プログラミング論I
(10) 配列

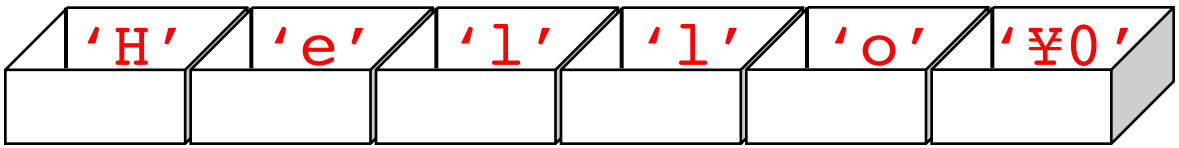
電子情報工学専攻　日浦 慎作

ログインしておいてください

配列とは

- 数学でよく使う，添字付きの変数に対応

  $a_1, a_2, \cdots, a_N$ のように，変数を番号で指定する
- どういうときに便利なのか？
  - まとまった数のデータを処理するとき
    - クラスのテストの平均点を求めるなど
  - 変数に対して一括処理，繰り返し処理をしたいとき
    - すべての変数の値を2倍する，など
  - 文字列を扱うとき
    - 文字列（例：“Hello”）は文字の羅列なので，
      C言語では配列で取り扱う．来週の講義のテーマ．

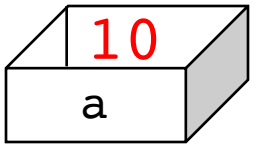


配列のイメージ

- 普通の変数

  `int a;`　整数を入れられる箱（名前a）を1つ用意

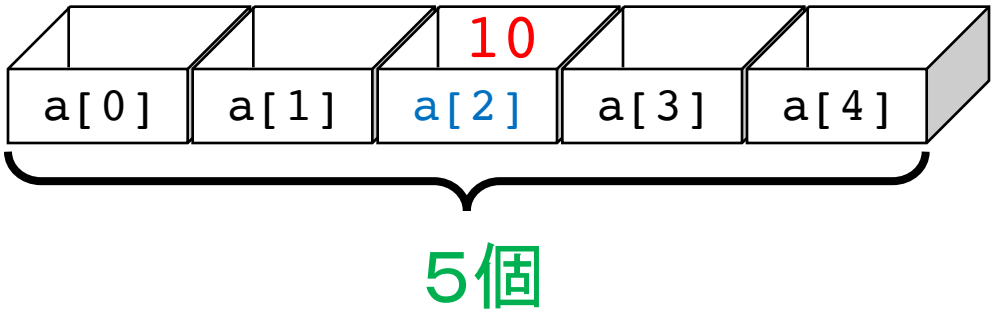
  `a = 10;`　数値10を箱に入れる



- 配列変数

  `int a[5];`　整数を入れられる箱を5個，用意

  `a[2] = 10;`　添字2の箱に10を入れる



C言語のプログラムでは・・

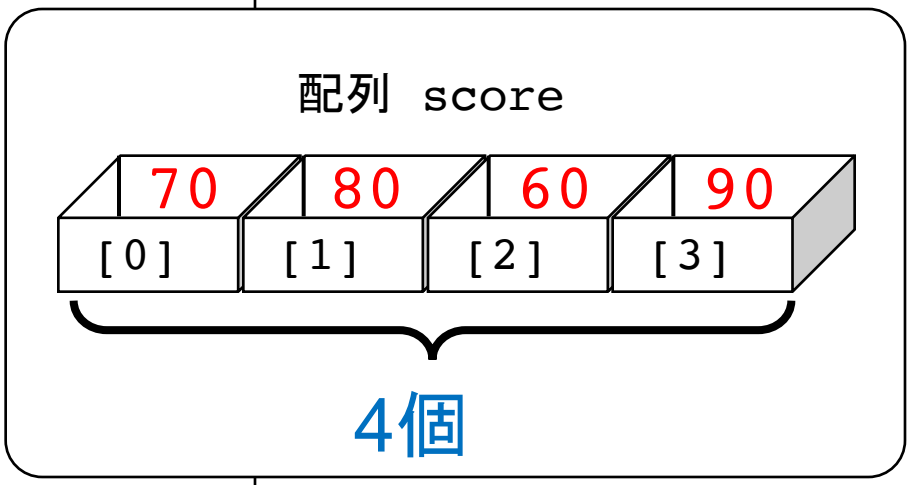
- 四角いカッコ [　] を使う
- 添字には，数値だけでなく，変数や数式も使える

