

2020回 計算機プログラミング演習

第9回

文字列と文字列配列の応用

文字列と文字列操作

- 文字列と文字列操作

- char 型 (一文字)

- 文字列 (char 型の配列)

- 文字列の代入と演算

- strcpy() ※ ☆
 - sprintf() ※
 - gets() ※
 - strlen() ☆
 - strcmp() ※ ☆

```
※ #define _CRT_SECURE_NO_WARNINGS 1
☆ #include <string.h>
```

- 文字列の配列 (2次元以上の配列定義)

文字列

- char 型で扱われる
 - char 型とは本来 8bit (1 byte) の整数
(-128~127, unsigned 0~255)
 - (日本語と違って) 英語に使われる文字は記号を入れても100種類程度
 - 文字(例えば 'A')に番号(65) を付けて規格化(ASCII, JIS など)されている。
 - よって char 型は「文字の入れ物」として使われることが多い。
- char 型の配列を使う
 - 文字一つ('A')だけではあまり意味をなさない。
 - 意味を表すためには複数の文字を連ねる。-> 配列
 - 扱うのが単語や文章になり、必要な長さ(配列の大きさ)がまちまちである。
 - char (一文字) 'A', 文字列 "ABC"

char 型について考える(1)

```
/* char_test.c : 文字コード 65 ('A') と 97 ('a') を表示 */
#include <stdio.h>

int main( void ){

    char la, sa, lz;

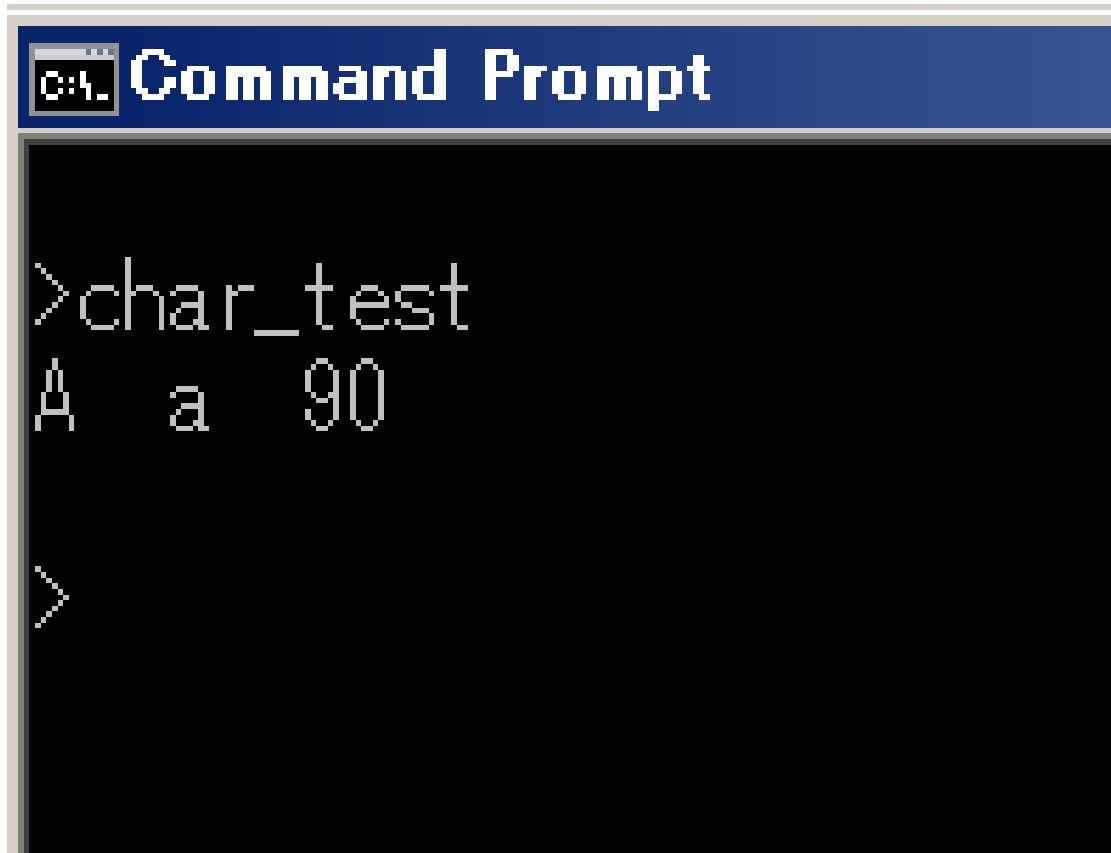
    la = 65;
    sa = 97;
    lz = 'Z';

    printf("%c %c %d¥n", la, sa, lz );

    return 0;
}
```

実行結果

- `%c` で文字に、`%d` で数字として表示されている。



```
C:\_ Command Prompt

>char_test
A  a  90

>
```

char 型について考える(2)

```
/* char_test.c */

#include <stdio.h>

int main( void ){

    char ch;

    ch = '電';
    printf("%c %d\n", ch, ch);

    return 0;
}
```

A terminal window showing the output of the program. The first column displays the character '電' (denwa) and the second column displays the integer value 100, which is the ASCII code for '電'.

