

4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。

C言語等で半角文字を使うときには

char 型 → 8bit の整数型

なので、char型が使われる。

```
char i;
```

変数 i をchar型(1バイト整数)で定義

i は半角1文字を表現できる。文字に対応する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

```
i =65; printf("%c", i);
```

A ASCIIコード65に対応する文字'A'が出力

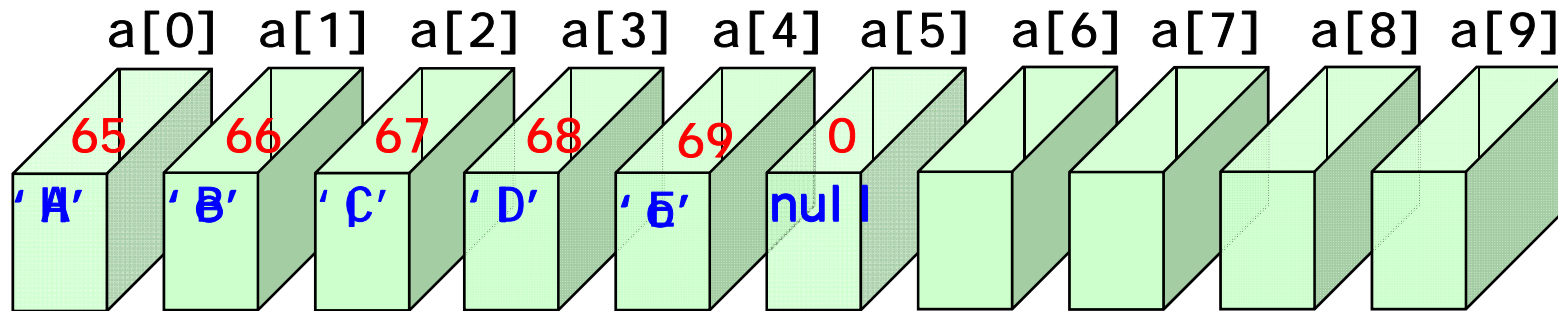
文字列の取扱い

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
```

配列変数では、変数名のみは0番目の要素のアドレス番号を示す。(a==&a[0])

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
```

```
printf("%sFG%s", a, "HIJ");
```