  数学で a_i とか b_{2j+1} とか書くときの i とか 2j+1と同様

```
int main(void) {
  int i;
  int score[4];

  score[0] = 70;
  score[1] = 80;
  score[2] = 60;
  score[3] = 90;

  for (i = 0; i < 4; i++) {
    printf("%d¥n", score[i]);
  }
  …
```



添字に変数を使用した例

プログラム例1

```
#include <stdio.h>

int main(void) {
  int i;
  int score[4];

  score[0] = 70;
  score[1] = 80;
  score[2] = 60;
  score[3] = 90;

  for (i = 0; i < 4; i++) {
    printf("%d¥n", score[i]);
  }
  return 0;
}
```

やってみよう:
データの数を6個に増やす
（データ値はなんでもよい）

赤字の部分に注意！

ex1.c

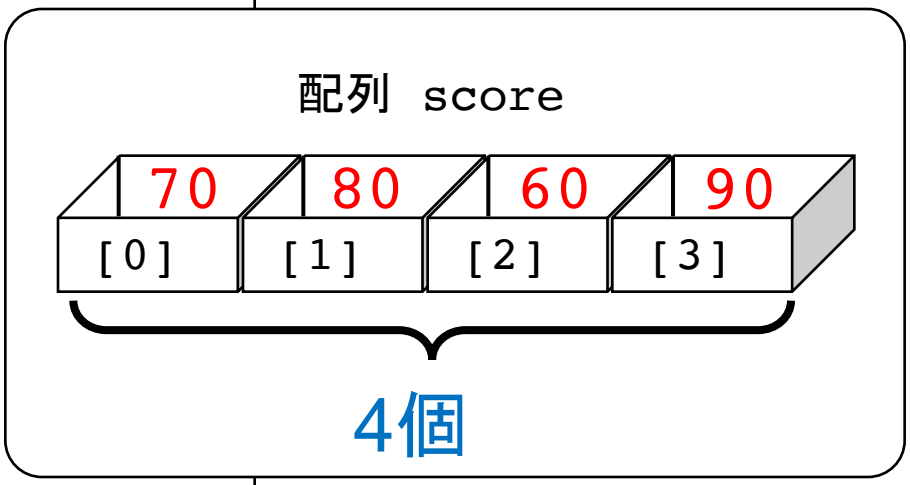
C言語での配列のルール(1)

- 【重要！】添字は必ず，0から始まる
  （他の言語では1から始まるものもあるが・・）

```
int main(void) {
  int i;
  int score[4];

  score[0] = 70;
  score[1] = 80;
  score[2] = 60;
  score[3] = 90;

  for (i = 0; i < 4; i++) {
    printf("%d¥n", score[i]);
  }
  …
```



だから，C言語では普通，4回繰り返すときには，変数を0から3まで増やして繰り返す

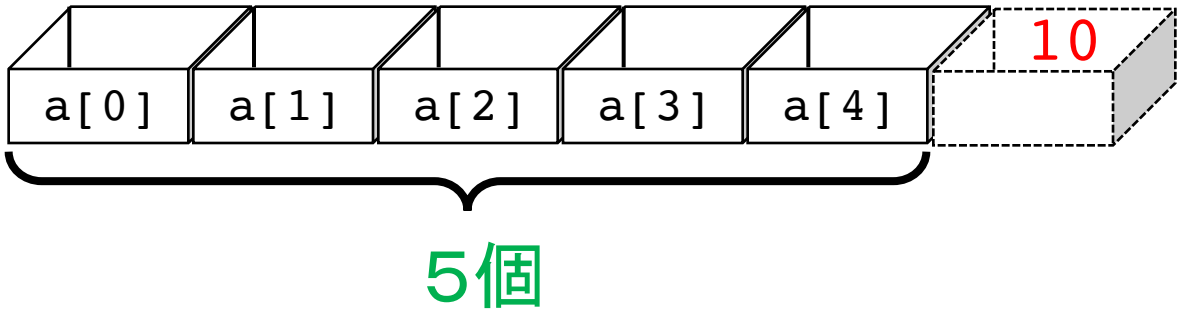
C言語での配列のルール(2)

