プログラミング１　第 11 回教材　　担 当　神保 雅人

2013 年 6 月 27 日（木）実施

ポインタの配列

C 言語では，ポインタを要素とする配列を次のように宣言して，利用することが出来る。

データ型 *ポインタ配列名[要素数]

例）char *p[3];

ポインタ配列の個々の要素はポインタ変数と同等であるので，例えば，個々の要素が char 型の 1 次元配列を指すポインタ配列を次の様に初期化して宣言することが出来る。

```
例）char a1[] = "Service";
    char a2[] = " ";
    char a3[] = "Innovation";
    char *p[] = {a1, a2, a3};
```

この例で 3 個の char 型の 1 次元配列を char 型の 2 次元配列で纏めると，

```
例）char a[][11] = {"Service", " ", "Innovation"};
    char *p[] = {a[0], a[1], a[2]};
```

と表せる。これらの例で，p[i]はポインタ配列 p の i 番目の要素を表し，ポインタ配列名 p は p[0]へのポインタの扱いとなる。また，*(p+i)は p[i]と同等な表現となる。なお，p[i][j]で p の i 番目の要素が指す先の j 番目の文字を表すことが出来る。

＊ 第 4 回の教材で扱った二つの仮引数を持つ main 関数の 2 番目の仮引数 char *argv[]はポインタ配列である。

＊＊ 2 次元配列の初期化では 2 番目の添え字に関する要素数は省略できないので，文字列が格納し得る大きさを指定する。

例題 1（**ポインタ配列要素： 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，prog11-1.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。なお，**複数回実行して，値が変わるものと変わらないものとを確認する**こと。

```
/* prog11-1.c */
#include <stdio.h>

int main(void)
{
    char a1[] = "Service";
    char a2[] = " ";
    char a3[] = "Innovation";
    char *p1[] = {a1, a2, a3};
    char *p2[] = {"Service", " ", "Innovation"};
```

```
    printf("a1 の値は%#x¥n", a1);
    printf("a2 の値は%#x¥n", a2);
    printf("a3 の値は%#x¥n¥n", a3);

    printf("p1 の値は%p¥n", p1);
    printf("p1[0]の値は%p¥n", p1[0]);
    printf("p1[1]の値は%p¥n", p1[1]);
    printf("p1[2]の値は%p¥n", p1[2]);
    printf("p1[0]の指す先に格納された文字列は[%s]¥n", p1[0]);
    printf("p1[1]の指す先に格納された文字列は[%s]¥n", p1[1]);
    printf("p1[2]の指す先に格納された文字列は[%s]¥n¥n", p1[2]);

    printf("p2 の値は%p¥n", p2);
    printf("p2[0]の値は%p¥n", p2[0]);
    printf("p2[1]の値は%p¥n", p2[1]);
    printf("p2[2]の値は%p¥n", p2[2]);
    printf("p2[0]の指す先に格納された文字列は[%s]¥n", p2[0]);
    printf("p2[1]の指す先に格納された文字列は[%s]¥n", p2[1]);
    printf("p2[2]の指す先に格納された文字列は[%s]¥n", p2[2]);

    return 0;
}
```

＊ ポインタ配列 p2 の各要素は，文字列リテラルを指していることに注意する。

例題 2（**ポインタ配列が指す文字列中の特定の文字**）

次のソースプログラムをテキストエディタで入力して，prog11-2.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。

```
/* prog11-2.c */
#include <stdio.h>

int main(void)
{
    char a[][11] = {"Service", " ", "Innovation"};
    char *p[] = {a[0], a[1], a[2]};
    int i, j;

    for (i=0; i<3; i++)
        printf("a[%d]の値は%#x¥n", i, a[i]);

    printf("====¥n");

    printf("p の値は%p¥n", p);

    for (i=0; i<3; i++)
        printf("*(p+%d)の値は%p¥n", i, *(p+i));

    for (i=0; i<3; i++)
        printf("*(p+%d)の指す先に格納された文字列は[%s]¥n", i, *(p+i));

    printf("====¥n");

    for (j=0; p[0][j] != '¥0'; j++)
```

```
        printf("p[0][%d] -> '%c'¥n", j, p[0][j]);

    return 0;
}
```

演習 1

文字列の大小関係を比較するライブラリ関数 strcmp 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。strcmp 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，ex11-1.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。

```
/* ex11-1.c */
#include <stdio.h>
#include <string.h>  /* strcmp を利用するために必要 */

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

int main(void)
{
    char str[][2)    ] = {"body", "book", "book", "apple"};
    3)      *p[] = {str[0], str[1], str[2], str[3]};
    int i;

    for (i=1; i<4; i++)
        disp(p[i-1], p[i]);

    return 0;
}

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

    diff = strcmp(c, d);

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

提出物：

1) 例題 1，2 の出力結果をコピーして貼り付けたテキストファイル res11.txt

2) 演習 1 のソースプログラムのファイル ex11-1.c の完成版