プログラミング論I
(12) 配列を引数にとる関数

電子情報工学専攻　日浦 慎作

ログインしておいてください

復習スライド

関数の作り方・使い方(2)

```
#include <stdio.h>

int square(int x);

int main(void) {
    int y;

    y = square(5);
    printf("result is %d¥n", y);
    return 0;
}

int square(int x) {
    return x * x;
}
```

戻り値の型は int である

引数の型は int である

引数の値 5 が，関数の引数変数 x に代入される

return の後ろの数式の値が，呼び出し側へ，戻り値として返される

配列を引数に取る関数

- 関数に，配列を渡したい．
- どんなことに用いるか？
  - 配列全体について処理をする関数を作る
    - 例１：配列の値の総和を求める
    - 例２：配列の値の中から，最大値を求める　など
  - 文字列の処理を行う関数を作る
    - 例１：文字列の文字数を数える
    - 例２：文字列を表示する　など

まとまった処理を関数にすると，プログラムがきれいになる

配列を引数に渡す

配列を受け取る関数であることを[ ]で宣言する

```
int minimum(int data[], int num);
```

プロトタイプ宣言

```
int main(void) {
      int seiseki[10] = {70, 85, … };
      int min;

      min = minimum(seiseki, 10);
      printf("minimum is %d¥n", min);
}
```

呼び出し側では，配列の名前のみ指定すればよい

```
int minimum(int data[], int num) {
      …..
}
```

受け取り側では，配列の要素数を指定する必要は無い

関数定義

```c
#include <stdio.h>

int minimum(int data[], int num);

int main(void) {
  int seiseki[10] = {70, 85, 100, 75, 57, 84, 73, 69, 59, 93};
  int min;

  min = minimum(seiseki, 10);
  printf("minimum is %d¥n", min);
}
```

実行結果

```
minimum is 57
```

```c
int minimum(int data[], int num) {
  int i;
  int min;

  min = data[0];
  for(i = 1; i < num; i++) {
    if(min > data[i]) {
      min = data[i];
    }
  }
  return min;
}
```

プログラム例1

配列中の最小値を求める関数

- 第1引数：データ（int の配列）
- 第2引数：データ数（配列の要素数）

ex1.c

配列を引数に渡すときのポイント

- 引数が配列であることを [ ] で明示する
  プロトタイプ宣言　関数定義　の2箇所（同じにする）
  [ ] により配列として認識され，利用できる
- 配列の要素数（[ ] 内の数字）は不要
  – プロトタイプ宣言や関数定義のときに
    要素数を書いても良いが，無視される
- 配列全体を関数に渡すときには，[ ]をつけない
  – [i]をつけると，i番目の要素だけを関数に
    渡すという意味に解釈される

```c
#include <stdio.h>

int minimum(int data[5], int num);

int main(void) {
  int seiseki[10] = {70, 85, 100, 75, 57, 84, 73, 69, 59, 93 };
  int min;

  min = minimum(seiseki, 10);
  printf("minimum is %d¥n", min);
}

int minimum(int data[5], int num) {
  int i;
  int min;

  min = data[0];
  for(i = 1; i < num; i++) {
    if(min > data[i]) {
      min = data[i];
    }
  }
  return min;
}
```

ここに数字を書いても，無視される

関数側では，引数で受け取った配列の要素数はわからないので，別の引数により知らせる必要がある

ex1b.c

練習課題

- 練習１

  ex1.c（配列中の最小値を求めるプログラム）の関数を変更して，seisekiの平均値を求めるプログラムを作成せよ.

  余力があれば，関数の戻り値を int でなく float にして，小数で平均点が表示されるようにせよ.

