

# 第14回 文字列処理

アルゴリズムとデータ構造ならびに同演習

柏原 昭博

akihiro.kashihara@inf.uec.ac.jp

# 講義内容

---

- 第11回目 ヒープ
- 第12回目 ソート（整列）
- 第13回目 クイックソート・その他のソート
- **第14回目 文字列処理（基本）**
- 第15回目 文字列処理（配列処理・文字探索）

# 文字と文字列

---

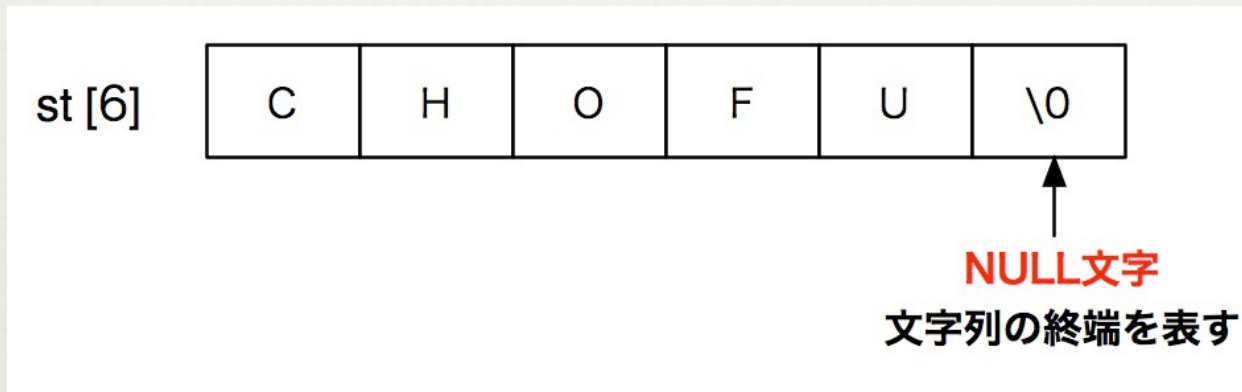
## □ 文字の表現

- 文字変数で表現する
- `char c = 'A';` // ' ' で囲む
- 計算機内部では、文字コード番号として表現
  - ASCIIコードでは、Aは  $65_{(10)}$ 、Zは  $90_{(10)}$  に対応
  - コード表の例：<http://ja.wikipedia.org/wiki/ASCII>
  - 演習14-1
- 文字に対する計算
  - コード番号を利用して計算できる
    - `c = 'A' + 7;` (cは'H'となる)
  - 演習14-2

# 文字と文字列

## □ 文字列の表現

- 配列表現：文字の配列で表現



- 初期化

```
char st [6]={ 'C', 'H', 'O', 'F', 'U', '\0' };
```

```
char st []="CHOFU" // “内の列を文字列リテラルと呼ぶ
```

