４Ｍ　情報工学Ⅱ

第１１回

ポインタと文字列

授業予定

- ~~第１回　情報工学Ｉの復習~~
- ~~第２回　数値データの受け渡し~~
- ~~第３回　配列データの受け渡し~~
- ~~第４回　関数の設計・総合演習~~
- ~~第５回　ファイル書き込み~~
- ~~第６回　ファイル読み込み~~
- ~~第７回　総合演習~~
- ~~第８回　前期中間試験~~
- ~~第９回　試験の解答と解説~~
- ~~第１０回　ポインタの基礎~~
- 第１１回　ポインタと文字列
- 第１２回　構造体の宣言と利用
- 第１３回　構造体の配列的利用
- 第１４回　総合演習
- 第１５回　前期末試験

＜解説＞
前回の課題
13

```c
#include <stdio.h>
↓
void sort3(int *n1, int *n2, int *n3)↓
{↓
        int max, min, sum=*n1+*n2+*n3;↓
↓
        if(*n1>*n2){↓
                max=*n1;        min=*n2;↓
        }else{↓
                max=*n2;        min=*n1;↓
        }↓
        if(*n3>max)             max=*n3;↓
        if(*n3<min)             min=*n3;↓
↓
        *n1=min;↓
        *n2=sum-max-min;↓
        *n3=max;↓
}↓
↓
int main (void)↓
{↓
        int na, nb, nc;↓
↓
        puts("３つの整数を入力して下さい");↓
        printf("整数A：");      scanf("%d", &na);↓
        printf("整数B：");      scanf("%d", &nb);↓
        printf("整数C：");      scanf("%d", &nc);↓
        ↓
        sort3(&na, &nb, &nc);↓
↓
        puts("昇順に並べ替えました");↓
        printf("整数Aは%dです。¥n", na);↓
        printf("整数Bは%dです。¥n", nb);↓
        printf("整数Cは%dです。¥n", nc);↓
↓
        return(0);↓
}↓
[EOF]
```

ポインタ受けには ＊ をつける

2値の比較で
最大・最小

3つ目の値を比較
最大・最小を入替え

昇順に入替え

ポインタ渡しには & をつける

＜解説＞
前回の課題
14

List 10-9

```
/*
    二つの実数値を交換する（間違え）
*/

#include  <stdio.h>

/*---  nx・nyが指すオブジェクトの値を交換  ---*/
void swap(int *nx, int *ny)
{
    int  temp = *nx;
    *nx = *ny;
    *ny = temp;
}

int main(void)
{
    double  dx, dy;

    puts("二つの実数を入力してください。");
    printf("実数Ｘ：");    scanf("%lf", &dx);
    printf("実数Ｙ：");    scanf("%lf", &dy);

    swap(&dx, &dy);

    puts("これらの値を交換しました。");
    printf("実数Ｘは%fです。\n", dx);
    printf("実数Ｙは%fです。\n", dy);

    return (0);
}
```

間違い修正

(double *nx, double *ny)
※宣言型を合わせる

実行結果例

```
二つの整数を入力してください。
実数Ｘ：53.5
実数Ｙ：21.68
これらの値を交換しました。
実数Ｘは9980.450456です。
実数Ｙは50.568782です。
```