日本語(漢字コード)は2バイトコードなので char(1バイト)では収まらない。(一文字としては使えない。)

文字列 (char 型の配列)(1)

- 文字列に使う変数の定義
 - 扱う文字数+1 以上の大きさの配列を使う。
 - 日本語(漢字コード)を使う場合は (文字数)*2 + 1 以上
 - +1 には文字の終了コード($\backslash 0$)が必要なため。

```
/* char_test.c : 文字列の定義と初期値代入 */  
#include <stdio.h>
```

```
int main( void ) {
```

```
    char date[9] = "Thursday";  
    char jdate[7] = "木曜日";
```

文字列の初期値代入

```
    printf(" %s は %s です\n", jdate, date);
```

```
    return 0;
```

木曜日 は Thursday です

```
}
```

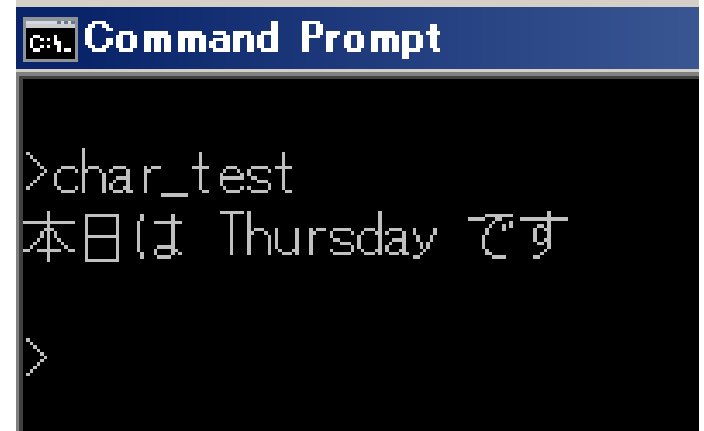
日本語(漢字コード)は char の配列で表示可

文字列 (char 型の配列) (3)

- 文字列を初期値ではなく、代入する
 - 一文字ずつ代入する (面倒くさいが...)

```
char date[9] ;  
  
date[0] = 'T';  
date[1] = 'h';  
date[2] = 'u';  
date[3] = 'r';  
date[4] = 's';  
date[5] = 'd';  
date[6] = 'a';  
date[7] = 'y';  
date[8] = '¥0';
```

```
printf("今日は %s です¥n", date);
```



```
cmd Command Prompt  
>char_test  
今日は Thursday です  
>
```

問題なく、ビルド、実行できる。

文字列 (char 型の配列) (4)

- 文字列を初期値ではなく、代入する
 - 最後の終了コード(¥0) を削除してみる。(コメントにして実行しない)

```
char date[9] ;

date[0] = 'T';
date[1] = 'h';
date[2] = 'u';
date[3] = 'r';
date[4] = 's';
date[5] = 'd';
date[6] = 'a';
date[7] = 'y';
// date[8] = '¥0'; ← コメントアウト(実行しない)

printf("本日は %s です¥n", date);
```



ビルド、実行できるが、終了コード(¥0) がないので余分な文字を表示する。

文字列 (char 型の配列) (5)

- strcpy () 関数を使用する

- #define _CRT_SECURE_NO_WARNINGS 1

- #include <string.h> を追加

```
/* char_test.c : 文字列の定義と文字列の代入 */
```

```
#define _CRT_SECURE_NO_WARNINGS 1
```

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

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

文字列に関するヘッダーファイルを
インクルード

```
int main( void )
```

```
{
```

```
    char date[9] ;
```

```
    strcpy( date, "Thursday" );
```

配列に文字列をコピー ただし¥0もコ
ピーすることに注意！

```
    printf( "今日は %s です¥n", date );
```

```
    return 0;
```

```
}
```

今日は Thursday です

文字列 (char 型の配列) (6)

