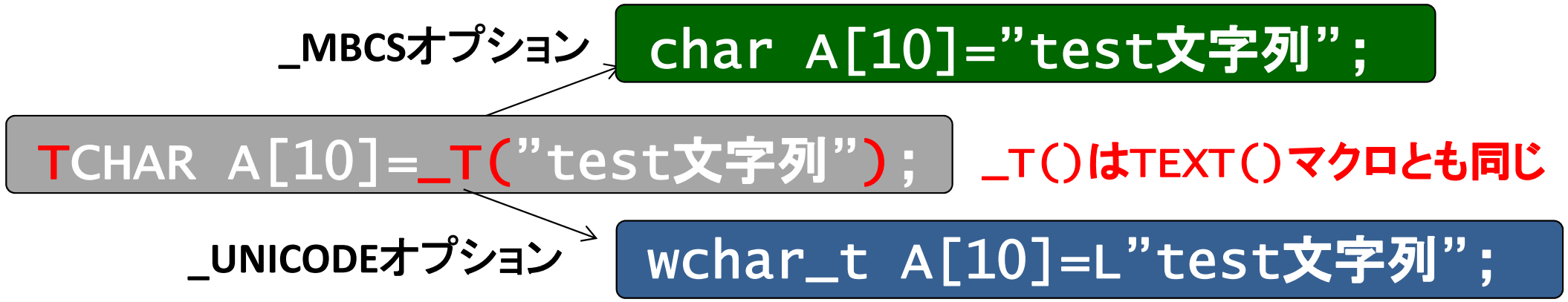
ワイド文字列を扱う場合、文字列定数(""で囲まれた文字列)の前にはLプレフィクスを付ける。これを付けないと文字列がマルチバイト文字列にみなされる。

ワイド文字を扱う関数は先頭にwが追加される等、関数名も異なるマルチバイト文字関数を使うと文字化けするので注意

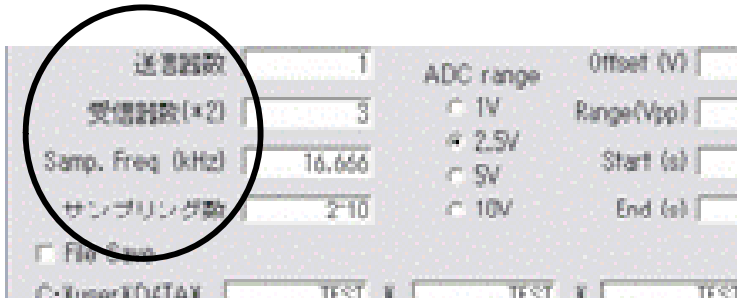
TCHAR型

マルチバイト文字列か、ワイド文字列のどちらを使うかをコンパイル時に指定すれば、自動的に変換してくれる型。この型で書いていれば1種類のプログラムで済む

mlibはUNICODE文字を前提につくられている。def.h内の文字列はTCHAR型
コンパイラは標準でTCHAR型をワイド文字（ wchar_t ）型に設定。



スタティック



・文字のみを表示する。主に、エディットボックスの説明用に対になって利用する

・def.h内の定数ST_NUMで指定した個数のスタティックを配置可

・スタティック番号 i は 0 から ST_NUM-1 まで

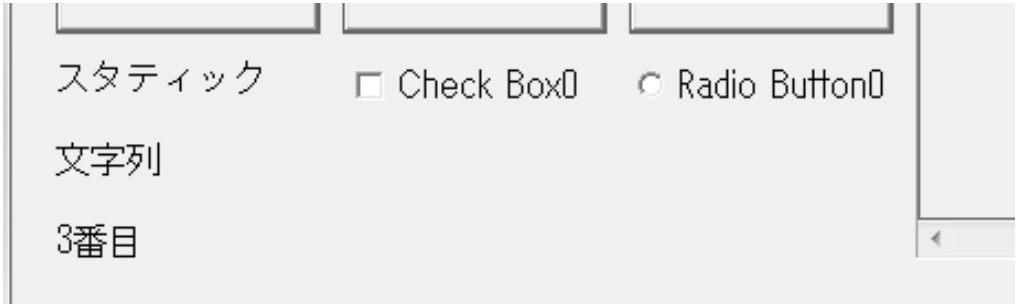
・位置、サイズ、キャプションはsST[スタティック番号]構造体を変更すればよい。

i番目のスタティックの設定	sST[i].x	スタティック左上のx座標
	sST[i].y	スタティック左上のy座標
	sST[i].w	スタティックの幅
	sST[i].h	スタティックの高さ
	sST[i].name	スタティックの文字列(TCHAR型)