※宣言時にしか使えない！

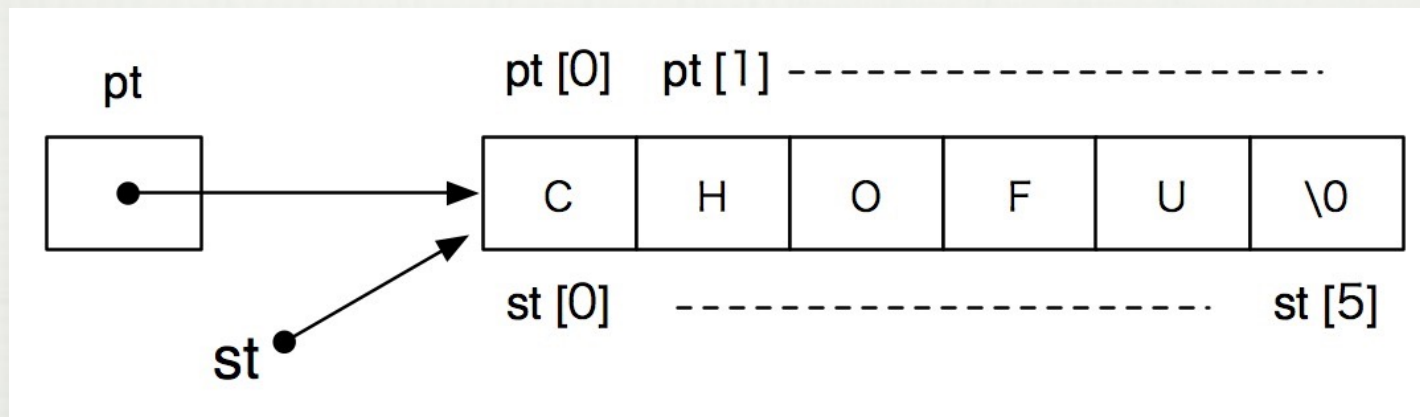
※一つ一つ文字要素を代入するしかない



# 文字と文字列

## □ 文字列の表現

□ ポインタ表現：char \*pt = “CHOFU”;



□ 文字列を表す配列名stもポインタと同じ役割を果たす

# 演習14-1&14-2

---

- 入力した文字の文字コードを出力するプログラムを作りなさい。  
(sample14-1.c 参照) 入力する文字を色々変えて, 文字コード表と合っているかどうか確認しなさい.
- 'A' + 7が'H'となることを確認するプログラムを作りなさい。  
(sample14-2.c参照)

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

sample14-2.c

```
int main (void)
{
    char c='A';
    int i;

    printf("%c + ", c);
    scanf ("%d", &i);

    c= c+i;
    printf("%c \n", c);
    printf("%d \n", c);

    return 1;
}
```

# 演習14-3

---

- アルファベット大文字2文字の差を計算するプログラムを作りなさい. (sample14-3.c 参照)



```
#include <stdio.h>

int main (void)
{
    char c1='A';
    char c2='Z';
    int tmp;

    tmp = c2 - c1;

    printf("Z-A:%d", tmp);
    printf("\n");

    return 1;
}
```

# 演習14-3

---

- sample14-4.cを参考に、文字列を初期化する方法を理解しなさい。色々と初期値を変えて確認しなさい。

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

sample14-4.c

```
int main (void)
```

```
{
```

```
    char st[10];
```

```
    char st1[10]="CH0FU-L";
```

```
    char st2[10]={ 'C', 'H', '0', 'F', 'U', '-', 'R', '\0' };
```

```
    st[0]= 'C';
```

```
    st[1]= 'H';
```

```
    st[2]= '0';
```

```
    st[3]= 'F';
```

```
    st[4]= 'U';
```

```
    st[5]= '-';
```

```
    st[6]= 'D';
```

```
    printf("St: %s \n", st);
```

```
    printf("St1: %s \n", st1);
```

```
    printf("St2: %s \n", st2);
```

```
    return 1;
```

```
}
```

