

cp-12. 文字列

(C プログラミング入門)

URL: <https://www.kkaneko.jp/pro/adp/index.html>

金子邦彦





文字列とは

- 数字，文字，記号の列
- 文字列データを扱うための最も簡単な方法は，文字の**配列**を使う方法である．

内容



例題 1． 文字列と長さの表示

文字の配列としての文字列, 文字列の長さ, 文字列の末尾

例題 2． 文字列のコピー

例題 3． 文字列の連結

例題 4． 文字列の比較

例題 5． 文字列の検索

文字列のためのライブラリ関数

例題 6． 文字列のメモリアドレス

文字列を扱う関数

例題 7． 曜日の表示

文字列の配列

- 文字列を扱うような簡単な関数を理解し，自分で書けるようになる



例題 1 . 文字列と長さの表示

- 文字列を読み込んで，長さを表示するプログラムを作る.

例) "Computer" の長さは 8

"高度プログラミング演習" の長さは
2 2

- ここでの文字列は，半角の「空白文字」を含まないものとする
- 半角文字は 1，全角文字は 2 として数える



```
#include <stdio.h>
```

```
#include <string.h>
```

```
#pragma warning(disable:4996)
```

```
int main()
```

```
{
```

```
char x[100];
```

```
int len;
```

```
printf("string=");
```

```
scanf("%s", x);
```

```
len = strlen(x);
```

```
printf("strlen(%s) = %d¥n", x, len);
```

```
return 0;
```

```
}
```

文字列を格納するための配列の宣言
(「char」とあるのは「文字」という意味)

文字列データの読み込み

「%s」は文字列の意味. 「x」は、配列の先頭要素のメモリアドレスの意味

文字列の長さを求める

文字列データと長さの表示.
「%s」は文字列の意味



文字列と長さの表示

実行結果の例（１）

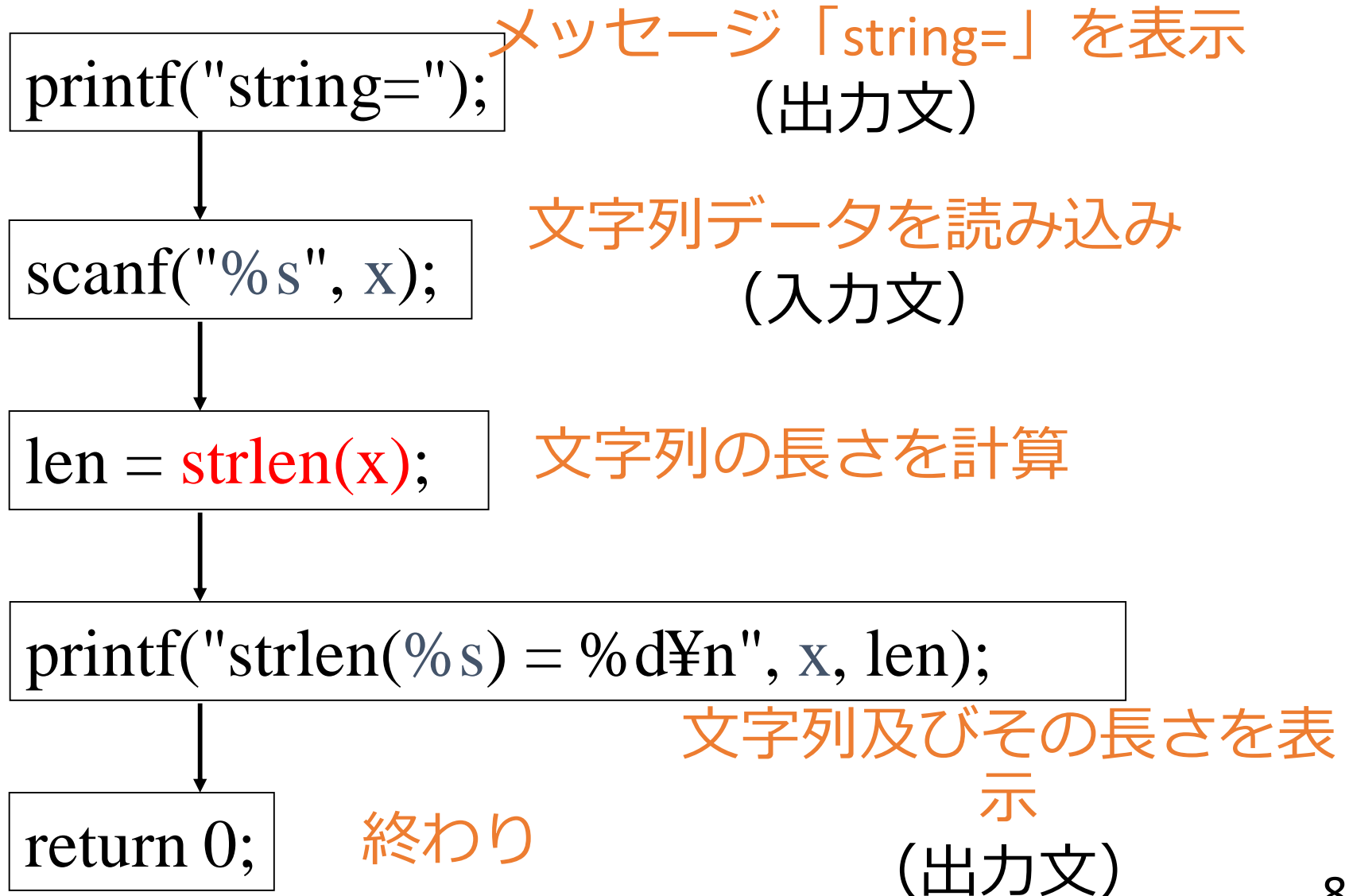
```
string=Computer  
strlen(Computer) = 8
```