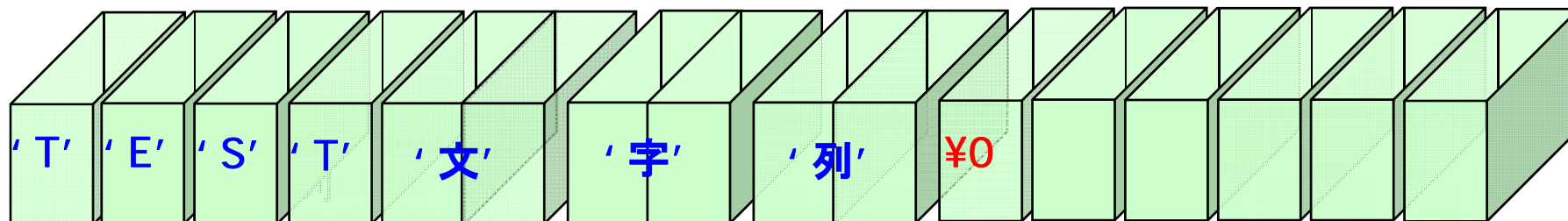
```
ABCDEFGHIJ
```

Windows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

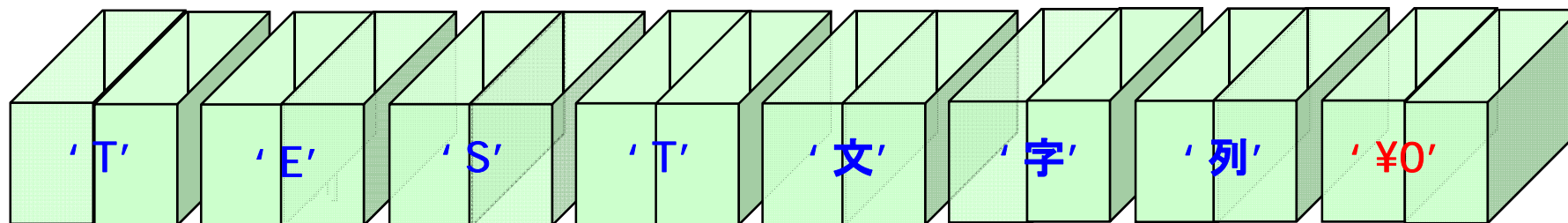
マルチバイト文字列(char型)

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に、特別なことは考えない

ワイド文字列(UNICODE)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、関数名が異なる

最近のWindowsはWin32APIがUNICODE文字しか対応しないので、古いOSと新しいOSではプログラムの互換性がない。

両方で動かすには、関数名や定数の定義が異なる2種類のプログラムが必要

TCHAR型

マルチバイト文字か、ワイド文字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。この型で書いていけば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。

_MBCSオプション

```
char A[10]="test文字列";
```

```
TCHAR A[10]=_T("test文字列");
```

_T()はTEXT()マクロと同じ

_UNICODEオプション

```
wchar_t A[10]=L"test文字列";
```

_MBCSオプション

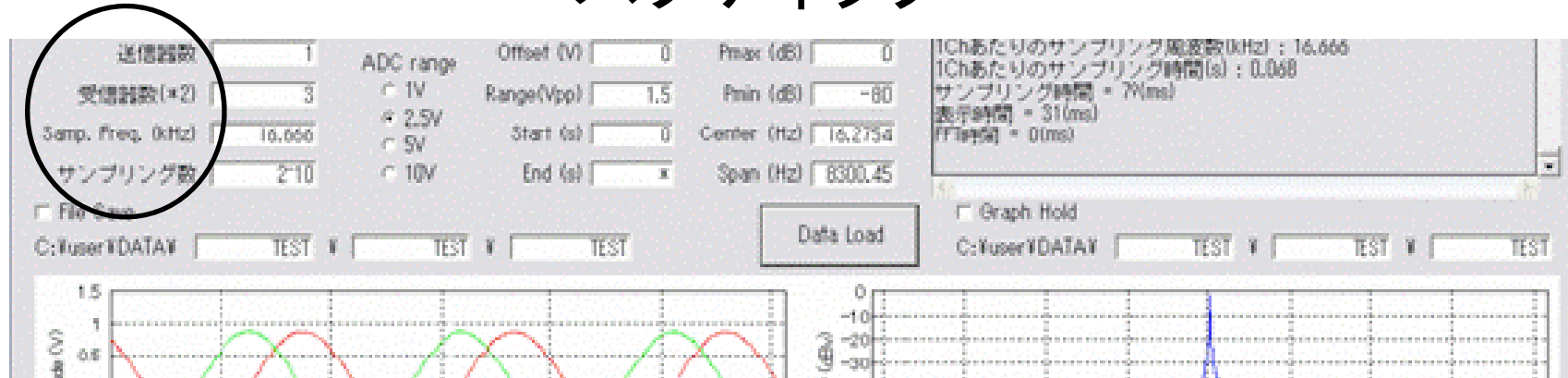
```
printf("文字列", char型変数等);
```

```
_tprintf(_T("文字列"), TCHAR型変数等);
```

