

# プログラミング基礎


<http://bit.ly/prog2d>

## 文字列と標準ライブラリ関数

前期 第9週

2017/6/19


```
1: #include <stdio.h>
2: #include <ctype.h>
3:
4: char str1[50];
5:
6: void toupper_array();
7:
8: void toupper_array()
9: {
10:     int i;
11:     for(i=0; str1[i]!='\0'; i++) {
12:         str1[i] = toupper(str1[i]);
13:     }
14: }
```



**【Point 1】** 文字に対する標準ライブラリ関数を繰り返しの  
中で使うと、文字列に対する処理に使うことができる  
(ctype.hのライブラリ関数は p. 466参照)

str1に文字列を入力し、関数を呼び出し後に表示すると、str1の文字列の変化が確認できる。

```
15:
16: int main( )
17: {
18:     printf("str1 > ");
19:     scanf("%s", str1);
20:     toupper_array();
21:     printf("str1: %s\n", str1);
22:     return 0;
23: }
```



# 【練習9-1】

サンプルプログラムをコンパイルして、  
実行結果を確認しましょう。

（コンパイルには「-lc」（エルとシー）  
を付けて実行してください）

# 【課題9-1】

**str1 に格納された文字列に対して、小文字に変換する関数tolower\_array()を作成してください。**  
**使用する標準ライブラリ関数はp. 466の<ctype.h>を参照しましょう。**

[この関数のプロトタイプ宣言]

```
void tolower_array();
```

```
/* str1に格納されている文字列に対して繰り返し処理する */
```

```
/* tolower()を使って、それぞれの文字を小文字に変換する */
```

# 【課題9-1】

[ 配列str1をグローバル変数として宣言する ]

```
char str1[50];
```

[mainでの処理]

```
printf("str1 > ");  
scanf("%s", str1);  
tolower_array();  
printf("str1: %s\n", str1);
```

[実行結果]

```
str1 > XBOX_One  
str1: xbox_one
```

# 【課題9-2】

**str1 に格納された文字列に対して、数字を「\*」（アスタリスク）に変換する関数asterisk\_array()を作成してください。**

[この関数のプロトタイプ宣言]

```
void asterisk_array();
```

```
/* str1に格納されている文字列に対して繰り返し処理する */
```

```
/* 関数isdigit()を使ってそれぞれの文字が数値かどうかを判別する */
```

# 【課題9-2】

[ 配列str1をグローバル変数として宣言する ]

```
char str1[50];
```

[mainでの処理]

```
printf("str1 > ");  
scanf("%s", str1);  
asterisk_array();  
printf("str1: %s\n", str1);
```

[実行結果]

```
str1 > Xbox360  
str1: Xbox***
```



# 【課題9-3】

**str1 に格納された文字列に対して、小文字を大文字にし大文字を小文字にする（つまり文字の大文字/小文字を入れ替える）関数change\_char()を作成してください。**

[この関数のプロトタイプ宣言]

```
void change_char();  
/* str1に格納されている文字列に対して繰り返し処理する */  
/* 関数islower(), isupper()を使って、  
    それぞれの文字の大文字/小文字を判別する */  
/* 判別に応じて関数toupper(), tolower()を使って文字を変換する */
```

# 【課題9-3】

[ 配列str1をグローバル変数として宣言する ]

```
char str1[50];
```

[mainでの処理]

```
printf("str1 > ");  
scanf("%s", str1);  
change_char();  
printf("str1: %s\n", str1);
```

[実行結果]

```
str1 > IpHONE5s  
str1: iPhone5S
```

# 【課題9-4】

まず、8文字分の文字列を格納する配列str2（つまり、終端文字も含めると9文字分必要）を、以下のようにグローバル変数として宣言する。

```
char str2[9];
```

このstr2に格納された2進数を表す8桁の文字から、**10進数の整数**を計算し返す関数todecimal()を作成してください。

ただし、str2には常に'0'か'1'の文字が8個入っている（つまり、常に8桁の2進数を表す文字列が入っている）ものとしてプログラムを作成してよいです。

# 【課題9-4】

[この関数のプロトタイプ宣言]

```
int todecimal();  
/* strに格納されている8個の文字に対して繰り返し処理する */  
/* 繰り返し処理で、i番目の文字が'1'であるかどうかを判別する */  
/* '1'という文字である場合は、その桁に対応したpow()の計算をする */  
/* (コンパイルには -lm が必要) */
```

[例]

「10011010」という文字列の場合、  
「 $2^7 + 2^4 + 2^3 + 2^1 = 154$ 」となる  
(符号ビットは考慮しなくてよい)

# 【課題9-4】

[ 配列str2をグローバル変数として宣言する ]

```
char str2[9];
```

[main()での処理]

```
printf("str2> ");  
scanf("%s", str2);  
printf("todecimal: %d\n", todecimal());
```

[実行結果]

```
str2> 10011010  
todecimal: 154
```

# まだ余裕のある人は…【課題9-5】

課題8-4で作成したプログラムを8桁以外の2進数にも対応できるように、マクロ「#define」を使って改良してみましょう。

# 小テストについて

## 小テストの注意点

- 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）
- 小テスト中は、演習室外へのネットワークアクセスは遮断される。

# 小テストについて

## 小テスト中に参照できるもの

- 教科書, 配付資料
- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
- 小テストでは紙媒体のものは参照可能
- 上記以外の情報を参照することは不正行為とする  
例：USBで接続された機器に保存されているファイルの参照  
ネットワークを介した情報の参照、など