実行結果の例（２）

```
string=高度プログラミング演習  
strlen(高度プログラミング演習) = 22
```



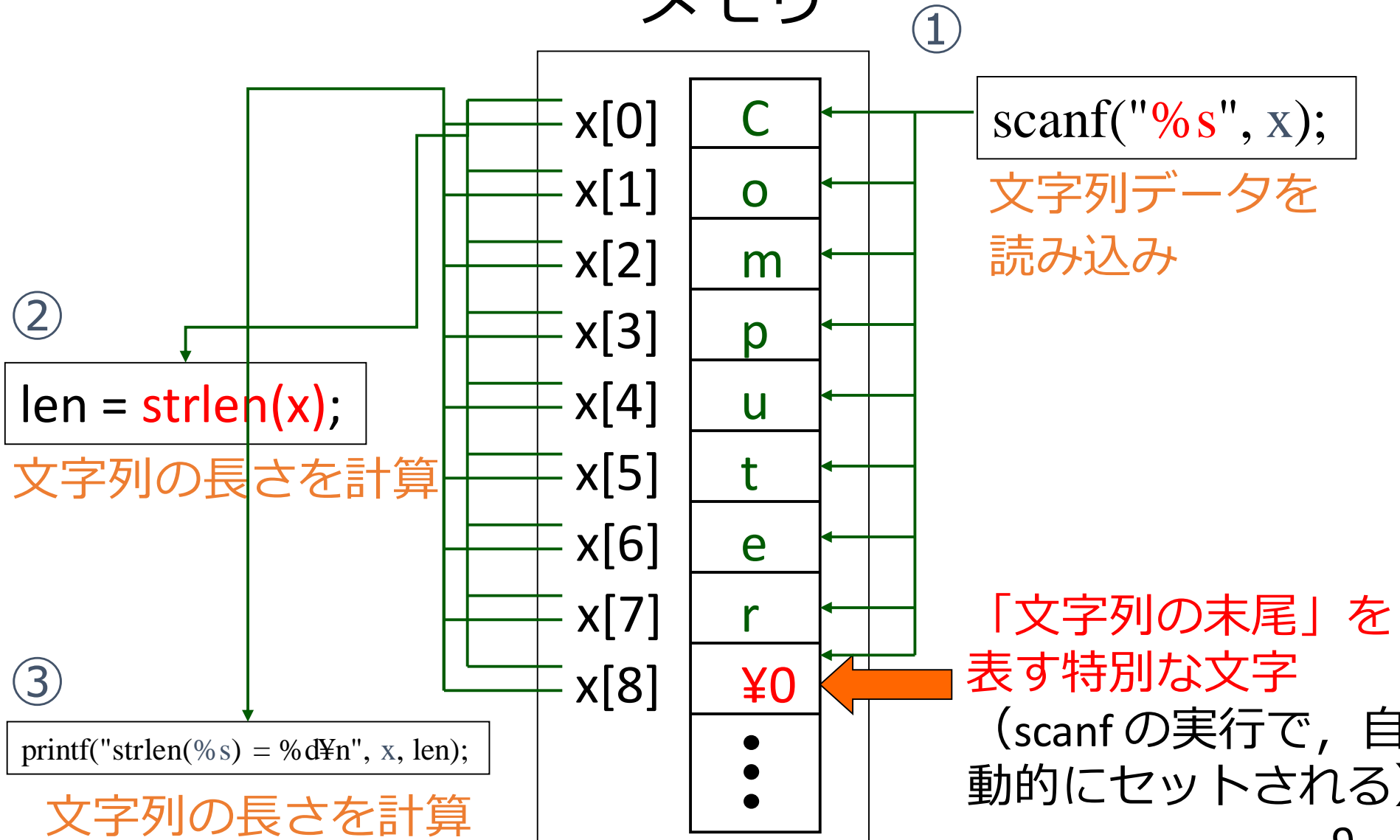
プログラム実行順



プログラムとデータ



メモリ





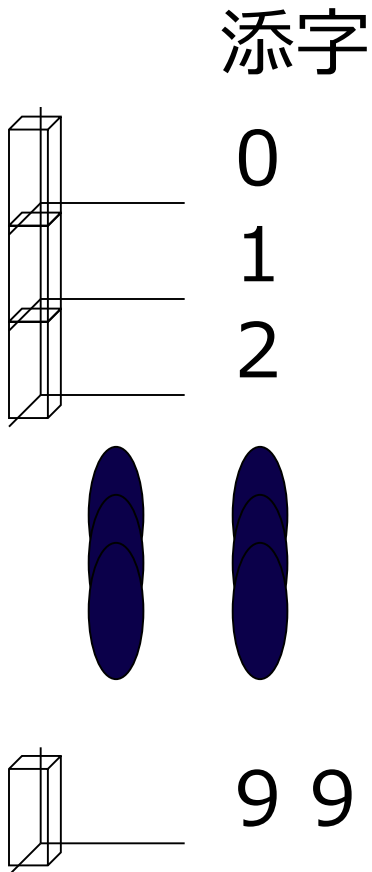
文字列の宣言

```
char x[100];
```

文字 名前 配列のサイズ
データ は x は 1 0 0

- 文字列データを扱うための最も簡単な方法は、文字の**配列**を使う方法である。
 - 配列には、名前とサイズがある
 - 配列を使うために、配列の使用をコンピュータに伝えること（宣言）が必要

文字列用の配列のサイズ



- 配列の添字は **0 から (サイズ - 1)**

例) `char x[100];` と宣言したら,
サイズは 1 0 0, 添字は 0 から 9 9

- 実際に使えるのは, **0 から (サイズ - 2) まで**

例) `char x[100];` と宣言したら,
実際に使えるのは, 0 から 9 8 まで
(最大 9 9 文字まで入る)

「文字列の末尾」を表す特別な文字を
入れるのに, 1 つ使われる



文字列の入力文

`scanf("%s", x);`
書式 読み込むべき変数名

- 「書式」と読み込むべき変数名を書く
 - 文字列の場合, 変数名の前には「&」を付けない



文字列の出力文

```
printf("strlen(%s) = %d¥n", x, len);
```

書式

表示すべき変数名

- 「書式」と表示すべき変数名を書く



課題 1 . 文字列の逆転

- 半角文字からなる文字列を読み込んで、逆転した文字列をつなげて、回文を作り、それを表示するプログラムを作りなさい。

例) 読み込んだ文字列

C	o	m	p	u	t	e	r	¥0
---	---	---	---	---	---	---	---	----

「文字列の末尾」を表す特別な文字
(scanf の実行で、自動的にセットされる)