- sprintf () 関数を使用する

```
#define _CRT_SECURE_NO_WARNINGS 1
```

```
/* char_test.c : 文字列の定義と文字列の代入 */
```

```
#define _CRT_SECURE_NO_WARNINGS 1  
#include <stdio.h>
```

```
int main( void )  
{
```

```
    char date[9] ;
```

指定したフォーマットで配列に文字列を作成
確保する要素数は¥0に注意

```
    sprintf( date, "Thursday" );
```

```
    printf( "本日は %s です¥n", date );  
    return 0;
```

```
}
```

本日は Thursday です

文字列 (char 型の配列)(7)

- sprintf() 関数はstrcpy() より簡単
 - 使い慣れた printf() と同様な書き方ができる

```
/* char_test.c : 文字列の定義と文字列の代入 */
```

```
#define _CRT_SECURE_NO_WARNINGS 1
```

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

```
int main( void )
```

```
{
```

```
    char date[10], mess[100] ;
```

```
    int day = 23 ;
```

```
    sprintf(date, "Thursday");
```

```
    sprintf(mess,
```

本日23日はThursdayです。

```
        "本日%d日は%sです。¥n", day, date);
```

```
    printf( mess ) ;
```

```
    return 0;
```

```
}
```

指定したフォーマットで配列に文字列を作成 確保する要素数は¥0に注意

gets() , strlen() 関数について

```
gets (char str) ;
```

- 標準入力(キーボード)から改行文字で終わる文字列を読み込む
- 文字列を配列strに格納
- 配列strに格納するときに改行文字は'¥0'に置き換わる

```
strlen (char str) ;
```

- 文字列strの長さを返す
- 配列strの終了コード¥0の前までの文字数を返す

gets() , strlen()

```
/* char_test.c : 文字列の定義と文字列の代入 */
#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>
#include <string.h>

int main( void ){
    char s_name[30];
    int length;

    printf("名前を入力してください ");
    gets(s_name);
    length = strlen(s_name);
    printf("私の名前は%sです。 %d 文字です。¥n",
           s_name, length);
    return 0;
}
```



名前を入力してください宮大太郎
私の名前は宮大太郎です。8文字です。

2バイト文字
だと倍になる

文字列の比較 strcmp()


```
strcmp( char str1, char str2 )
```

- str1とstr2が等しいならば 0 を返す
- str1がstr2より大きい(str1>str2)ならば正の値を返す
- str1がstr2より小さい(str1<str2)ならば負の値を返す


文字列の比較 strcmp()

```
/* char_test.c : 文字列の定義と文字列の代入 */
#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>
#include <string.h>
int main( void ) {
    char s_name[30];

    printf("文字を入力してください ");
    gets(s_name);
    if ( strcmp( s_name, "abc" ) == 0 ) {
        printf("OK¥n") ;
    } else {
        printf("NG¥n" );
    }
    return 0;
}
```



A terminal window showing the execution of the program. The prompt "文字列を入力してください" is followed by the input "abc". The output is "OK", indicating a successful comparison.



A terminal window showing the execution of the program. The prompt "文字列を入力してください" is followed by the input "abcdef". The output is "NG", indicating a failed comparison.

第3回課題に関する説明

整列アルゴリズム(ソートプログラム)

多くの数値のデータがある場合, ある規則(昇順, 降順等)に従いデータを整列すること

例) テストの点数が良い順に並べる, 学籍番号順に並べる等

ソートのアルゴリズムは複数存在するが, 本講義では以下を紹介する

- バブルソート
- 選択ソート(課題1)
- 挿入ソート(課題2)