  なお平均値の正解は 76.5 である

プログラム例2　文字列中の数字の個数を数える関数

ex2.c

```
#include <stdio.h>

int countNum(char str[]);

int main(void) {
  char message[] = "Hello 123!";
  int len;

  len = countNum(message);
  printf("num of ¥"%s¥" is %d¥n", message, len);
}

int countNum(char str[]) {
  int i, cnt = 0;

  for(i = 0; str[i] != '¥0'; i++)
    if('0' <= str[i] && str[i] <= '9')
      cnt++;
  return cnt;
}
```

文字列は終端記号（ターミネータ）で終わりがわかるので，要素数を伝える必要がない

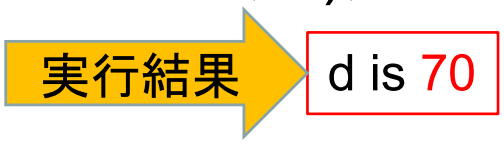
実行結果

```
num of "Hello 123!" is 3
```

関数での，配列の値の変更について

```
void change(int data);

int main(void) {
  int d = 70;

  change (d);
  printf("d is %d¥n", d);
}

void change(int data) {
  data = 0;
}
```

ex2a.c



関数内で変数の値を変更しても，
もとの変数の値は変わらない

```
void change(int data[], int num);

int main(void) {
  int d[10] = {70};

  change (d, 10);
  printf("d[0] is %d¥n", d[0]);
}

void change(int data[], int num) {
  data[0] = 0;
}
```

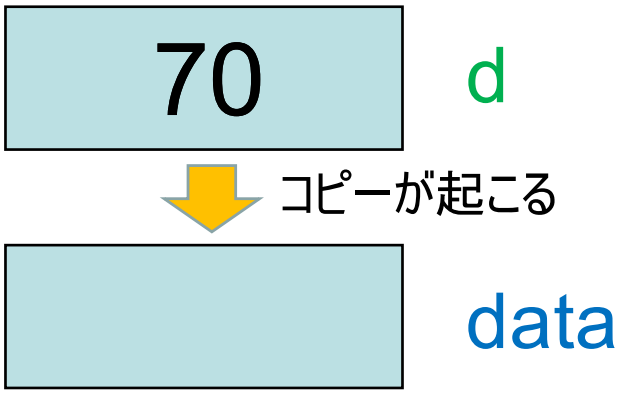
ex2b.c

関数内で配列の値を変更すると，
もとの配列の値も変わる

引数を関数に渡す仕組み(1)

```
void change(int data);

int main(void) {
  int d = 70;

  change (d);
  printf("d is %d¥n", d);
}

void change(int data) {
  data = 0;
}
```

実行位置



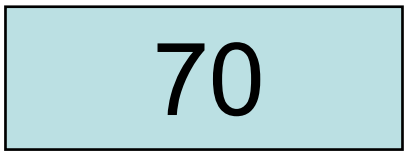
値渡し:
別の変数への値のコピー

普通の変数を関数の引数に渡した場合:関数で変数の値を変更しても,
もとの(呼び出し側の)変数の値は変わらない. なぜなら, 変数 data は 変数 d とは別物であり,
値がコピーされているだけだからである. これを値渡しと呼ぶ. (第8回「関数」のスライド参照)

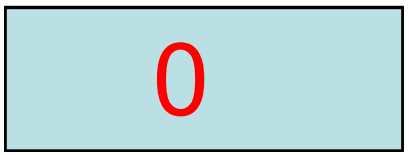
引数を関数に渡す仕組み(2)

```
void change(int data);

int main(void) {
  int d = 70;

  change (d);
  printf("d is %d¥n", d);
}

void change(int data) {
  data = 0;
}
```