# 課題14-1

---

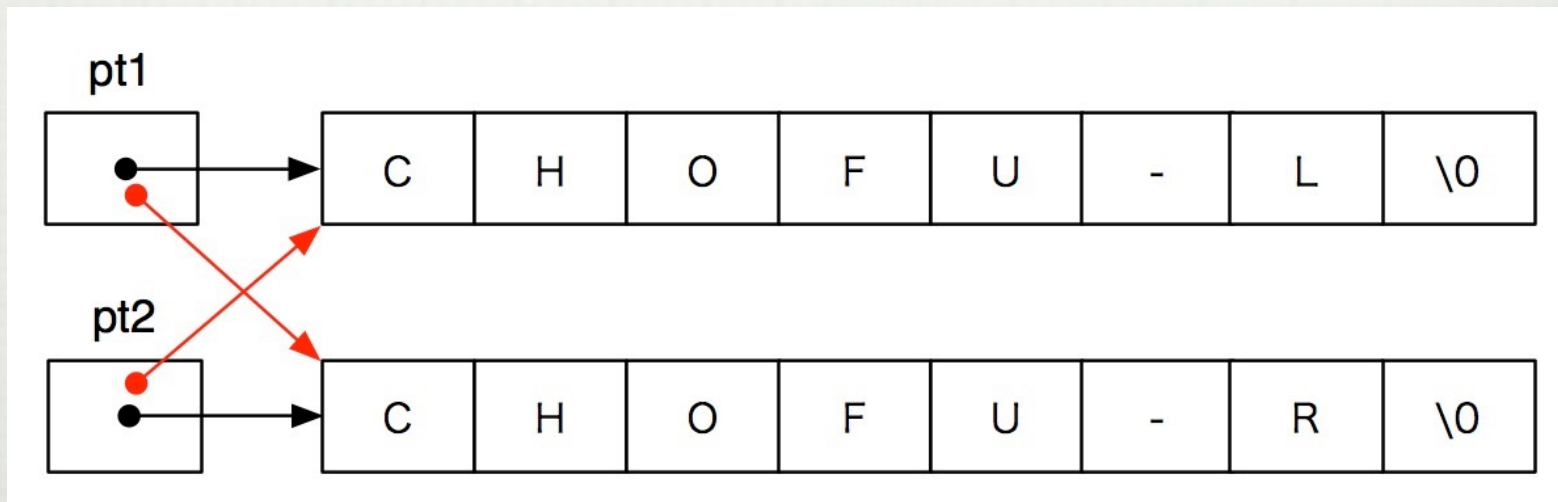
- ‘A’と、入力した任意のアルファベット大文字1文字の差を計算するプログラムを作りなさい。(ex14-1.c)  
なお、文字の入力は、文字列入力に用いるscanfを使うこと。

```
char str [2];  
scanf ("%s", str);
```



# 演習14-4

- 以下のように、2つの文字列へのポインタpt1とpt2を交換して、それぞれのポインタが指す文字列を出力するプログラムを書きなさい。 (sample14-5.c)



## 復習：ポインタのポインタ

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

```
int main() {  
    char univ[] = "University of Electro-Communications";  
    char *univ_p;  
    char **univ_pp; /*ポインタのポインタ*/  
  
    univ_p = univ;  
    univ_pp = &univ_p;  
  
    printf("%x\n" , univ);          /*配列univの先頭アドレス*/  
    printf("%s\n\n" , univ);        /*University of Electro-Communications*/  
  
    printf("%x\n" , &univ_p);        /*ポインタアドレス*/  
    printf("%x\n" , univ_p);         /*配列univの先頭アドレス*/  
    printf("%s\n\n" , univ_p);       /*University of Electro-Communications*/  
  
    printf("%x\n" , univ_pp);         /*ポインタアドレス*/  
    printf("%x\n" , *univ_pp);        /*配列univの先頭アドレス*/  
    printf("%s", *univ_pp);           /*University of Electro-Communications*/  
  
    return 0;  
}
```