_UNICODEオプション

```
wprintf(L"文字列", Wchar_t 型変数等);
```

スタティック



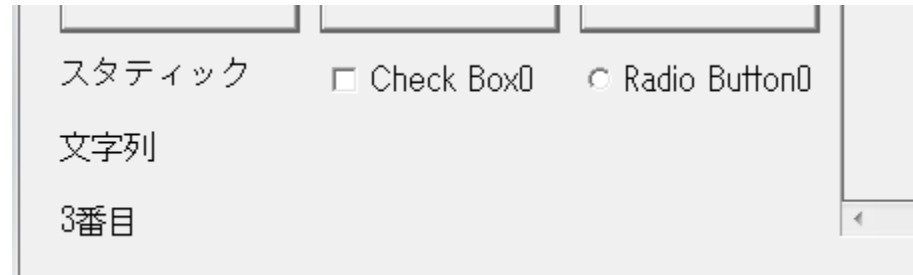
- ・文字のみを表示する。主に、エディットボックスの説明用に対になって利用する
- ・def.h内の定数`ST_NUM`で指定した個数のスタティックを配置できる
- ・位置、サイズ、キャプションは`sST[スタティック番号]`構造体を変更すればよい。

<code>sST[i].x</code>	スタティック左上のx座標
<code>sST[i].y</code>	スタティック左上のy座標
<code>sST[i].w</code>	スタティックの幅
<code>sST[i].h</code>	スタティックの高さ
<code>sST[i].name</code>	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、ラジオボタンのキャプションは**TCHAR型**で表記されるので注意

演習4

4-1. スタティクを3つ作成し、そのテキストに“スタティク”、“文字列”、“3番目”の名前を付ける



ヒント sST[0].name 等はdef.h、component()関数内で指定しないと反映されない、メインプログラム中にこれらの変数を変えてもダメ。

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型変数アドレス, TCHAR型文字列);
wsprintf(TCHAR型変数アドレス, TCHAR型文字列, 変数等);

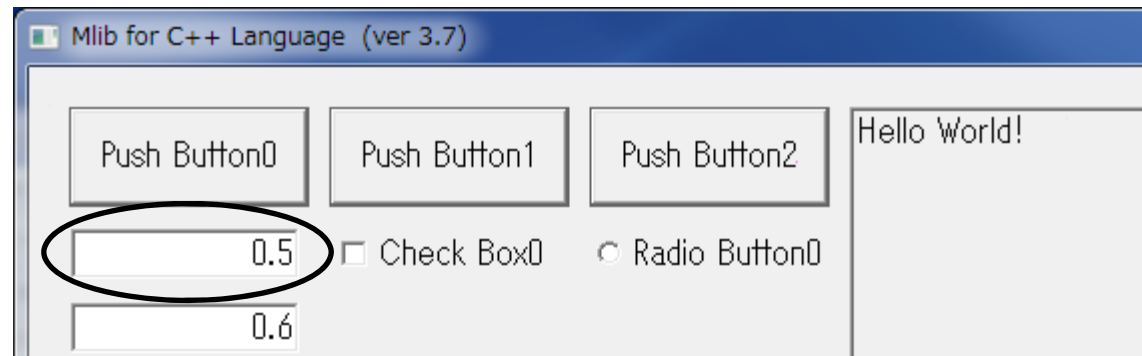
```
TCHAR a[10], b[10];  
lstrcpy(a, TEXT("DEF"));
```

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()`関数等で指定したエディットボックスから数値、テキストを取得できる。
- ・`Set_text()`関数等で文字列の出力が可能である。
- ・`def.h`内の定数`ED_NUM`で指定した個数のエディットボックスを配置できる
- ・位置、サイズ、キャプションは`sED[エディット番号]`構造体を変更すればよい。

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

mllibの出力関数

mllibのメインプログラム内ではchar型の文字列関数が用意されている

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

i番目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番目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文と全く同じ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);  
    Set_double(0,a);  
    Set_char(1,"abc");  
    Set_char(1,c);
```

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

mllibの入力関数

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

i番目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番目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番目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**の文字列変数にコピーする。

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