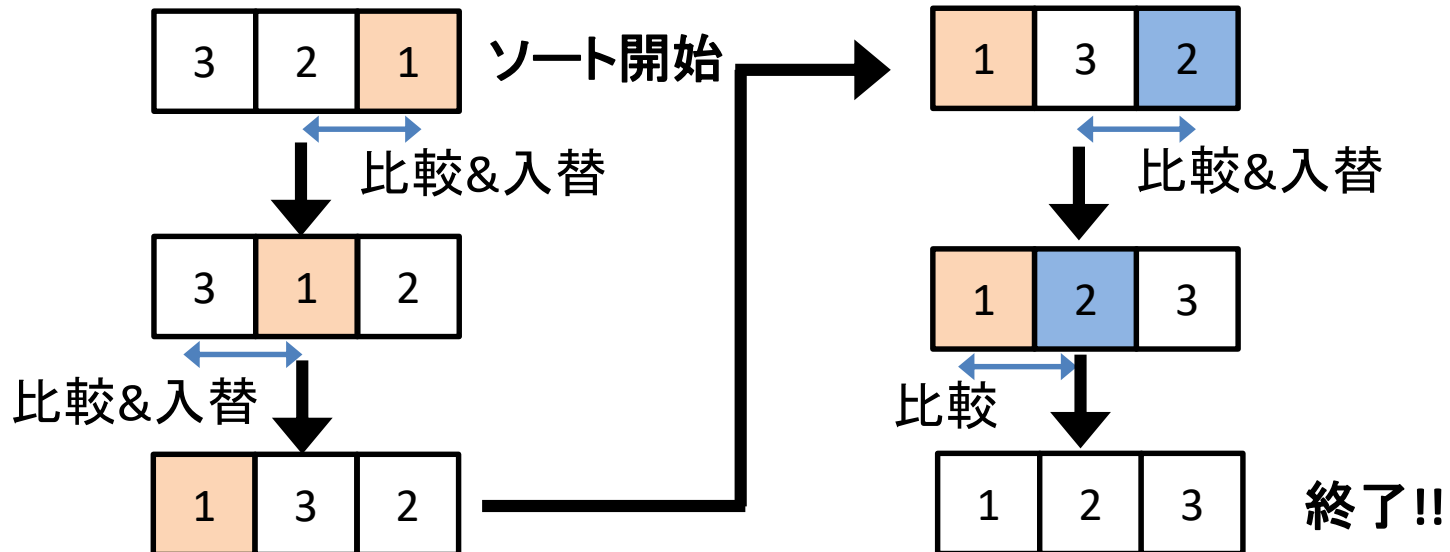
他にもマージソートがある

バブルソートとは

最も簡単な整列アルゴリズムであり、隣り合う配列の要素を比較し、必要であれば入れ替える



- ① 右側から隣り合う2つの要素を比較し、小さい数が左になるように入れ替える
- ② ①を繰り返す(ただし、比較の際に既に左に小さい数がある場合は入れ替え不要)



演習 (バブルソート)

左から「2, 5, 6, 1, 4, 9, 8, 7, 0, 3」と数字が並んでいる.
これを左から昇順「0, 1, 2, 3, 4, 5, 6, 7, 8, 9」となる**バブルソートプログラム**を作成し, その実行結果をコマンドプロンプト上に出力せよ.

```
Bubble sort の結果は  
0,1,2,3,4,5,6,7,8,9,
```