ポインタと配列

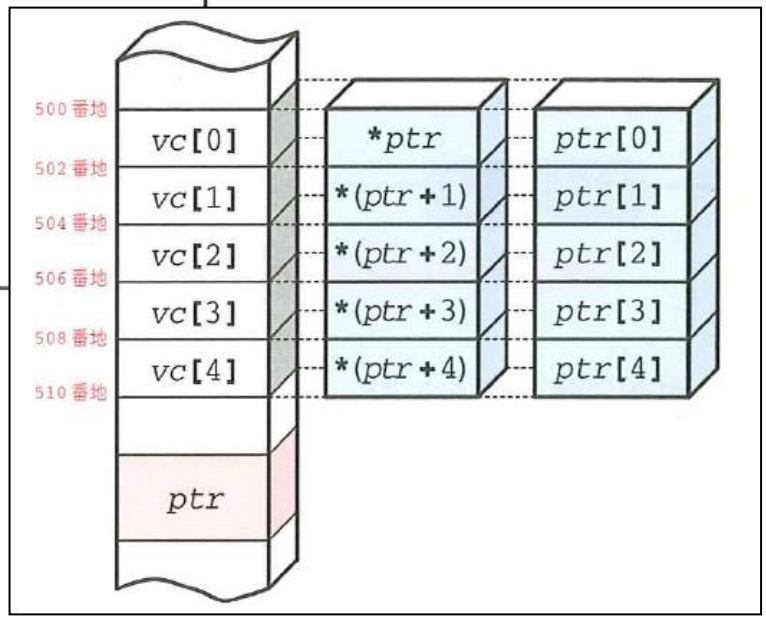
- ポインタは配列と同じように扱える

List 10-10

```
/*
    配列とポインタ
*/

#include  <stdio.h>

int main(void)
{
    int  i;
    int  vc[5] = {10, 20, 30, 40, 50};
    int  *ptr = &vc[0];

    for (i = 0; i < 5; i++)
        printf("vc[%d] = %d    ptr[%d] = %d    *(ptr + %d) = %d\n",
                                  i,  vc[i], i, ptr[i], i, *(ptr + i));

    return (0);
}
```

実行結果

```
vc[0] = 10    ptr[0] = 10    *(ptr + 0) = 10
vc[1] = 20    ptr[1] = 20    *(ptr + 1) = 20
vc[2] = 30    ptr[2] = 30    *(ptr + 2) = 30
vc[3] = 40    ptr[3] = 40    *(ptr + 3) = 40
vc[4] = 50    ptr[4] = 50    *(ptr + 4) = 50
```



ポインタと文字列

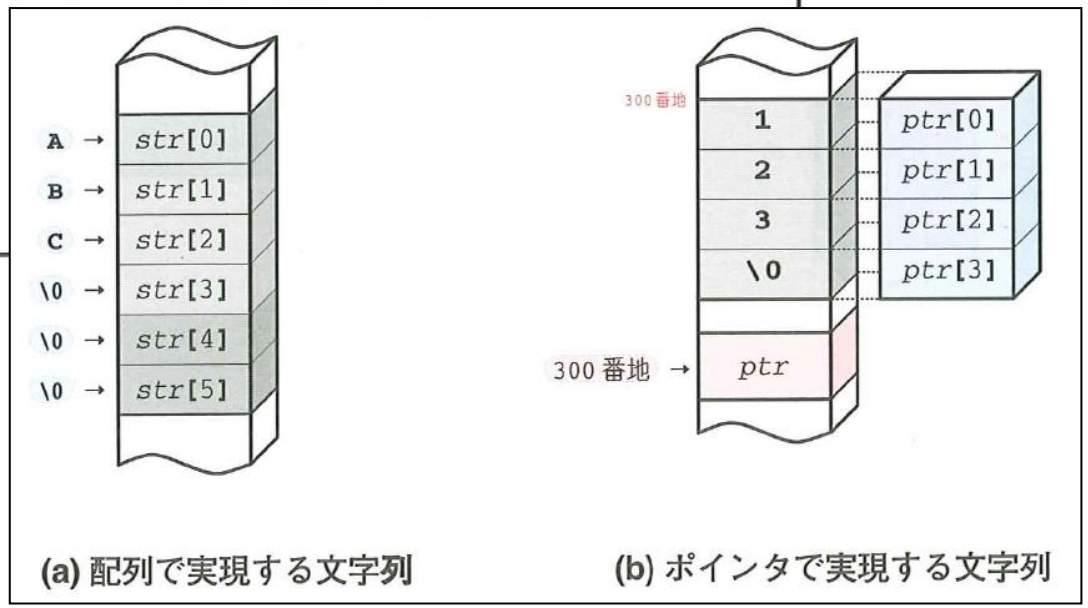
- 文字の配列もポインタによって，取扱いが広がる。

List 11-4

```
/*
    配列で実現する文字列とポインタで実現する文字列（その２）
*/

#include  <stdio.h>

int main(void)
{
    char   str[6] = "ABC";      /* 配列で実現する文字列 */
    char  *ptr    = "123";      /* ポインタで実現する文字列 */

    printf("str = \"%s\"\n", str);
    printf("ptr = \"%s\"\n", ptr);

    return (0);
}
```