```
    a=Get_double(0);  
    b=Get_int(1);  
    Get_char(2,c);
```

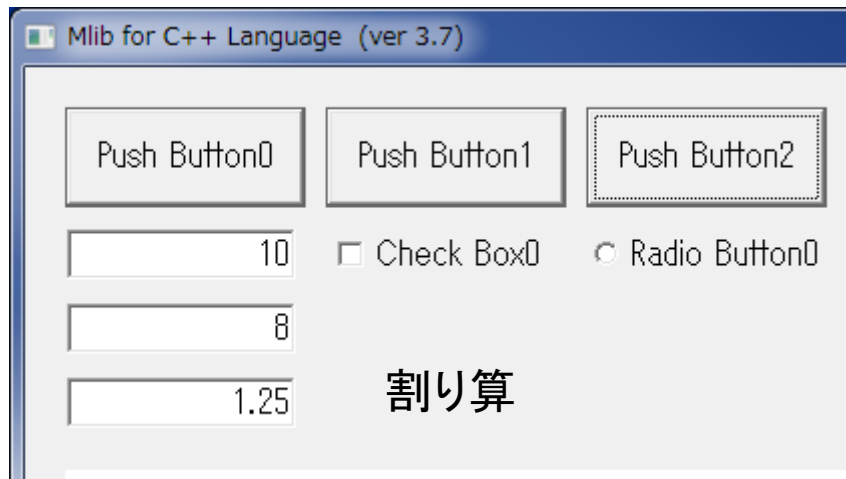
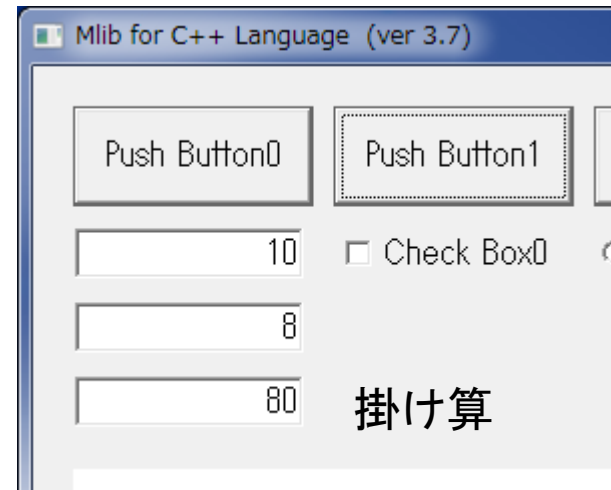
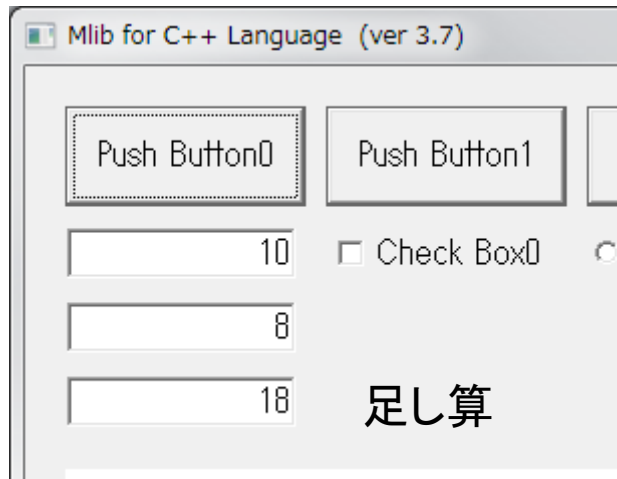
0番のエディットボックスの実数値を変数aに代入

1番のエディットボックスの整数値を変数bに代入

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5

5-1. 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読み取り、それぞれ a、b とすると、プッシュボタン0を押したときは $a+b$ を、プッシュボタン1を押したときは $a*b$ をプッシュボタン2を押したときは a/b をエディットボックス2に出力するプログラムを作れ。



mllibでの文字列の使用まとめ

mllib.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。

ウインドウに表示するための文字列にはTCHAR型を使う必要あり

mllibを利用するメインプログラム内

以下の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型を使っていけばよい

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。