回文

C	o	m	p	u	t	e	r	r	e	t	u	p	m	o	C	¥0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----

- ここでの文字列は、半角の「空白文字」を含まないものとする



課題 1 のヒント

- 添字を付けて、文字単位での読み書きを行う。

例) 1 文字目に, 5 文字目の文字をコピー

`x[0]=x[4];`

- 回文を作るときに, 文字列の末尾 (¥ 0) をセットすることを忘れない

例) 要素 `x[10]` に 「文字列の末尾」をセット

`x[10]='¥0';`



例題 2 . 文字列のコピー

- 文字列を読み込んで、文字列のコピーを行うプログラムを作る



例題 2 . 文字列のコピー

```
#include <stdio.h>
#include <string.h>
#pragma warning(disable:4996)
int main()
{
    char s1[80], s2[80];
    printf("s1=");
    scanf("%s", s1);
    strcpy(s2, s1);
    printf("s1=%s, s2=%s\n", s1, s2);
    return 0;
}
```



文字列のコピー

実行結果の例（１）

s1=test

s1=test, s2=test

実行結果の例（２）

s1=あいうえお

s1=あいうえお, s2=あいうえお



例題 3 . 文字列の連結

- 2つの文字列を読み込んで、文字列の連結を行うプログラムを作る



例題 3 . 文字列の連結

```
#include <stdio.h>
#include <string.h>
#pragma warning(disable:4996)
int main()
{
    char s1[80], s2[80];
    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    strcat(s1, s2);
    printf("s1=%s, s2=%s¥n", s1, s2);
    return 0;
}
```



文字列の連結

実行結果の例（１）

s1=abc

s2=def

s1=abcdef, s2=def

実行結果の例（２）

s1=あい

s2=うえお

s1=あいうえお, s2=うえお



例題 4 . 文字列の比較

- 2つの文字列を読み込んで、文字列の比較を行うプログラムを作る



```
#include <stdio.h>
#include <string.h>
#pragma warning(disable:4996)
int main()
{
    char s1[80], s2[80];
    int n;
    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    n = strcmp(s1, s2);
    if ( n < 0 ) {
        printf( "%s<%s¥n", s1, s2 );
    }
    else if ( n == 0 ) {
        printf( "%s==%s¥n", s1, s2 );
    }
    else if ( n > 0 ) {
        printf( "%s>%s¥n", s1, s2 );
    }
    return 0;
}
```

文字列の比較



実行結果の例（１）

```
s1=configure  
s2=control  
configure<control
```

実行結果の例（２）

```
s1=happy  
s2=angry  
happy>angry
```

実行結果の例（３）

```
s1=give  
s2=give  
give==give
```




例題 5 . 文字列の検索

- ¥を含むファイル名を，文字列として読み込んで，最後の¥以降の部分を表示するプログラムを作る.
 - ¥を含まない場合には，そのまま表示すること

文字列の検索



```
#include <stdio.h>
```

```
#include <string.h>
```

```
#pragma warning(disable:4996)
```

```
int main()
```

```
{  
    char name[80], base[80];  
    char *p, *search;  
    printf("name=");  
    scanf("%s", name);  
    search = name;  
    while( ( p = strstr(search, "¥¥") ) != NULL ) {  
        search = p + 1;  
    }  
    strcpy( base, search );  
    printf("name=%s, base=%s¥n", name, base);  
    return 0;  
}
```

NULLの意味

strstr関数では、検索
文字列が見つからない
ことを意味する





文字列の検索

実行結果の例

```
name=a:¥test¥file.txt
```

```
name=a:¥test¥file.txt, base=file.txt
```

文字列用のライブラリ関数



- 文字列用のライブラリ関数を使うときには、次の1行をプログラムに含めること

#include <string.h>

- コピー : **strcpy**
- 長さ取得 : **strlen**
- 連結 : **strcat**
- 比較 : **strcmp**
- 検索 : **strstr**