実行結果

```
str = "ABC"
ptr = "123"
```



(a) 配列で実現する文字列　　(b) ポインタで実現する文字列

ポインタによる文字列の操作

- ポインタをうまく活用して，文字列を操作しよう。

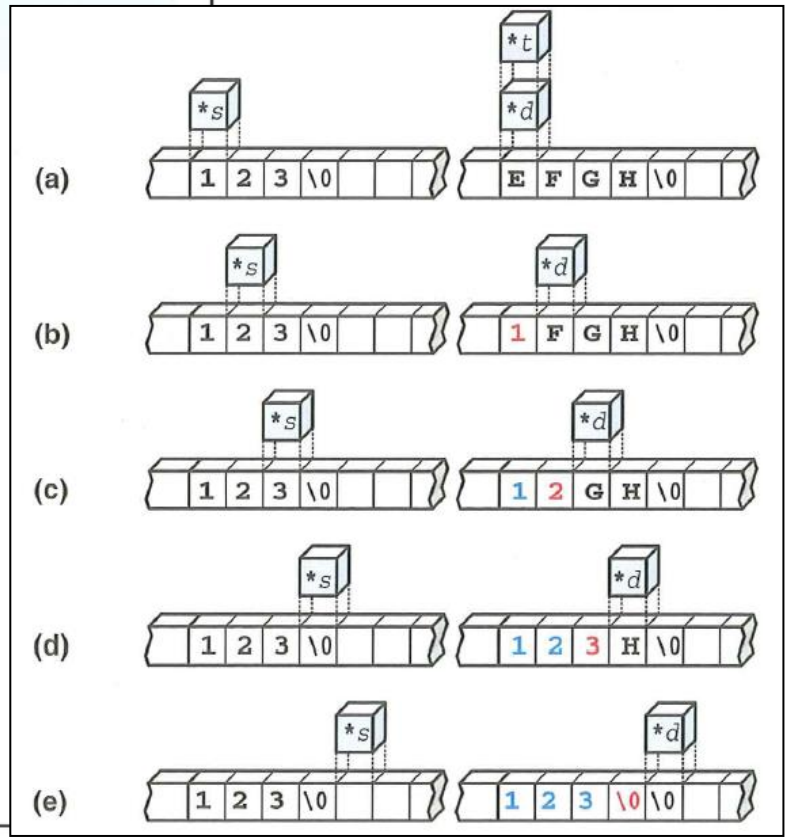
List 11-7

```
/*
    文字列をコピーする
*/

#include  <stdio.h>

/*--- 文字列sをdにコピーする ---*/
char *str_copy(char *d, const char *s)
{
    char  *t = d;

    while (*d++ = *s++)
        ;
    return (t);
}

int main(void)
{
    char  s1[128] = "ABCD";
    char  s2[128] = "EFGH";

    printf("文字列s1：");      scanf("%s", s1);

    str_copy(s2, s1);

    puts("s1をs2にコピーしました。");
    printf("s1 = %s\n", s1);
    printf("s2 = %s\n", s2);

    return (0);
}
```

実行例

```
文字列s1：123↵
s1をs2にコピーしました。
s1 = 123
s2 = 123
```

```
unsigned  i = 0;
while (d[i] = s[i])
    i++;
```



文字列を扱うライブラリ関数

ヘッダ　　#include <string.h>

strlen	
ヘッダ	#include <string.h>
形　式	size_t *strlen*(const char *s)
解　説	sが指す文字列の長さ（ナル文字は含まない）を求める。
返却値	sが指す文字列の長さを返す。

strcpy	
ヘッダ	#include <string.h>
形　式	char *strcpy(char *s1, const char *s2)
解　説	s2が指す文字列を、s1が指す配列にコピーする。コピー元とコピー先が重なる場合の動作は未定義とする。
返却値	s1の値を返す。

strncat	
ヘッダ	#include <string.h>
形　式	char *strncat(char *s1, const char *s2, size_t n)
解　説	s2が指す文字列を、s1が指す配列の末尾に連結する。s2の長さがnより長い場合は、切り捨てる。コピー元とコピー先が重なる場合の動作は未定義とする。
返却値	s1の値を返す。