実行位置



d



data

data の値が変わるだけ

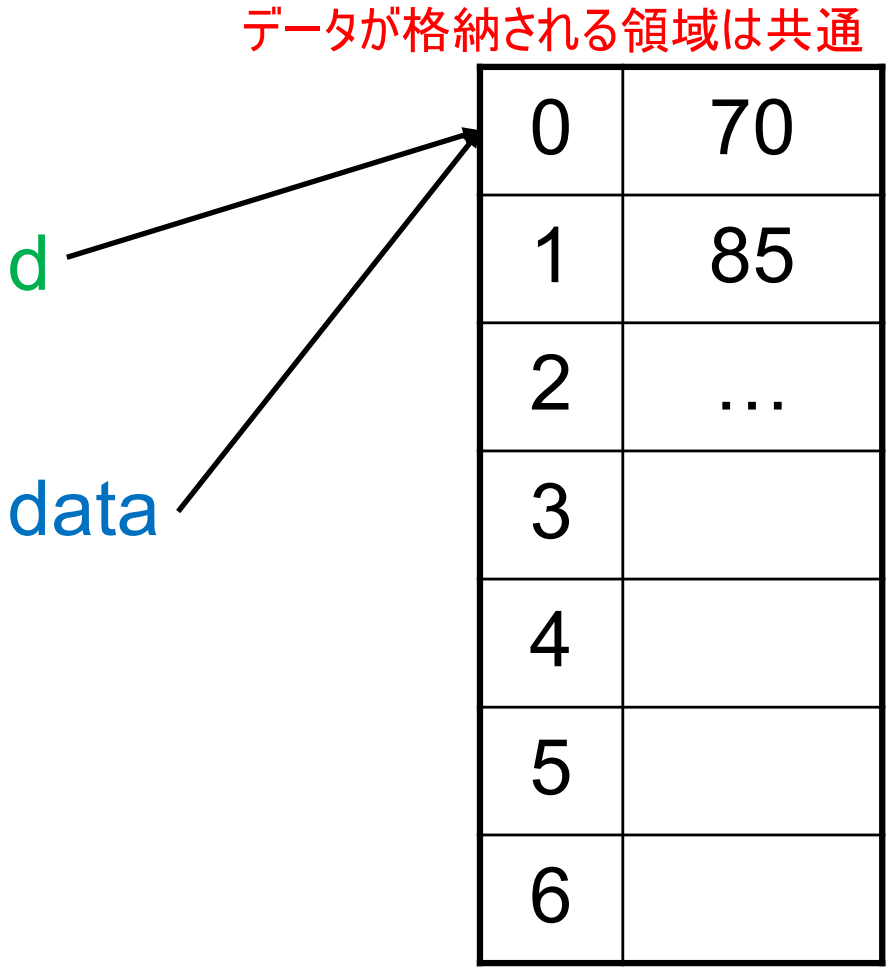
値渡し:
別の変数への値のコピー

普通の変数を関数の引数に渡した場合:関数で変数の値を変更しても,
もとの(呼び出し側の)変数の値は変わらない. なぜなら, 変数 data は 変数 d とは別物であり,
値がコピーされているだけだからである. これを値渡しと呼ぶ. (第8回「関数」のスライド参照)

配列を関数に渡す仕組み(1)

```
int change(int data[], int num);

int main(void) {
        int d [7] = {70, 85, … };

        change (d, 7);
        printf(“d[0] is %d¥n”, d[0]);
}

int change(int data[], int num) {
        data[0] = 0;
}
```