など



例題 6 . 文字列のメモリアドレス

- 文字列から, 各文字のメモリアドレスを表示する関数を作る



```
#include <stdio.h>
```

```
#include <string.h>
```

```
#pragma warning(disable:4996)
```

```
void printstring( char* s )
```

```
{
```

```
    int i;
```

```
    for (i=0; i<strlen(s); i++) {
```

```
        printf("address(%c) = %p\n", s[i], &s[i]);
```

```
    }
```

```
    return;
```

```
}
```

```
int main()
```

```
{
```

```
    char x[100];
```

```
    printf("string=");
```

```
    scanf("%s", x);
```

```
    printstring(x);
```

```
    return 0;
```

```
}
```

「%p」はメモリアドレス

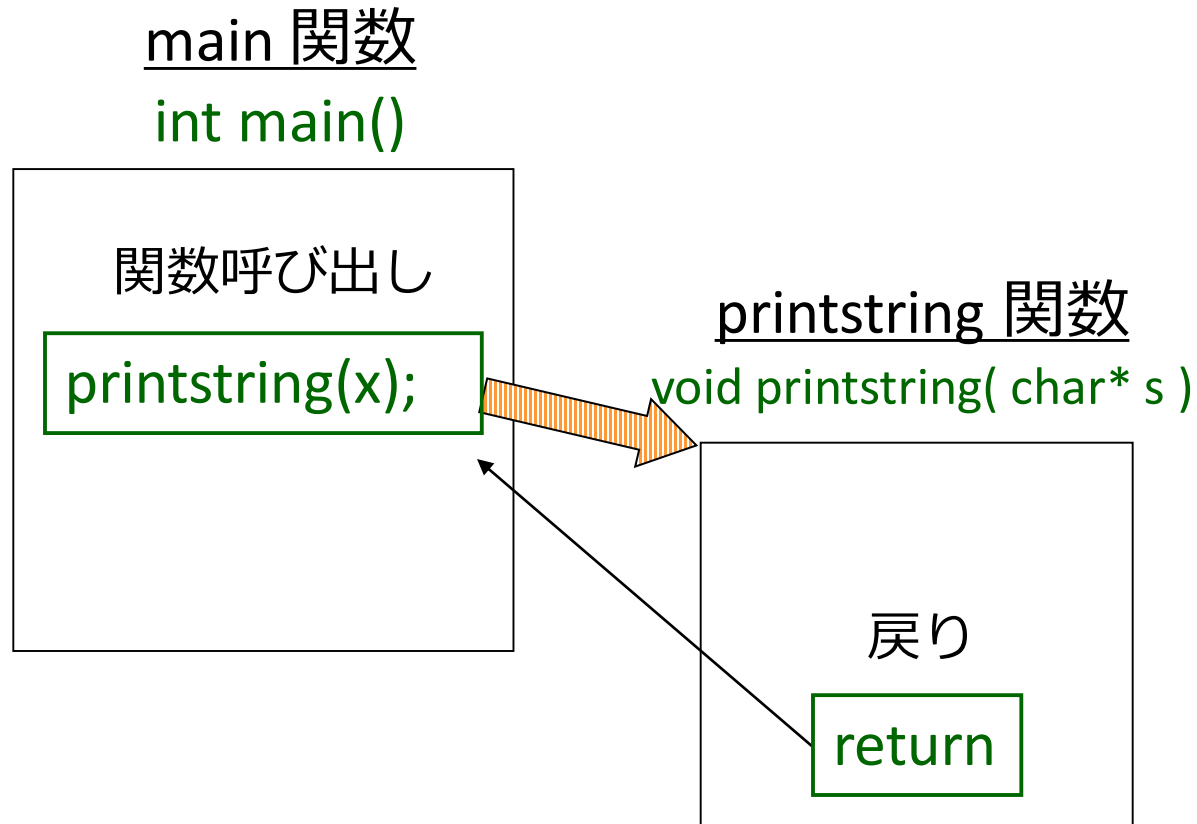
の表示

「&」はメモリアドレス
の取得

「s[i]」はi番目の文字
という意味

「%c」は1文字の表示

関数呼び出しの流れ





文字列のメモリアドレス

実行結果の例

表示された
メモリアドレス

address(C) =	ffbef440
address(o) =	ffbef441
address(m) =	ffbef442
address(p) =	ffbef443
address(u) =	ffbef444
address(t) =	ffbef445
address(e) =	ffbef446
address(r) =	ffbef447