文字列変換関数

- 文字列を整数値や浮動小数点値へと変換する

ヘッダ	#include <stdlib.h>

atoi

ヘッダ	**#include** <stdlib.h>
形　式	**int** ***atoi*****(const char** ****nptr*****)**
解　説	*nptr*が指す文字列を、**int**型の表現に変換する。
返却値	変換された値を返す。結果の値が**int**型で表現できないときの動作は定義されない（処理系に依存する）。

atol

ヘッダ	**#include** <stdlib.h>
形　式	**long** ***atol*****(const char** ****nptr*****)**
解　説	*nptr*が指す文字列を、**long**型の表現に変換する。
返却値	変換された値を返す。結果の値が**long**型で表現できないときの動作は定義されない（処理系に依存する）。

atof

ヘッダ	**#include** <stdlib.h>
形　式	**double** ***atof*****(const char** ****nptr*****)**
解　説	*nptr*が指す文字列を、**double**型の表現に変換する。
返却値	変換された値を返す。結果の値が**double**型で表現できないときの動作は定義されない（処理系に依存する）。

課題15

入力した文字データの先頭文字だけを“大文字”に変換するプログラムをポインタを使って作成せよ。

```
Z:¥Desctop>11-1
文字列を入力して下さい:abcdefg⏎
変換前:abcdefg
変換後:Abcdefg
```

ヒント:大文字に変換する関数　toupper

toupper　（教科書　p.223）

ヘッダ	#include <ctype.h>
形　式	int toupper(int c)
解　説	英小文字を対応する英大文字に変換する。
返却値	cが英小文字であれば、英大文字に変換した値を返す。そうでなければ、cをそのまま返す。

課題15

```
1  #include <stdio.h>↓
2  #include <ctype.h>↓
3  ↓
4  void str_htoupper( ここを考えよう )↓
5  {↓
6  ↓
7  ↓
8  ↓        ここを考えよう
9  ↓
10 ↓
11 }↓
12 ↓
13 int main (void)↓
14 {↓
15         char str[100];↓
16 ↓
17         printf("文字列を入力して下さい：");↓
18         scanf("%s", str);↓
19 ↓
20         printf("変換前：%s¥n", str);↓
21 ↓
22         str_htoupper( ここを考えよう );↓
23 ↓
24         printf("変換後：%s¥n", str);↓
25 ↓
26         return(0);↓
27 }↓
28 [EOF]
```

課題16

- 文字列の整列
  文字列を昇順(ABC順)に並べ替えるプログラムをポインタ操作を用いて作成せよ。

ヒント:文字列の大小比較関数　strcmp

（教科書　p.263）

strcmp

ヘッダ	`#include <string.h>`
形　式	`int strcmp(const char *s1, const char *s2)`
解　説	*s1*が指す文字列と*s2*が指す文字列の大小関係（先頭から順に1文字ずつ比較していき、異なる文字が出現したときに、それらの文字の対に成立する大小関係とする）の比較を行う。
返却値	等しければ0、*s1*が*s2*より大きければ正の整数値、*s1*が*s2*より小さければ負の整数値を返す。

課題16

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

#define    N    6

void prtdata(char *p, char *s[])
{
        int i;

        printf("%s:",p);
        for(i=0; i<N; i++)
                printf("%-8s", s[i]);
        printf("¥n");
}
void str_sort(char *p[])
{
        int i, j, m;
        char *w;

        for(i=N-1; i>=1; i--){
                m=i;
                for(j=i-1; j>=0; j--){
                        if(strcmp(p[j], p[m])>0)
                                m=j;
                }
                w=p[i];

                ここを考えよう

                ここを考えよう

                prtdata("                    ", p);
        }
}

```

```c
int main (void)
{
        int i;
        char *c[N]={"BB","FFFFFF","A","EEEEE","CCC","DDDD"};

        prtdata("元のデータ   : ", c);

        str_sort(c);

        prtdata("ソート後     : ", c);

        return(0);
}
[EOF]
```

```
Z:¥Desctop>11-2
元のデータ  :BB   FFFFFF A    EEEEE  CCC    DDDD
            :BB   DDDD   A    EEEEE  CCC    FFFFFF
            :BB   DDDD   A    CCC    EEEEE  FFFFFF
            :BB   CCC    A    DDDD   EEEEE  FFFFFF
            :BB   A      CCC  DDDD   EEEEE  FFFFFF
            :A    BB     CCC  DDDD   EEEEE  FFFFFF
ソート後    :A    BB     CCC  DDDD   EEEEE  FFFFFF
```