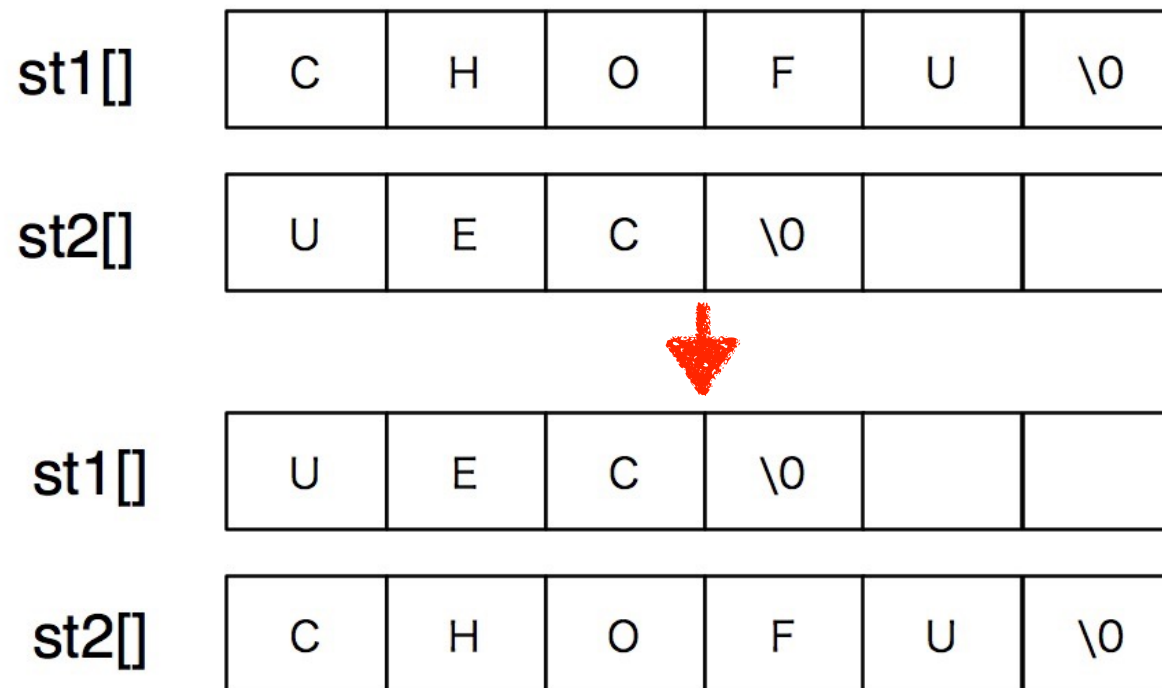
## sample14-5.c

```
#include <stdio.h>
void swap_pt(char **, char **);
int main (void)
{
    char *pt1="CH0FU-L";
    char *pt2="CH0FU-R";
    printf("pt1: %s \n", pt1);
    printf("pt2: %s \n", pt2);
    swap_pt (&pt1, &pt2);
    printf("After swap \n");
    printf("pt1: %s \n", pt1);
    printf("pt2: %s \n", pt2);
    return 1;
}
```

```
void swap_pt (char **x, char **y)
{
    char *tmp = *x;
    *x = *y;
    *y = tmp;
}
```

# 課題14-2

- 2つの配列が表現する文字列を、配列の要素ごとに交換する関数  
swap\_str (char \*st1, char \*st2)  
を書きなさい. (ex14-2.c)