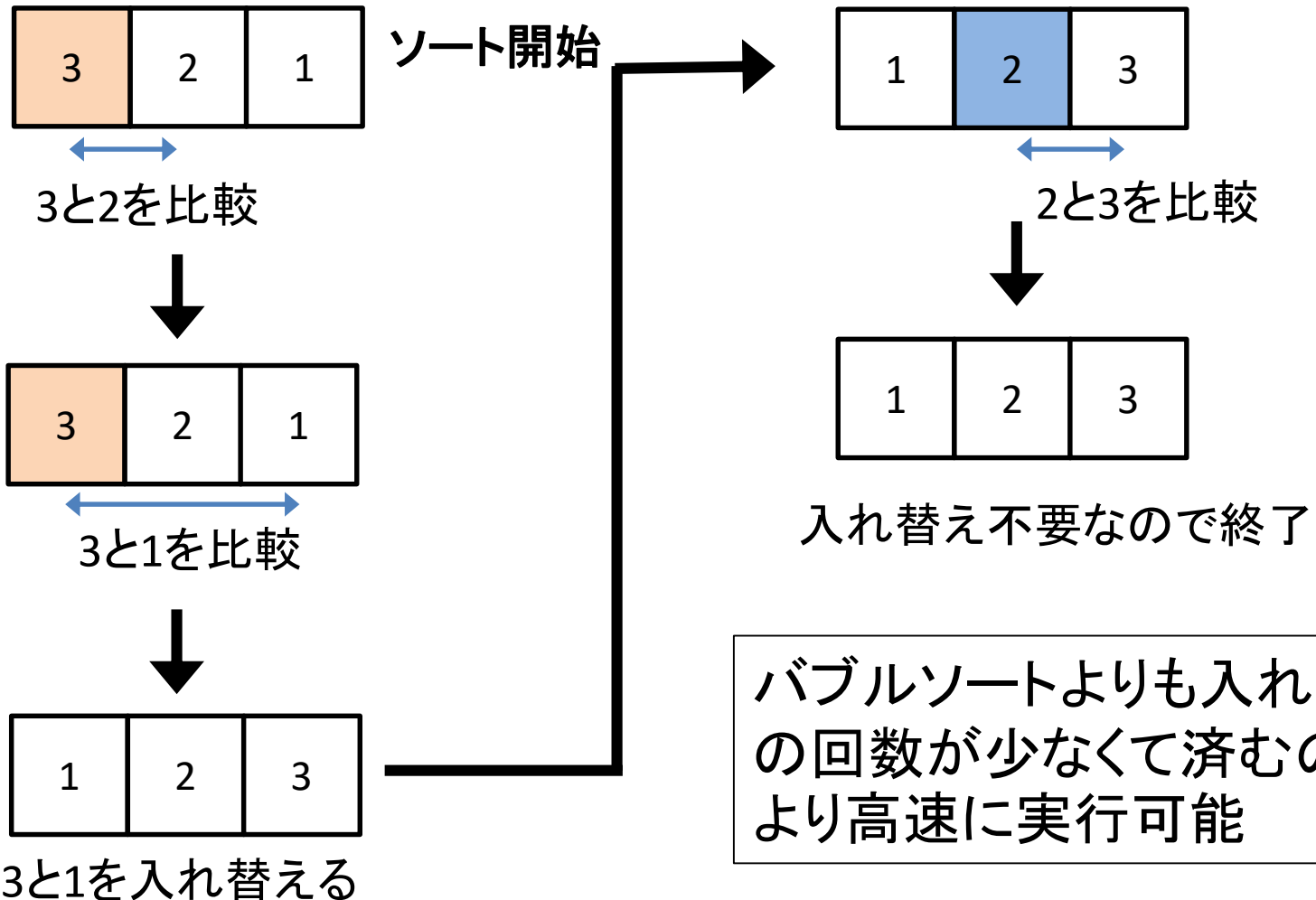
バブルソート(プログラム例)

```
1  /* Bubble_sort.c */
2
3  #include <stdio.h>
4
5  int main(void) {
6
7      //配列の要素数
8      int idx = 9;
9
10     //配列の確保
11     int num[] = {2,5,6,1,4,9,8,7,0,3};
12
13     //一時的な退避用の領域確保
14     int temp_space = 0;
15
16     for (int i = 1; i < idx; i++) {
17         for (int j = 0; j < idx; j++) {
18             if (num[idx - j - 1] > num[idx - j]) {
19                 //要素交換の為に退避
20                 temp_space = num[idx - j];
21                 //要素の交換
22                 num[idx - j] = num[idx - j - 1];
23                 //退避していた要素を代入
24                 num[idx - j - 1] = temp_space;
25             }
26         }
27     }
28
29     //SORTされたかを確認
30     printf("Bubble sort の結果は\n");
31     for (int i = 0; i < idx + 1; i++) {
32         printf("%d,", num[i]);
33     }
34     printf("\n");
35
36     return 0;
37 }
```

Bubble sort の結果は
0,1,2,3,4,5,6,7,8,9,

選択ソート(課題1)とは

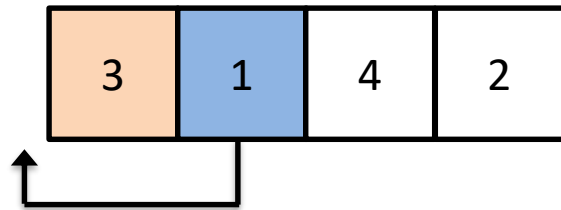
最初の要素(添え字がゼロ)を最小値と仮に決めて、他の全ての要素と比較し、最も小さい数字と入れ替える



挿入ソート(課題2)とは

まだ整列されていない要素を, 既に整列された要素に
の適切なところに挿入するソートアルゴリズム

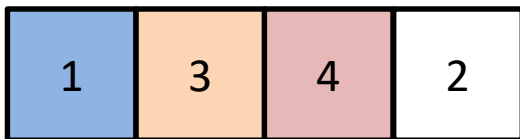
1は3よりも小さいので3の左に1を挿入



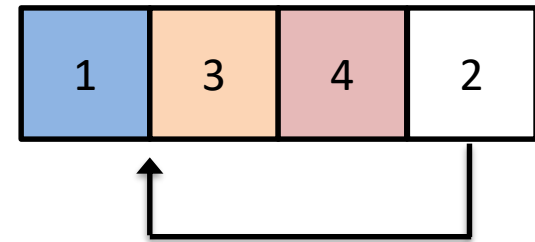
挿入



4は1, 3よりも大きいので挿入なし



2は1よりも大きく, 3, 4よりも
小さいので1と3の間に挿入



挿入



終了!!

