

シラバス

19,20. 文字列

文字配列を使った処理方法の理解

事前学修：授業スライド「第10回 文字列」と教科書p.154～p.163の熟読。（60分）

事後学修：授業スライドの用語とその意味の理解。C言語における文字型，文字列，文字列の入力と代入方法，ヌル文字を説明できること。（60分）

演習課題のプログラムの完成と提出。（180分）

プログラミングの基礎及び演習

日本大学 工学部 情報工学科

プログラミングの基礎 第10回

■ 文字配列

- 複数の文字の扱い
- 文字列と文字の違い
- 文字列の入力(scanf)と代入
- 2次元配列と文字列の配列
- 演習課題のヒント

- 教科書 p.154～p.163

文字の扱い

■ これまでに扱ってきたデータの型

- 整数型 `int`型
- 浮動小数点型 `double`型

■ では、文字はどのように扱うか？

- 文字型 `char`型
- 文字型は、1 個の文字を格納するデータ型
- 複数の文字による文字列
⇒ 文字型の配列を使う (文字列)

文字型

- C言語では文字は「'」（シングルクォーテーション）記号で囲んで表現
 - 例: 'A'
 - '' 記号で囲めるのは「1文字」だけ

'ABC' ⇒ エラー

```
char ch;  
ch = 'A';  
printf("変数chの値: %c\n", ch );
```

実行結果：
変数chの値: A

printfで文字を表示
⇒ %c

文字と文字列

- 人間にとっては
1文字の文字も複数文字からなる文字列も
大きな違いはない.
- C言語では,
1文字の文字と1文字以上の文字列を
区別している.
 - 1文字の文字と 1文字の文字列もC言語では,
扱いが異なる.

文字列

- C言語では文字列は
「"」（ダブルクォーテーション）記号で
囲んで表現
 - 例: "abc" "a"
 - ""記号で囲めるのは「1文字」でも 可

"A" ⇒ エラーではない

- 文字列は,文字の 1 次元配列として扱う

プログラミングの基礎 第10回

■ 文字配列

- 複数の文字の扱い
- 文字列と文字の違い
- 文字列の入力(scanf)と代入
- 2次元配列と文字列の配列
- 演習課題のヒント

- 教科書 p.154～p.163

文字列

- C言語では文字列は「"」記号で囲む
 - 例： "これは文字列の例です"
 - 複数文字を囲むこともできる（1文字でも可）
 - 文字列は、文字型（char型）の配列である

文字列 "ABC" を配列で表すと...

A	B	C	¥0
[0]	[1]	[2]	[3]

長さ3の文字列は、
大きさ4の配列になる

ナル文字（null文字）
「文字列の最後」を意味する記号
'¥0'とも書く

文字列と文字の違い

■ 'A' と "A" はどう違うか？

'A'

「文字」

純粹に**1文字だけ**
を表す→配列ではない

A

複数文字を入れることは
できない

"A"

「文字列」

'A'という文字 + **ナル文字**
の**2文字**を表す→配列を用いる

A	¥0
---	----

何文字でもOK

「文字列」には、必ず末尾に
ナル文字が付く
(ナル文字がないと、どこが
文字列の末尾なのか分からないから)

文字列の例 (1)

例題11.1

```
int main( void ){  
    char name[7] = { 't', 'a', 'n', 'a', 'k', 'a', '¥0' };  
    printf( "%s¥n", name );  
    return 0;  
}
```

文字列は%sで表示

「配列の名前」だけを書く
「添字」は付けない

文字列は「文字の配列」なので、
通常の配列と同様に初期化できる。
最後にナル文字を付けるのを忘れずに

「配列の大きさ」には、ナル文字の分も
含まれる

name[0] [1] [2] [3] [4] [5] [6]

t	a	n	a	k	a	¥0
---	---	---	---	---	---	----



文字列の例 (2)

""を使って書くときには
ナル文字は省略
自動的にナル文字が入る

文字列の代入方法は色々ある

```
int main(void) {  
    char name[7];  
  
    name[0] = 't';  
    name[1] = 'a';  
    name[2] = 'n';  
    name[3] = 'a';  
    name[4] = 'k';  
    name[5] = 'a';  
    name[6] = '\0';  
    printf("%s\n", name);  
    return 0;  
}
```