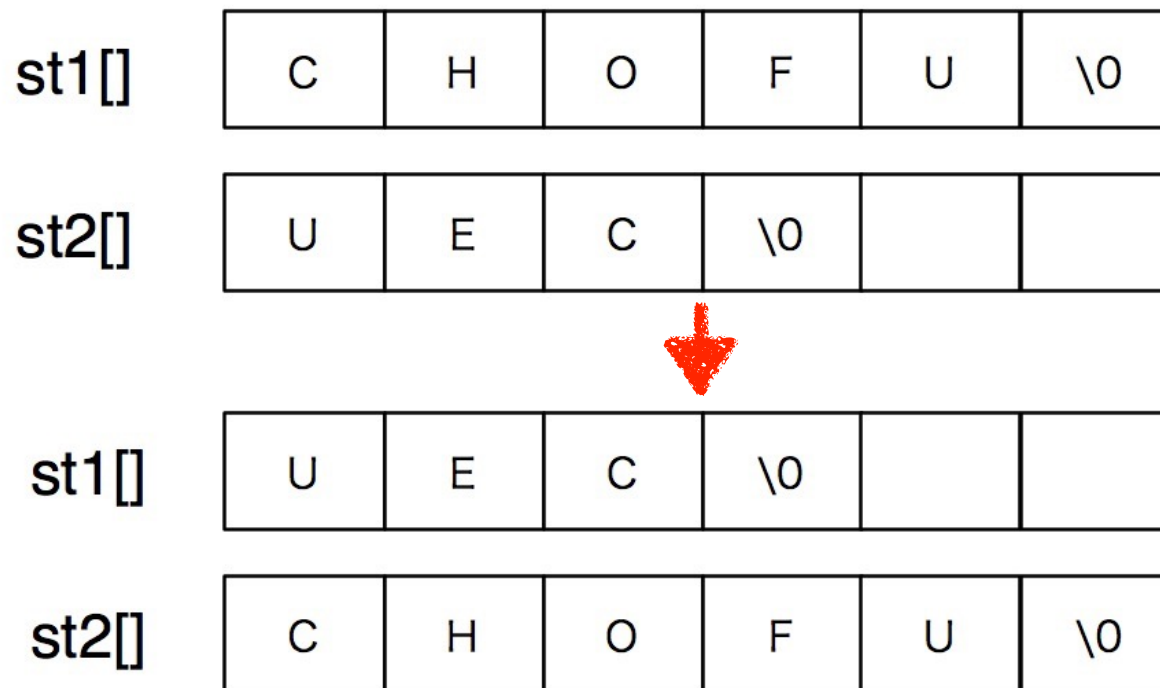




# 課題14-2

## □ ヒント

2つの文字列へのポインタをインクリメントしながら、ポインタが指す要素がともにNULL文字になるまで、要素を交換する。



# 文字列の基本処理

---

- 数値変換
- 文字列の長さ（文字数）をはかる
- 文字の探索
- 文字列の比較
- 文字列のコピー

# 文字列の数値変換

---

- atoi関数
  - 文字列を整数に変換する  
変数 = atoi (文字列配列名);
  - 数値以外の文字列は0に変換
  - #include <stdlib.h>

# 文字列の数値変換

---

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

```
int main (void)
{
    char str[]="145";
    char str1[]="-145";
    int num=atoi (str);
    int num1=atoi(str1);

    printf("str: %d \n", num);
    printf("str1: %d \n", num1);

    return 1;
}
```



sample14-6.c




# 文字列の長さをはかる

---

- 文字列の先頭からNULL文字を探索する

0	1	2	3	4	5
C	H	O	F	U	\0



# 演習14-5

---

- scanfで入力した文字列の長さを求める関数  
`str_len(const char *s)`  
を書きなさい。 (sample14-7.c)

## sample14-7.c

```
#include <stdio.h>
int str_len (const char *);
int main (void)
{
    char str[100];
    printf("String: \n");
    scanf ("%s", str);
    printf("Length of the string: %d \n", str_len (str));
    return 1;
}
```

```
int str_len (const char *st)
{
    int leng=0;
    while (st[leng])
        leng++;
    return leng;
}
```

# 演習14-5

---

- strlen関数を使って文字列の長さを求めるプログラムを作りなさい. (sample14-7-1.c)

**size\_t strlen (const char \*s);**

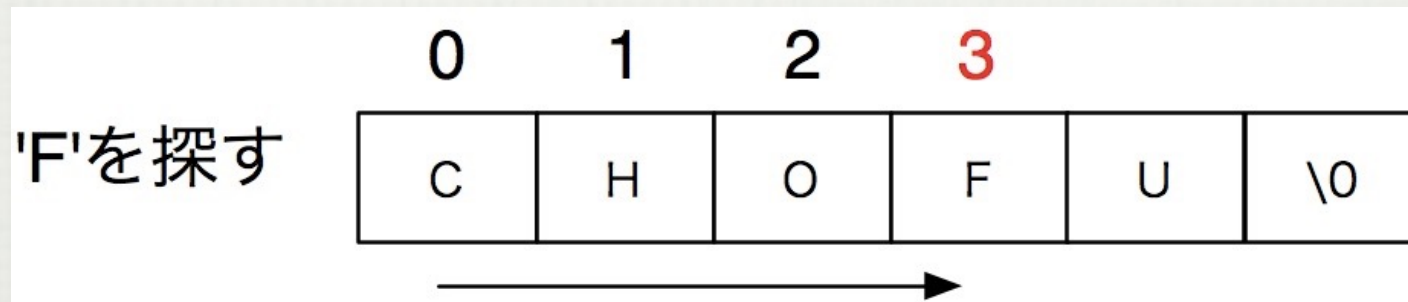
※#include <string.h>が必要



# 文字列から文字探索

---

- 文字列の先頭から任意の文字を探索する  
成功すれば文字の配列位置を返す.  
失敗すれば-1を返す. (sample14-8.c参照)



```
#include <stdio.h>
#include <string.h>
int main (void)
{
    char str[100];
    char temp_str[2];
    int ch;
    char *i;
    printf("文字列: \n");
    scanf ("%s", str);
    printf("探索文字: \n");
    scanf ("%s", temp_str);
    ch = temp_str[0];
    i = str_chr (str, ch);
    if (*i != '\0')
        printf("文字%cは%sの%d番目に存在 \n", ch, str, (int)( i-str+1 ));
    else
        printf("文字%cは%sに存在しない \n", ch, str);
    return 1;
}

char *str_chr (const char *str, int c)
{
    c=(char)c;
    while (*str!=c && *str!='\0')
        str++;
    return (char *)str;
}
```

