プログラミング１　第 10 回教材　　担 当　神保 雅人

2012 年 6 月 20 日（木）実施

ポインタ変数と文字列

前回は，ポインタ演算が用いられる典型的な例として，ポインタ変数が 1 次元配列を指す場合を挙げたが，特に，char 型の配列に格納された文字列に対し，ポインタ変数に配列の 0 番の要素の先頭アドレスを代入して文字列を指すことで，配列そのものを操作するよりも便利な利用法が存在する。

なお，文字列リテラルは，その文字列が格納されている領域の先頭アドレスを表すので，次の例のようにポインタ変数への代入が可能である。この場合には，配列の初期化の際のような文字列の文字数+ 1 の領域を新たに確保することは行われない。

例）char *p = "Programming Language C";

例題 1（1 次元配列と文字列リテラルとの違い）

次のソースプログラムをテキストエディタで入力して，prog10-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* prog10-1.c */
#include <stdio.h>

int main(void)
{
    char dep1[] = "Faculty of Service Innovation";
    char *dep2 = "Faculty of Service Innovation";
    char *p;

    for (p=dep1; *p!='¥0'; p++)
        printf("%s¥n", p);

    for (p=dep2; *p!='¥0'; p++)
        putchar(*p);

    return 0;
}
```

【解説】

1. 配列の初期化の際に，代入するデータの要素数（この場合には文字数+1）を以て配列の要素数として設定することができる。この場合，[]内は空とする。

2. putchar は 1 文字を画面に表示するライブラリ関数である。

例題 2 （文字列の複写）

次のソースプログラムをテキストエディタで入力して，prog10-2.c の名前を付けて保存する。

それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* prog10-2.c */
#include <stdio.h>
#define  MAX  256

char *cpystr(char *, const char *);

int main(void)
{
    char str[MAX];
    char *uname = "Chiba University of Commerce";
    char *p = str;

    printf("cpystr(p, uname)の戻り値： %s¥n", cpystr(p, uname));
    printf("cpystr(p, uname)の呼び出しの結果，p の指す文字列： %s¥n", p);

    return 0;
}

char *cpystr(char *a, const char *b)
{
    char *c = a;     /* a のアドレスは変えずに指し示す中身だけ書き込んでいくため */

    while (*b)       /* b の指す文字が'¥0'でない間は継続 */
    {
        *c = *b;
        c++;
        b++;
    }
    *c = '¥0';       /* この箇所は，while (*c++ = *b++) ;とも書ける。 */

    return a;
}
```

【解説】

1. ユーザ定義関数 cpystr の 2 番目の引数は複写元の文字列を指すポインタを渡すため，書き換わらないよう，const 型修飾子を付けている。

2. この関数の 1 番目の引数には複写先の文字列を指すポインタを渡し，複写後の a の値を戻す仕様となっている。

＊ ポインタ変数 p が指し示す char 型の配列としては，充分大きな要素数のものを用意する。

例題 3 （文字列の長さ）

次のソースプログラムをテキストエディタで入力して，prog10-3.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* prog10-3.c */
#include <stdio.h>

unsigned int lenstr(const char *);

int main(void)
{
    char str[] = "Students' Projects";

    printf("lenstr の戻り値： %d¥n", lenstr(str));
    printf("sizeof の結果： %d¥n", sizeof str);

    return 0;
}

unsigned int lenstr(const char *a)
{
    unsigned int n;

    for (n = 0; *a != '¥0'; n++, a++)
        ;

    return n;
}
```

【解説】

sizeof は変数やポインタがメインメモリ上で占める領域の大きさをバイト単位で計算する。

演習 1

文字列の大小関係を比較するユーザ定義関数 cmpstr と，それを利用して**二つの文字列の大小関係を画面に表示する**ユーザ定義関数 disp を作成し，これらの関数の動作を確かめるプログラムを考える。文字列の大小関係は辞書順で定義される。cmpstr は，先頭からの位置が同じ文字同士の文字コードを順次比較し，等しくなくなった時点でその差を戻し，最後まで一致していれば 0 を戻す仕様となっている。この空欄 1)　　，2)　　，3)　　を埋めてソースプログラムを完成させ，テキストエディタで入力して，ex10-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* ex10-1.c */
#include <stdio.h>

int cmpstr(const char *, const char *);
1)      disp(const char *, const char *);

int main(void)
{
```

```
    char str1[] = "body";
    char str2[] = "book";
    char str3[] = "book";
    char str4[] = "apple";

    disp(str1, str2);
    disp(str2, str3);
    disp(str3, str4);

    return 0;
}

int cmpstr(const char *a, const char *b)
{
    2)      (*a == *b)
    {
        a++;
        b++;

        if (*a == '¥0')
            return 0;
    }

    return *a - *b;
}

void disp(const char *c, const char *d)
{
    int diff;

    diff = cmpstr(c, d);

    if (diff < 0)
        printf("[%s] < [%s]¥n", c, d);
    else if (diff 3)      0)
        printf("[%s] == [%s]¥n", c, d);
    else
        printf("[%s] > [%s]¥n", c, d);
}
```

演習 2 (余裕のある人向け)

例題 3 の文字列の長さを測定するユーザ定義関数 lenstr を利用して，**文字列を連結する**ユーザ定義関数 catstr を作成し，この関数の動作を確かめるプログラムを考える。catstr は，1 番目の引数で与えた文字列に 2 番目の引数で与えた文字列を連結して戻す仕様となっている。この空欄 1)　，2)　，3)　を埋めてソースプログラムを完成させ，テキストエディタで入力して，ex10-2.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。(提出するファイルは ex10-2.c の完成版とする)

```
/* ex10-2.c */
#include <stdio.h>
#define MAX 256

unsigned int lenstr(const char *);
1)      *catstr(char *, const char *);

int main(void)
{
    char str1[MAX] = "Service ";
    char str2[] = "Innovation";

    printf("文字列 1: [%s]¥n", str1);
    printf("文字列 2: [%s]¥n", str2);
    printf("連結後の文字列: [%s]¥n", 2)      (str1, str2));

    return 0;
}

unsigned int lenstr(const char *a)
{
    unsigned int n;

    for (n = 0; *a != '¥0'; n++, a++)
        ;

    return n;
}

char *catstr(char *a, const char *b)
{
    3)      *c = a + lenstr(a);  /* a の指し示している先の最後の要素のアドレスをセット */

    while (*b)
    {
        *c = *b;
        c++;
        b++;
    }
    *c = '¥0';

    return a;
}
```

＊ 文字列の複写，文字列の長さの測定，文字列の大小関係の比較，文字列の連結は，それぞれ，文字列操作のライブラリ関数 strcpy，strlen，strcmp，strcat を利用して行うことが出来る。これらを利用するには，#include <string.h>が必要である。今回の例題や演習問題はこれらの関数の実装の仕方の一例となっている。なお，実用的なプログラムでは，複写する文字列や連結する文字列のバイト数を指定する strncpy や strncat を用いた方が安全である。

提出物：

1) 例題 1，2，3 の出力結果をコピーして貼り付けたテキストファイル res10.txt

2) 演習 1 のソースプログラムのファイル ex10-1.c の完成版