1文字ずつ
代入できる
これは文字
なので
で囲む

初期化時に代入する場合のみ
配列の大きさは省略可能
(文字列の長さから計算可能)

```
int main(void) {  
    char name[7] = "tanaka";  
    printf("%s\n", name);  
    return 0;  
}
```

例題11.2

```
int main(void) {  
    char name[] = "tanaka";  
    printf("%s\n", name);  
    return 0;  
}
```

代入は「宣言時」のみ可能

```
int main(void) {  
    char name[7];  
    name[] = "tanaka";  
    printf("%s\n", name);  
    return 0;  
}
```

エラー



文字列の扱い

文字列を逆にするプログラム

例題11.3

```
int main(void) {  
    int i;  
    char eman[7], name[] = "tanaka";  
    for( i = 0; i <= 5; i++ ) {  
        eman[5-i] = name[i];  
    }  
    eman[6] = name[6];  
    printf( "eman = %s¥n", eman );  
    return 0;  
}
```

省略不可

省略可

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
name	t	a	n	a	k	a	¥0
eman	a	k	a	n	a	t	¥0

実行結果

eman = akanat

```
i=0  
    eman[5] = name[0]  
i=1  
    eman[4] = name[1]  
i=2  
    eman[3] = name[2]  
i=3  
    eman[2] = name[3]  
i=4  
    eman[1] = name[4]  
i=5  
    eman[0] = name[5]
```

文字列の宣言と初期化

- `char eman[7], name[] = "tanaka";`
- `name` の方は“tanaka”で初期化
6文字＋最後のナル文字でサイズが
決まるので、
配列要素の個数 7 を省略できる
- `eman`の方は初期化していないので、
配列サイズの指定が必要

プログラミングの基礎 第10回

■ 文字配列

- 複数の文字の扱い
- 文字列と文字の違い
- 文字列の入力(scanf)と代入
- 2次元配列と文字列の配列
- 演習課題のヒント

- 教科書 p.154～p.163

文字列の入力

■ scanfで文字列を入力する方法：

例題11.4

```
char name[10];  
scanf("%s", name);
```

ナル文字を含めて10文字まで読み込める
(実質9文字)
それ以上読み込ませると誤動作する

%s で読み込む

配列名だけ書く
添字は付けない

端末から

```
**@cse-gw[53]:./a.out  
tanaka[Enter]
```

のように数値と同様に入力

通常のscanfと異なり、
変数名の前に&記号を書かない
(**文字列の場合のみ例外**)

文字列の代入 (1)

■ 配列をまとめて代入することは不可能

例1 :

```
int array1[3] = { 1, 2, 3 };  
int array2[3];
```

~~**array2[] = array1;**~~

例2 :

```
char str1[] = "ABCD";  
char str2[5];
```

~~**str2[] = str1;**~~

どちらもエラーになる

文字列も配列の一種
(文字型の配列)



文字列の代入 (2)

■ 文字列をまとめて代入（コピー）する関数

strncpy(代入先文字列, 代入元文字列,
コピーできる最大文字数)

#include <string.h> が必要

```
#include <stdio.h>
#include <string.h>
#define LEN 10
int main(void) {
    char name[LEN];
    strncpy(name, "tanaka", LEN-1);
    printf("name = %s\n", name);
    return 0;
}
```

例題11.7改題

実行結果

name = tanaka

文字列"tanaka"を
文字列nameに代入
格納可能な文字数
(文字配列の
要素数) も指定

strncpy

- `strncpy(char dst[], char src[], int max);`

<code>char dst[]</code>	コピー先文字列
<code>char src[]</code>	コピー元文字列
<code>int max</code>	コピーできる最大文字数

- 文字列srcを文字列dstに最大max文字コピー
- srcの長さがmax以下の場合はmax数文字をコピーするが、ナル文字'¥0'の自動付加は**行わない**ので注意が必要
- srcの長さがmaxがより小さい場合はdstには残りmaxの長さまでナル文字'¥0'が埋め込まれる