- 変数定義時の[ ]内の数字は，準備する変数の個数
  - int a[5]; とした場合，使える添え字は0から4まで．
    a[5]は存在しないので，アクセスしてはならない．
  - 添え字が要素数を超えているかどうかはチェックされない．
    超えるとプログラムが異常動作，強制終了することがある

```
int a[5];
a[5] = 10;
```

整数を入れられる箱を5個，用意

添字5の箱はないので，ダメ！



配列の初期値の設定方法

番号	データ
0	85
1	72
2	48
3	96
4	84
5	77
6	58

```
int seiseki[7] = {85, 72, 48, 96,
                  84, 77, 58};
```
もしくは
```
int seiseki[] = {85, 72, 48, 96,
                 84, 77, 58};
```

- この場合，添え字を省略することも出来る
  - 省略した時，配列の大きさ（要素数）は，初期値の数と同じになる（上の例では7個）
- 初期値は要素数より少なくても良い

`int seiseki[7] = {85, 72, 48};`
とした場合，`seiseki[3]`から`seiseki[6]`の変数も用意され，初期値は0になる

プログラム例2

```
#include <stdio.h>

int main(void) {
  int data[5] = { 3, 4, 2, 6, 0 };
  int i;

  for(i = 0; i < 5; i++) {
    printf("data[%d] = %d¥n", i, data[i]);
  }
  return 0;
}
```

ex2.c

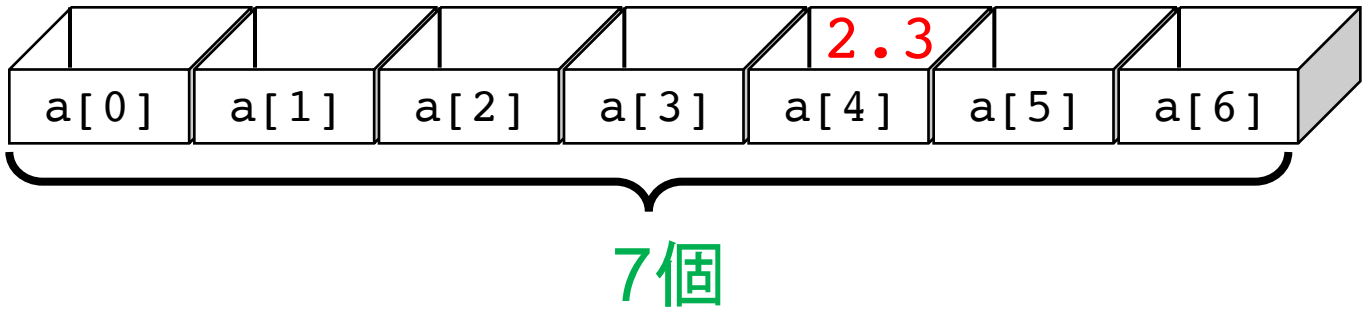
やってみよう:

データの表示を逆順にする

C言語での配列のルール(3)

- どのような型でも配列に出来る
  - int, double, その他なんでも可能

```
float a[7];    小数を入れられる箱を7個，用意
a[4] = 2.3;    4番の箱に 2.3 を入れる
```



C言語での配列のルール(4)

- 添え字には整数値しか使えない.
  - 添字に float や double 型の変数は使用不可.
  - 数式に 2.3 などの小数が混じるのもダメ.

- エラーになる例(添字が整数でない例)

```
float a[7];
double b = 4;


a[3.5] = 24;          //  添字に小数はエラーになる
a[b] = 23;            //  添字には，小数型の変数もダメ
a[1.5 * 2] = 6;       //  計算結果が整数でも，途中で
                      //  小数を使うとダメ
```

C言語での配列のルール(5)

- 配列の大きさ(要素数)は定数で指定する
  - 配列の要素数に，変数値を使うのは不可※.

```
int a = 3;
float data[a];
```

```
void func(int b) {
  float data[b];
  ...
}
```

エラーになる例

```
float data[10];
```

```
double data[5 * 2];
```

OKの例

より正確に言うと，プログラム実行時に大きさが決まるような書き方は不可.
コンパイルの時点で，大きさが決まらなければならない※.