実行位置

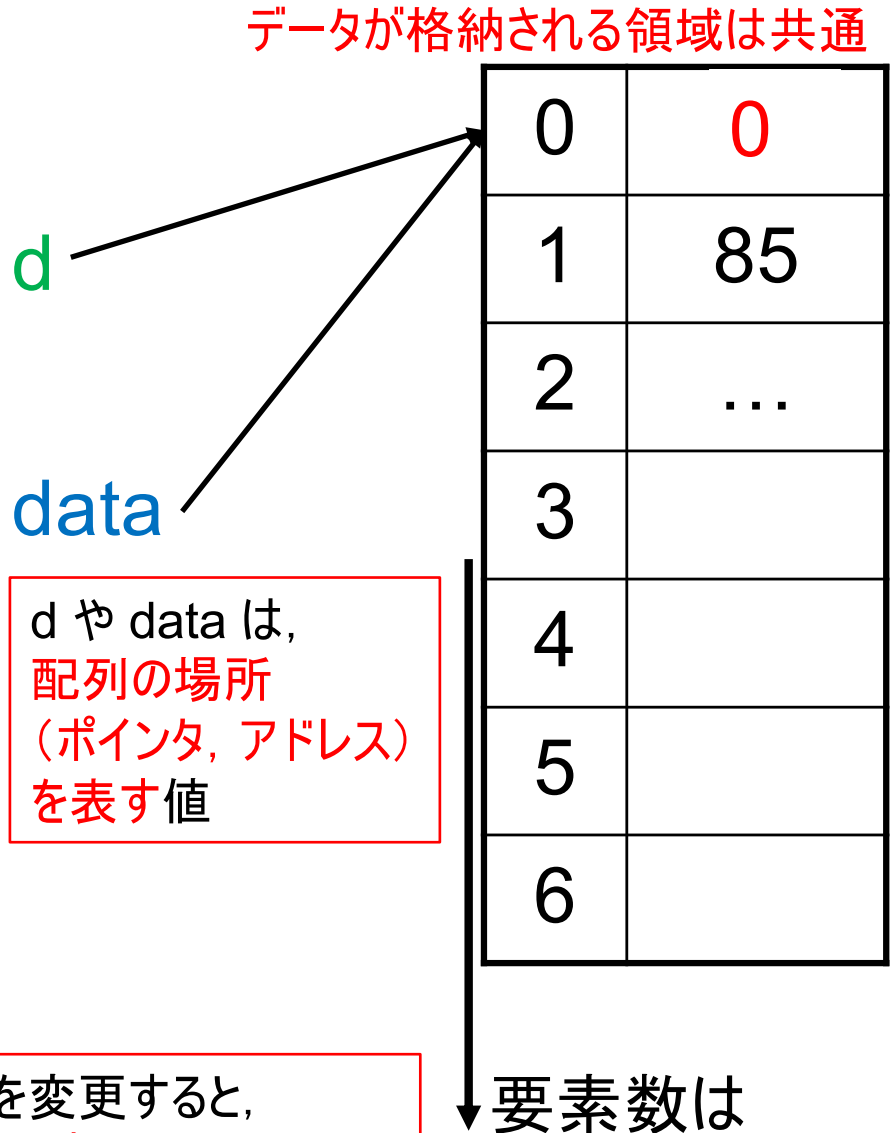


配列を関数の引数に渡した場合：関数で配列の値を変更すると，もとの（呼び出し側の）変数の値も変わる．なぜなら，関数には配列の場所（ポインタ）が渡されているからである．これをポインタ渡しと呼ぶ．

配列を関数に渡す仕組み(2)

```
int change(int data[], int num);

int main(void) {
        int d [7] = {70, 85, … };

        change (d, 7);
        printf("d[0] is %d¥n", d[0]);
}

int change(int data[], int num) {
        data[0] = 0;
}
```

実行位置



d や data は，
配列の場所
（ポインタ，アドレス）
を表す値

配列を関数の引数に渡した場合：関数で配列の値を変更すると，
もとの（呼び出し側の）変数の値も変わる．なぜなら，関数には配列の
場所（ポインタ）が渡されているからである．これをポインタ渡しと呼ぶ．

プログラム例4　文字列の大文字を小文字に変える関数

ex4.c

```
#include <stdio.h>

void decapitalize(char str[]);

int main(void) {
  char message[] = "Hello World!";

  decapitalize(message);
  printf("result: ¥"%s¥"¥n", message);
}

void decapitalize(char str[]) {
  int i;

  for(i = 0; str[i] != '¥0'; i++)
    if('A' <= str[i] && str[i] <= 'Z') {
      str[i] = str[i] - 'A' + 'a';
    }
}
```

文字列は配列の一種なので，さきに説明したように，関数により文字（配列の要素の値）を変更すると，呼び出し元の値も変化する．

実行結果

```
result: "hello world!"
```

練習課題

- 練習２

  ex4.c（文字列の長さを求めるプログラム）の関数を変更して，文字列中の ' ' （スペース，空白文字）を '_'（アンダースコア）に置き換えるプログラムを作成せよ.