x[0]	c	ffbef440
x[1]	o	ffbef441
x[2]	m	ffbef442
x[3]	p	ffbef443
x[4]	u	ffbef444
x[5]	t	ffbef445
x[6]	e	ffbef446
x[7]	r	ffbef447
x[8]	¥0	
	⋮	

「文字列の末尾」を表す特別な文字
(scanfの実行で、自動的にセットされる)



関数への文字列の受け渡し

- 呼び出し側

- 変数名を書いて、文字列（文字列を格納した配列）の先頭メモリアドレスを、関数に渡す

例) `printstring(x);`

↑
文字列 x の先頭メモリアドレス
(&x[0] の省略形)

- 関数側

- 文字列を受け取る（つまり、文字列の先頭メモリアドレス）ことを宣言しておく

`void printstring(char* s)`

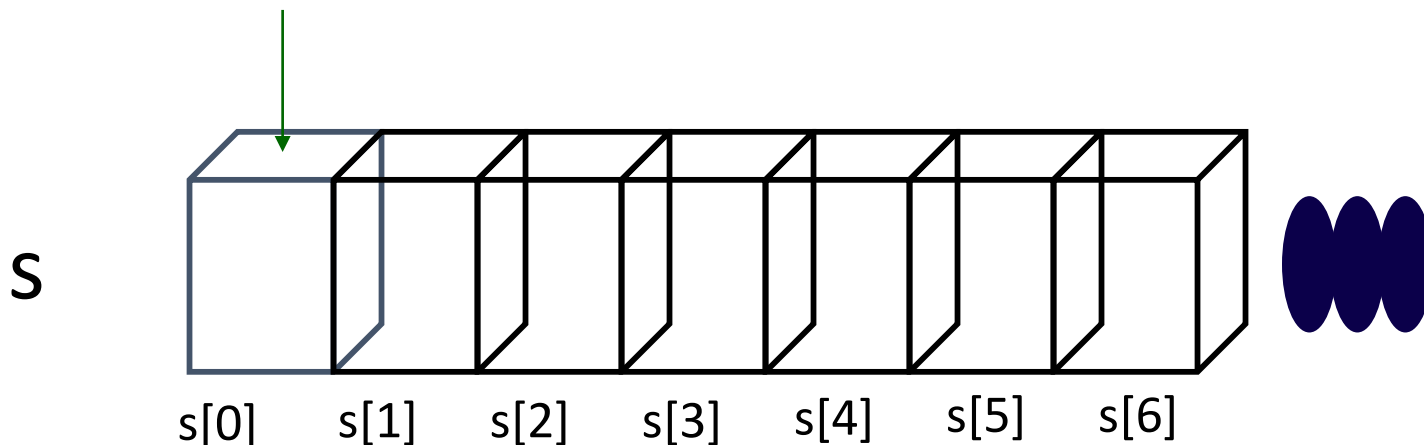
↑
「文字列の先頭メモリアドレス」
を受け取って、s として使うという

文字列とポインタ



プログラム例 : `printstring(s);`

文字列の先頭



- プログラム中に文字列の名前（つまり、文字列を格納した配列名を単独で書くと、文字列の先頭メモリアドレスという意味



課題 2 . 文字列を扱う関数

- 3つの文字列を受け取って、辞書順に表示する関数を作成しなさい。同時に、この関数を使う main 関数を作成し、正しく動作することを確認すること。
 - strcmp 関数を使用すること