スタティック、ボタン、チェックボックス、ラジオボタンのnameプロパティはTCHAR型で表記されるので注意

演習3-1

３－１．スタティックを3つ作成し、そのテキストに“スタティック”、“文字列”、“3番目”の名前を付ける



ヒント　sST[0].name　等はdef.h、component()関数内で最初から指定しないとダメ、メインプログラム中でこれらの変数を変えても更新されない．

ヒント　TCHAR型文字列(wchar_t型)に文字列を代入するには以下を使う

```
lstrcpy(TCHAR型変数先頭アドレス,TCHAR型文字列);
wsprintf(TCHAR型変数先頭アドレス,TCHAR型文字列, 変数等);
```

文字列は代入文で代入できない

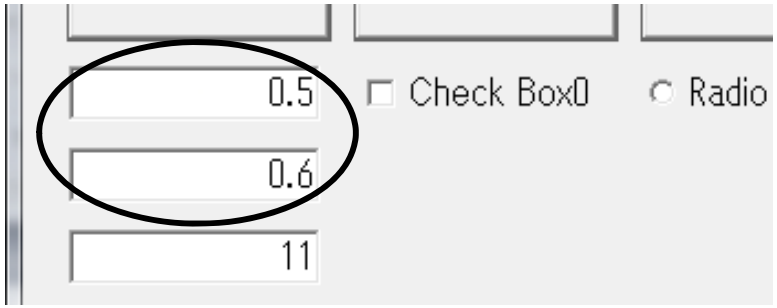
```
TCHAR a[10],b[10];
lstrcpy(a,TEXT(“DEF”));
```

TCHAR型文字列aに”DEF”を代入

```
wsprintf(b,TEXT(“ABC%s”),a);
```

TCHAR型文字列bに”ABCDEF”を代入

エディットボックス



・主にキーボード入力を行う。簡単な文字や整数値などを実行前に入力したり、ちょっとした結果を出力するのに使用する

・Get_int()関数等でエディットボックスから数値、テキストを取得

・Set_text()関数等で文字列の出力が可能である。

・エディットボックスの数がdef.h内の定数ED_NUMで指定

・エディットボックス番号 i は 0 番から ED_NUM-1 まで

・位置、サイズ、キャプションはsED[エディットボックス番号]構造体を変更する。

i番目のエディットボックスの設定	sED[i].x	エディットボックス左上のx座標
	sED[i].y	エディットボックス左上のy座標
	sED[i].w	エディットボックスの幅
	sED[i].h	エディットボックスの高さ
	sED[i].name	エディットボックスのキャプション(TCHAR型)

mlibの出力関数

mlibのエディット内文字列操作関数はchar型で定義されている。

```
void Set_double(int i , double x)
```

i番のエディットボックス内に、実数 x の値を出力する。

```
void Set_char(int i, char *Buffer)
```

i番のエディットボックス内に、指定したchar型変数の文字列を出力する。

```
void Printf(char *fmt, ...)
```

メモウインドウに書式付文字列表示を行う。使用例はprintf関数と全く同じchar型での書式指定が可能。

```
void main (int Number){
   double a=15;
   char c[100]="ABC";

   Set_double(0,1.2); 0番のエディットボックスの実数値1.2を出力
   Set_double(0,a);   0番のエディットボックスに実数変数aの数値を出力
   Set_char(1,"abc"); 1番のエディットボックスに文字列"abc"を出力
   Set_char(1,c);     1番のエディットボックスに文字列"ABC"を出力
```

使用例はこんな感じ

mlibの入力関数

```
int Get_int(int i)
```

i番のエディットボックス内の文字列を、整数値に変換して戻り値として返す。
数値以外の入力の時0を返す。

```
double Get_double(int i)
```

i番のエディットボックス内の文字列を、実数値に変換して戻り値として返す。
数値以外の入力の時0を返す。

```
void Get_char(int i, char *Buffer)
```