# 演習14-6

---

- sample14-8.cを参考に、文字列の末尾から任意の文字を探索する

成功すれば文字の配列位置を返す.

失敗すれば-1を返す.

(sample14-9.c)

# 演習14-6

---

- strchr, strrchr関数を使ってsample14-8.c, sample14-9.cを書き換えなさい。 (sample14-8-1.c, sample14-9-1.c)

**char \*strchr (const char \*s, int c);**

sが指す文字列の最も先頭側に出現する(char型に変換した)cを探す。探した文字へのポインタを返す。なければ空ポインタを返す。

**char \*strrchr (const char \*s, int c);**

sが指す文字列の最も末尾側に出現する(char型に変換した)cを探す。探した文字へのポインタを返す。なければ空ポインタを返す。

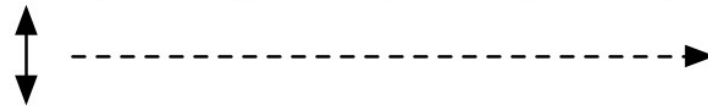
※#include <string.h>が必要



# 文字列の比較

- 2つの文字列（配列1，配列2）を先頭から比較し，  
すべての文字が同じ場合は0を返す  
配列1 > 配列2の場合は正の整数  
配列1 < 配列2の場合は負の整数を返す

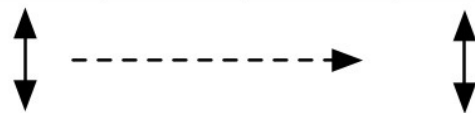
C	H	O	F	U	\0
---	---	---	---	---	----



C	H	O	F	U	\0
---	---	---	---	---	----

一致：0

C	H	O	F	U	\0
---	---	---	---	---	----



C	H	O	S	I	\0
---	---	---	---	---	----

不一致：負

# 演習14-7

---

- 入力した2つの文字配列を先頭要素から比較する関数を書きなさい。  
(sample14-10.c)

```
int str_cmp (const char *str1, char *str2)
{
    while (*str1 == *str2) {
        if ( *str1 == '\0'
            return 0; //文字列が等しい場合

        str1++; str2++;
    }
    return (unsigned char)*str1 - (unsigned char)*str2;
    //等しくない場合は文字の差を返す
}
```

# 演習14-7

---

- strcmp関数を使ってsample14-10.cを書き換えなさい.  
(sample14-10-1.c)

**int strcmp (const char \*s1, const char \*s2);**

s1,s2が指す文字列の大小関係（先頭から順に1文字ずつunsigned char型として比較する。等しければ0, s1が大きければ正, s2が大きければ負の整数を返す。

※#include <string.h>が必要



# 課題14-3

---

- strncmp関数は、第3引数で指定された数だけ、2つの配列の先頭要素から比較する。返り値はstrcmp関数と同様である。このstrncmpと同等の機能を果たす関数strn\_cmpを作りなさい。

```
int strncmp (const char *s1, const char*s2, size_t n);
```

```
※#include <string.h>
```

# 課題14-4

---

- strcpy関数と同等の機能を果たす関数str\_cpyを作りなさい.

**char \*strcpy (char \*dest, const char \*src);**

dest[]: コピー先配列, src[]: コピー元配列 (※\0で終端)  
コピー先配列の長さ $\geq$ コピー元配列の長さ

※#include <string.h>

# レポート提出

---

- ☐ 課題14-1～14-4
- ☐ 提出期限：7月29日 0:00（7月28日まで）
- ☐ 提出先：Webclass