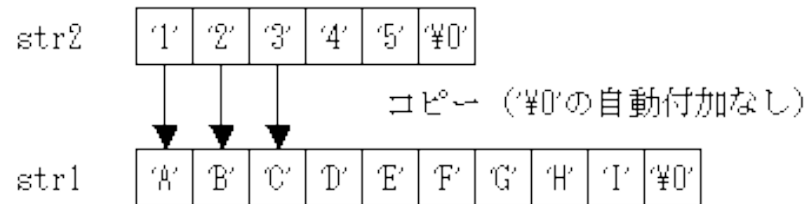
strncpy

```
char str1[] = "ABCDEFGHI";  
char str2[] = "12345";  
  
strncpy(str1, str2, 3); /* ①s2の長さがn以上のとき */  
printf("str1:%s\n", str1);  
strncpy(str1, str2, 8); /* ②s2の長さがnより少ない場合 */  
printf("str1:%s\n", str1);
```

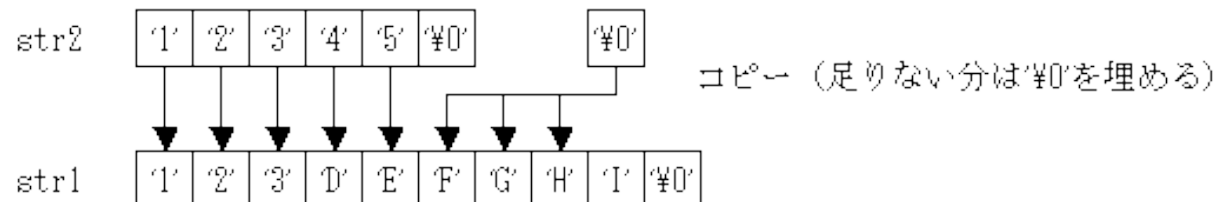
実行結果

```
str1:123DEFGHI  
str1:12345
```

① str2 の長さ $\geq n$



② str2 の長さ $< n$



strncpyを実行した
あとで、ナル文字の
入力を忘れないように

```
dst[max]='\0';
```

strncpyとstrcpy

- strncpy でなぜ文字数指定をするか？
- 代入される側の文字列変数の長さ
（文字の配列としての要素の数, サイズ）
が足りないと, プログラムが異常な動作をする可能性がある.
- strncpyから文字数の指定を省いたものに
strcpy関数があるが, **使わない方がよい**



プログラミングの基礎 第10回

■ 文字配列

- 複数の文字の扱い
- 文字列と文字の違い
- 文字列の入力(scanf)と代入
- 2次元配列と文字列の配列
- 演習課題のヒント

- 教科書 p.154～p.163

2次元配列の意味

- 2次元配列は,
「(1次元)配列を要素としてもつ配列」
 - 例: `int a[3][4];`
 - 「要素数 4 の(1次元)配列」を要素とする
要素数 3 の配列」

`a[0]`と書けば,
要素数 4 の1次元配列となる.
`a[0]`の要素は{ 3, 2, 4, 5 }

比較: `int b[4] = {1,2,3,4};`のとき
`b`と書けば,
要素 4 の一次元配列となる

a[0]	[0][0]	[0][1]	[0][2]	[0][3]
	3	2	4	5
a[1]	[1][0]	[1][1]	[1][2]	[1][3]
	5	3	4	2
a[2]	[2][0]	[2][1]	[2][2]	[2][3]
	2	3	4	4

文字列の配列 (1)

■ 文字列 ⇒ 1次元のchar型の配列

- char型配列一つで一つの文字列

- 例えば, `char str[6] = "abcde";`

■ では, 複数の文字列を表すには?