i番のエディットボックス内の文字列を読み取り、指定したchar型の文字配列変数Buffer[]にコピーする。

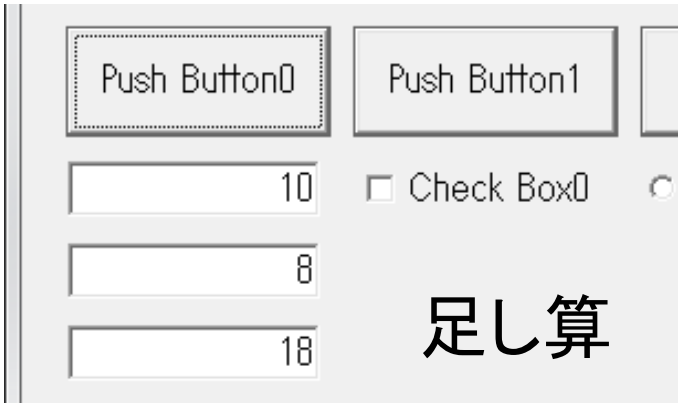
使用例はこんな感じ

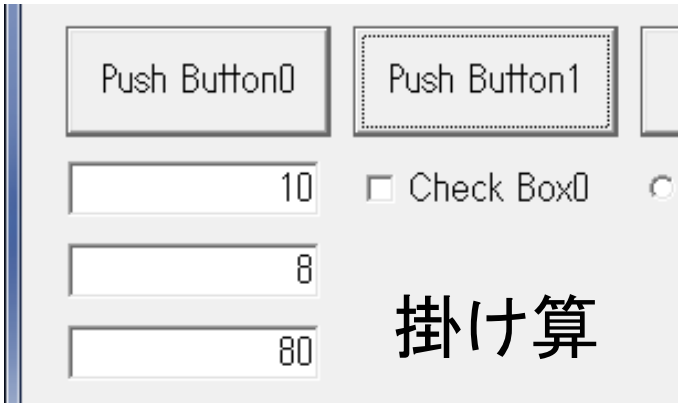
```
void main (int Number){
   double a;
   int b;
   char c[100];

   a=Get_double(0);  0番のエディットボックスの実数値を変数aに代入
   b=Get_int(1);     1番のエディットボックスの整数値を変数bに代入
   Get_char(2,c);    2番のエディットボックスの文字を文字配列cに代入
```

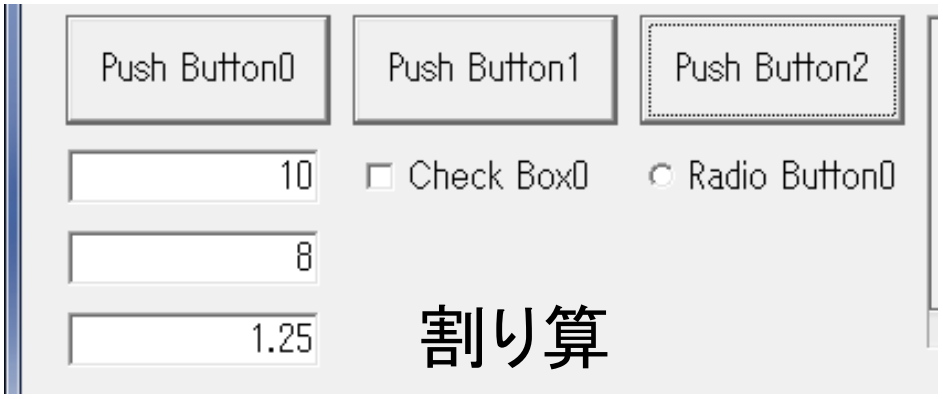
演習３－２

３－２．エディットボックス0と、エディットボックス1に入力された数値を、それぞれ変数a、b に代入し、エディットボックス2にその演算結果を出力するプログラムを作れ。

プッシュボタン0　→ a＋b
プッシュボタン1　→ a×b
プッシュボタン2　→ a÷b







mlibでの文字列の使用まとめ

mlib.h, def.h内部

ウインドウ内の文字表示にワイド文字を使用している。

ボタン等ウインドウの文字列にはTCHAR型を使うことで統一

mlibを利用するメインプログラム内

以下のmlib関数は関数内で自動的にワイド文字に変換してくれるので、
使いやすい char型を使っていればよい

メモウインドウへの出力	Printf関数
エディットボックスへの出力	Set_double関数, Set_char関数
エディットボックスからの入力	Get_int関数, Get_double関数, Get_char関数

Win32APIにすでに用意された関数や、サードパーティ製品の関数を使う場合には、ワイド文字列を使わないといけない。

その場合はTCHAR型の文字列と、_tで始まる文字列関数、_T(“ ”)や
TEXT(“ “)で囲まれた定数文字列を使えばよい。

演習　３－１

```
def.hでST_NUM=3に設定

component()関数内で


lstrcpy(sST[0].name,L"スタティック");

lstrcpy(sST[1].name,TEXT(“文字列“));

wsprintf(sST[2].name,TEXT(“%d番目“),3);
```

演習　３－２

```
void main (int Number){
  double a,b;
  a=Get_double(0);
  b=Get_double(1);

  switch(Number){
    case 0:
      Set_double(2,a+b);
      break;
    case 1:
      Set_double(2,a*b);
      break;
    case 2:
      Set_double(2,a/b);
      break;
  }
}
```