※ANSI-Cの拡張であるC99では可能になっている．例えばMacやLinuxのgccでは可能.

#define（マクロ）を使おう

- 文字列(単語)の置き換えができる
  - 以下の右の例では，NUMが30に置き換えられる

```
int main(void) {
  int i;
  int score[30];

  score[0] = 93;
  score[1] = 18;
  ...
  score[29] = 65;

  for (i = 0; i < 30; i++) {
    printf("%d¥n", score[i]);
  }
  …
```


```
#define NUM 30

int main(void) {
  int i;
  int score[NUM];

  score[0] = 93;
  score[1] = 18;
  ...
  score[29] = 65;

  for (i = 0; i < NUM; i++) {
    printf("%d¥n", score[i]);
  }
  …
```

なぜ #defineを使うのか？

- プログラムの修正を一括して行うことができる
  - 前ページの例では，先頭の行を以下のように変えるだけで，データ数を50に増やせる
    ```
    #define NUM 50
    ```
- ミスを防ぐことができる
  - 配列の大きさ（要素数）と，forのループ回数の食い違いを防ぐことができる
- プログラムが読みやすくなる
  - ただの数値は，意味がわからない．名前をつけることで，理解しやすくなる．

プログラム例3

```
#include <stdio.h>

int main(void) {
  int data[5] = { 3, 4, 2, 6, 0 };
  int i;

  for(i = 0; i < 5; i++) {
    printf("data[%d] = %d¥n", i, data[i]);
  }
  return 0;
}
```

ex2.c

例2のプログラムを

- データを小数にする
- データ数に #define を使う

ように直せ.

プログラム例3解答

```
#include <stdio.h>

#define NUM 5

int main(void) {
  float data[NUM] = { 3.2, 4.1, 2.3, 6.4, 0.2 };
  int i;

  for(i = 0; i < NUM; i++) {
    printf("data[%d] = %f¥n", i, data[i]);
  }
  return 0;
}
```

ex3.c

例2のプログラムを

- データを小数にする
- データ数に #define を使う

ように直せ.

コーディングルール

コーディングルールとは？

- 言語の文法では定められていないが，プログラムが
  わかりやすいように仲間内で決めた，自主的なルールのこと．

C言語でよく用いられるルールについて

- 変数名・関数名は，基本的に小文字とする
  - 途中でときどき，大文字を使うことはある（単語の頭文字など）

```
int scoreAverage;
int calcDistance(int x, int y);
```

- マクロ名は，原則，すべて大文字が使われる

```
#define STUDENT_NUMBER 100
```

- 段付け（インデント）と同様に，極力守って，
  きれいに書いてください！！

配列へのデータの読み込み

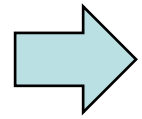
- 通常の変数と同じように scanf を用いる.

  & を忘れないように!

```
int a[5];
scanf(“%d”, &a[2]);
```

- もちろん，繰り返し処理で読み込むのも可能

```
int a[5], i;
for(i = 0; i < 5; i++) {
        printf(“data[%d] =“, i);
        scanf(“%d”, &a[i]);
}
```



実行例

```
data[0] =3
data[1] =2
data[2] =5
data[3] =1
data[4] =6
```

赤字はキーボード入力した値の例

プログラム例4

```
#include <stdio.h>

#define NUM 5

int main(void) {
  float data[NUM];
  float sum = 0;
  int i;

  for(i = 0; i < NUM; i++) {
    printf("data[%d] = ", i);
    scanf("%f", &data[i]);
  }

  for(i = 0; i < NUM; i++) {
    sum += data[i];
  }
  printf("sum of all data = %f¥n", sum);
  return 0;
}
```

やってみよう:

合計の代わりに平均を計算するように変更

ヒント

合計を計算した後（2つ目のforの後）で，合計値sumをデータ数NUMで割れば良い.

ex4.c