- 文字列の配列

文字列 = 「char型の1次元配列」

文字列の配列 = 「char型の1次元配列」の配列

= 「char型の2次元配列」

2次元配列 = 「1次元配列」の配列 だから

文字列の配列 (2)

```
char a[3][20] = { "Noguchi",  
                  "Higuchi",  
                  "Fukuzawa"};
```

「文字列の終わり」
よりも後の配列要素
には**ナル文字**が入る

a[0]	[0][0] N	[0][1] o	[0][2] g	[0][3] u	[0][4] c	[0][5] h	[0][6] i	[0][7] ¥0	[0][8] ¥0	[0][9] ¥0	...
a[1]	[1][0] H	[1][1] i	[1][2] g	[1][3] u	[1][4] c	[1][5] h	[1][6] i	[1][7] ¥0	[1][8] ¥0	[1][9] ¥0	...
a[2]	[2][0] F	[2][1] u	[2][2] k	[2][3] u	[2][4] z	[2][5] a	[2][6] w	[2][7] a	[2][8] ¥0	[2][9] ¥0	...

a[1]は “Higuchi” という文字列を表す

a[2][4]は 'z' という1文字を表す



文字列の配列 (3)

```
int main(void){
    int i;
    char a[3][20] = { "Noguchi",
                      "Higuchi",
                      "Fukuzawa" };

    for (i = 0; i < 3; i++) {
        hyoji(a[i]);
    }
    return 0;
}

void hyoji(char b[20]){
    printf("%s\n", b);
}
```

実行結果
Noguchi
Higuchi
Fukuzawa

$a[i]$ は「 i 番目の文字列」
を意味する

$a[i]$ という
「1次元char型配列」を
関数に渡す

$a[0]$ は"Noguchi" という文字列

「要素数20の1次元char型配列」
として受け取る

文字列の配列 (4)

■ strncpyを使って代入する

```
#include <stdio.h>
#include <string.h>
#define LEN 20
int main(void){
    int i;
    char a[3][LEN];
```

← strncpyを使うには
string.hが必要

← 文字列の配列を宣言
(長さ19の文字列を3つ)

```
    strncpy(a[0], "Noguchi", LEN-1);
    strncpy(a[1], "Higuchi", LEN-1);
    strncpy(a[2], "Fukuzawa", LEN-1);
```

← 文字列に代入
(a[i]は3つの文字列
のうち、(i+1)番目の
文字列を表す)

```
    for (i = 0; i < 3; i++) {
        printf("a[%d] = %s¥n", i, a[i]);
    }
    return 0;
```

← 文字列の表示

```
}
```

【演習課題】

- 演習課題提出システムの
「第10回」演習課題を実施
- 演習課題10-1, 10-2, 10-3, 10-4
 - 演習時間に，設計後に演習環境にリモートログインしプログラムを作成
 - 演習環境でコンパイル，テストし，完成
 - ソースファイル（10-x.c）をWinSCP等でPCへ転送
 - 課題提出システムへSubmitし各課題100点を目指す
 - 時間内に終了しない場合 ⇒ 宿題
- 授業翌週水曜日の午後5時までに必ず提出
- レポート，今回はなし

プログラミングの基礎 第10回

■ 文字配列

- 複数の文字の扱い
 - 文字列と文字の違い
 - 文字列の入力(scanf)と代入
 - 2次元配列と文字列の配列
 - 演習課題のヒント
-
- 教科書 p.154～p.163



演習課題のヒント(1) 文字列の長さ

■ (1) 関数get_lengthの考え方

- 文字列の末尾には、必ずナル(NULL)文字'¥0'がある.
- 文字列の先頭から順に文字数を数えていき、ナル文字が出現したところまでの文字数が、文字列の長さである.
- 長さの表示はmain関数内で行う

A	B	C	D	¥0
[0]	[1]	[2]	[3]	[4]

ナル文字が[4]なので、文字列の長さも 4 である

「文字列の長さ」にはナル文字は含まない



ヒント（２） %c でうまく入力ができない

- 一つ前の入力の改行文字が原因
- キーボードから入力された文字はバッファにいったん読み込まれ、エンターキーが入力されるとフラッシュ（バッファをクリア）される。
- scanf() では、正しく入力しても最後の改行文字はバッファの中に残るため、おかしい動作になる。
（scanf() 実行後に、scanf() の "%c" 指定したとき）

```
scanf("%d", &a1);
```

```
scanf("%c", &a2);
```

（回避方法）

```
scanf("%*c%c", &c);
```

↑ 改行の読み捨て

```
scanf(" %c", &c);
```

↑ 空白(改行含む)を読み飛ばす。

ただし、空白文字を入力したいときにはこの方法は使えない

入力しようとしたら、改行がscanfされる。