例題 7 . 曜日の表示



- 数字を読み込んで，曜日を表示するプログラムを作成する

0 Sun

1 Mon

2 Tue

3 Wed

4 Thr

5 Fri

6 Sat

曜日の表示



```
#include <stdio.h>
#pragma warning(disable:4996)
void print_youbi( int youbi )
{
    char x[7][4] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
    "Sat"};
    printf( "%s\n", x[youbi] );
    return;
}
int main()
{
    int n;
    printf("youbi=");
    scanf("%d", &n);
    print_youbi( n );
    return 0;
}
```



曜日の表示

実行結果の例

youbi=3

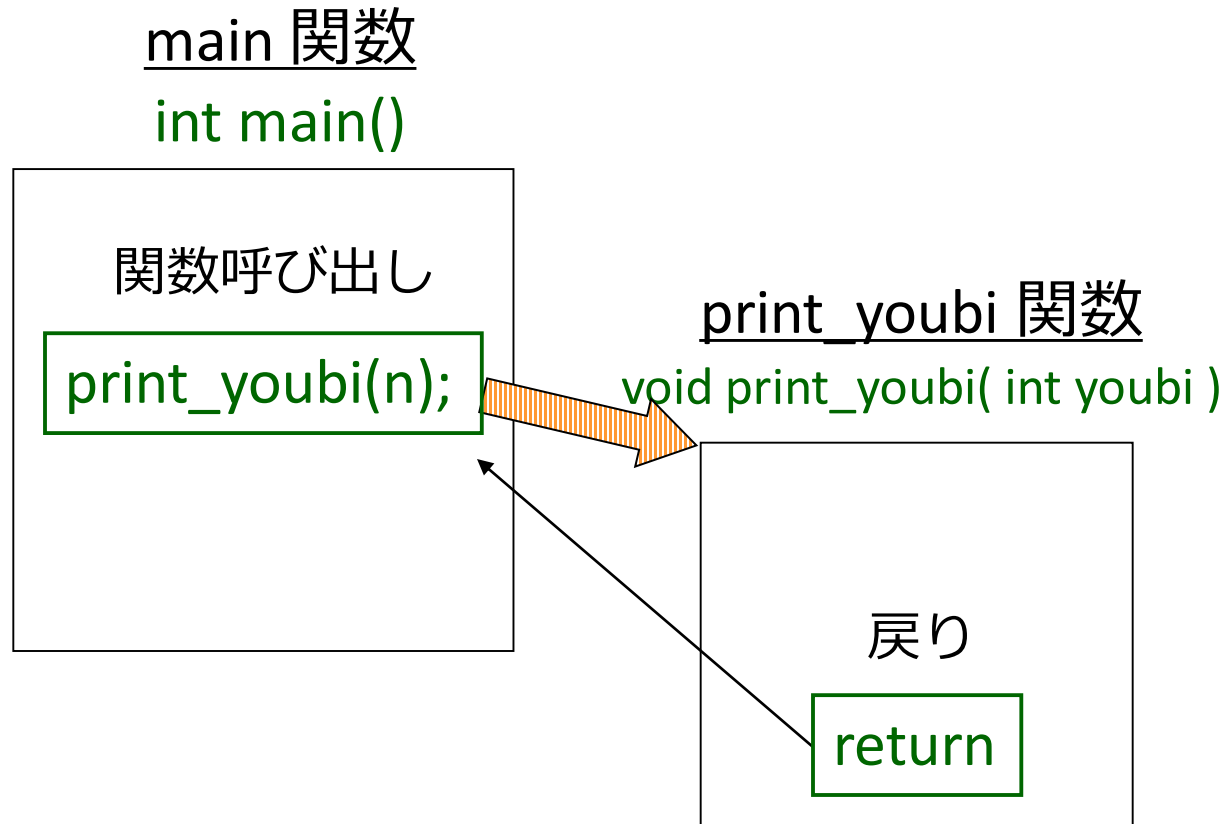
Wed

実行結果の例（２）

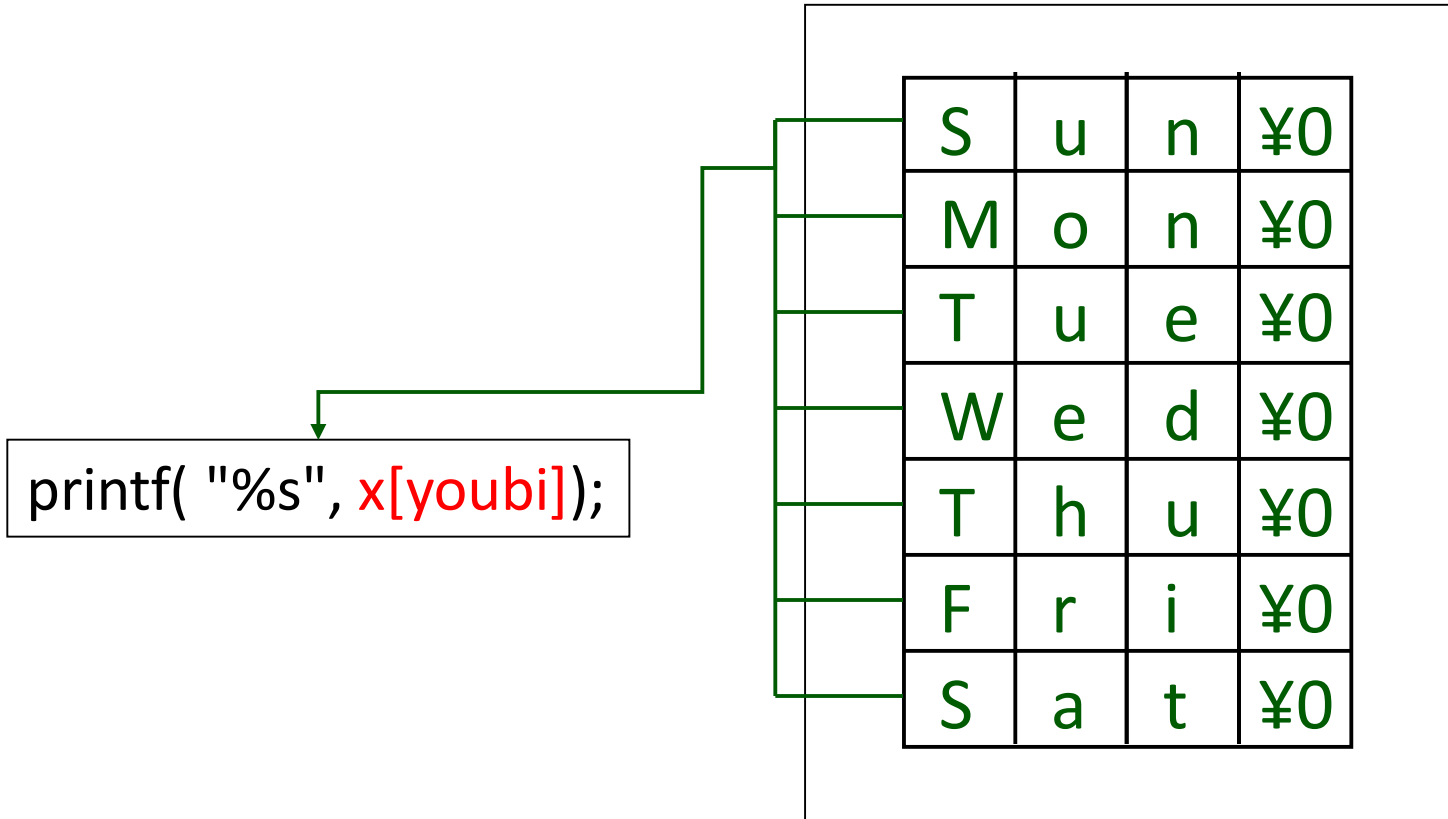
youbi=0

Sun

関数呼び出しの流れ



メモリ



課題 3 . 小数部分の抜き出し



- 「小数付きの数」を読み込んで、小数部分のみを抜き出して表示するプログラムを作りなさい.

例) “18.256” の小数部分は 256

- 「小数付きの数」を文字列データとして読み込むこと