  ヒント

  – スペースかどうか判定　　if(str[i] == ' ') {

  – アンダースコアに置き換える　　str[i] = '_';

（発展）ポインタの話

なぜ`scanf`では変数の前に　`&`　がいるのか？

- `&`　は「変数の場所（ポインタ）」を求める演算子.
- `&`　をつけることで，関数には変数に格納された値のかわりに，「変数の場所」が伝えられる.
- 関数は，その場所に，値を書き込みに行く.

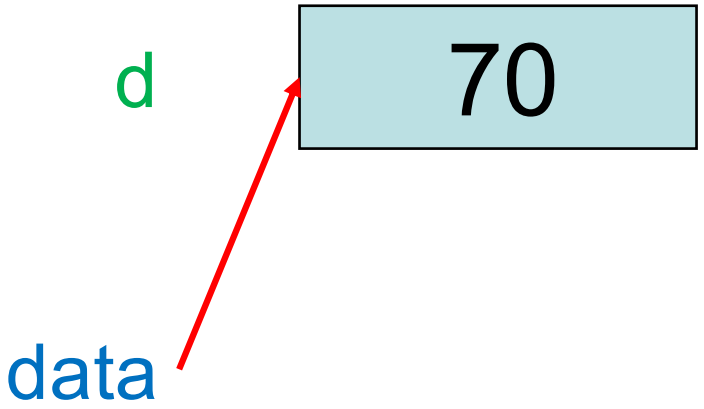
なぜ　`%s`　のときだけ　`&`　が不要なのか？

- 文字列は配列なので，そのまま渡せば「変数の場所」が伝えられる．関数はそこに書き込める.

詳しくは，２年生「プログラミング論２」「コンピュータ実習１」で学びます.

（発展）ポインタ渡し

```
void change(int *data);

int main(void) {
   int d = 70;

   change (&d);
   printf("d is %d¥n", d);
}

void change(int *data) {
   *data = 0;
}
```



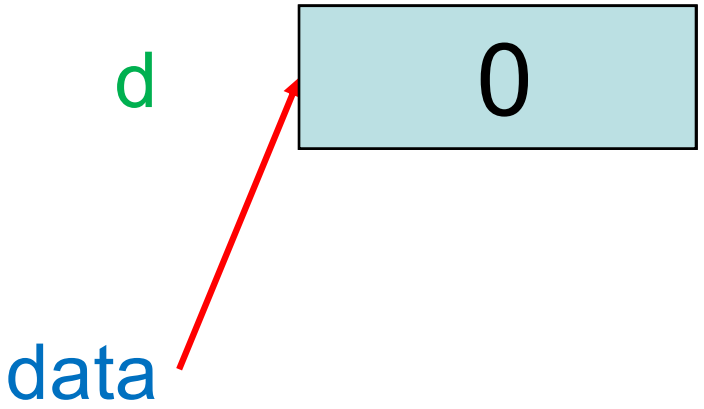
ポインタ渡し：
関数への場所の伝達

&	変数の場所（ポインタ，アドレス）を求める演算子
*（単項演算子）	ポインタ（アドレス）により差された変数にアクセスする演算子

（発展）ポインタ渡し

```
void change(int *data);

int main(void) {
   int d = 70;

   change (&d);
   printf("d is %d¥n", d);
}

void change(int *data) {
   *data = 0;
}
```

実行位置



ポインタ渡し：
関数への場所の伝達

&	変数の場所（ポインタ，アドレス）を求める演算子
*（単項演算子）	ポインタ（アドレス）により差された変数にアクセスする演算子

練習課題

- 練習３

  ex1.c（配列中の最小値を求めるプログラム）の関数を変更して，得点の分散を求めるプログラムを作成せよ.

  ヒント：分散の求め方には２種類の方法がある.

  – 方法１　まず平均を求める．つぎに，各要素の値から平均を引いた値の二乗の総和を求め，要素数で割る.

  $$\frac{1}{n}\sum_{n=1}^{n}(x_i-\overline{x})^2$$

  – 方法２　各要素の値の二乗の平均から，平均の二乗を引く.

  $$\frac{1}{n}\sum_{i=1}^{n}{x_i}^2-\overline{x}^2$$

  なお分散の正解は，175.25 である