プログラミング入門
第7回講義

- 配列（その１）
  - 配列の定義
  - 配列の使い方
  - 配列の落とし穴
  - 配列の使用例

マークのあるサンプルプログラムは
`/home/course/prog0/public_html/2006/source/`
下に置いてありますから、各自自分のディレクトリに
コピーして、コンパイル・実行してみてください

配列の必要性(p.204)

n人の得点の平均　　3人　→　簡単
24人→　大変！

```
mike = 58; jack = 73; lucy = 87; tom = 63;
.........
average = (float)(mike + jack + lucy +...)/24.0;
```

配列の必要性

名前ではなく、man0,man1と書くと楽か？？

→　手間は変わらない！

```
man0 = 58; man1 = 73; man2 = 87; man3 = 63;
.........
average = (float)(man1 + man2 + man3 +...)/24.0;
```

→　ループ（forなど）が使えるとうれしい！！

→　そこで配列を使う！

配列の定義(p.207)

- 配列の定義は以下のように普通の変数の後ろにカギかっこ[]をつけて個数を示す。
- 個数部分は正の整数の定数でなければいけない。

型　変数名[個数]

例:`int man[10]`

- 上例ではint型のmanという変数を10個宣言したことになる。

配列の要素と添字(1)(p.210)

```
int man[10];
```

- 上の宣言でmanという配列が宣言できた
  - 実際にはman[0] ,man[1],.....man[9] という10個の変数が宣言されたことになる。
  - この一つ一つの変数を「配列の要素」と言う。要素一つ一つを普通の変数として使用出来る。
- 配列変数のカッコ内に書く数字（この例の場合、0,1,...9）を「添字」又は「インデックス」と呼ぶ。

配列の要素と添字(2)

```
int man[10];
```

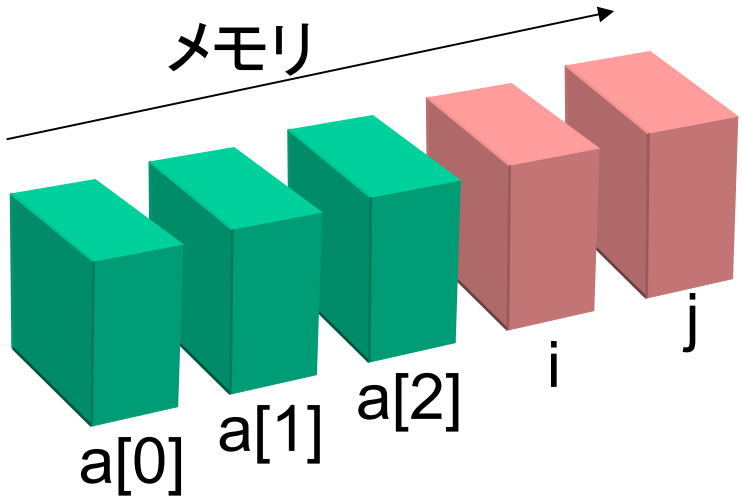
- 上の宣言を「要素」と「添字」を使って表現すると以下のようになる。

manはman[0]からman[9]までの10個の要素を持ったint型の配列で、添字の範囲は0から9までである。

配列の要素と添字(3)

```
int a[3],i,j;
```

と宣言すると、メモリ内には下図のような感じで変数の領域がとられて行く。(正確には場合によって異なる)
なお、箱一つの大きさはint型の大きさ(普通4バイト)である



やってみよう

- 下のような配列を宣言し、その要素を全て書いてみよう。

heightは5個の要素を持った浮動小数点（単精度）型の配列である。

配列を使った例

24人の成績を入力し、平均を求めるプログラム

```
#include <stdio.h>

main()
{
   int i, man[24], sum = 0 ;

   for(i = 0 ; i < 24 ; i++){
      scanf("%d",&man[i]);
      sum = sum + man[i];
   }
   printf("Average = %f\n",(float)sum/24.0);
}
```

24人分の成績を持つ配列

普通の変数と同じ使い方が出来る

printfの中で計算することも出来る

/home/course/prog0/public_html/2006/lec/source/lec07-1.c

Lec07-8の答え 宣言 float height[5]; 要素 height[0],height[1],height[2],height[3],height[4]

配列の宣言の注意点

- 宣言の際の要素数は「定数」であること
- 要素数が変数という宣言は出来ない

```
int i = 10;
int man[i];
```

- 変数の宣言（配列以外も同じ）はまとめて一番最初に行う。途中で使用するからと言って、途中で宣言することは出来ない。（コンパイルエラーになる）

```
int i;
i = 10;
int man[10];
```

配列変数の使用方法(p.210)

- 各要素は、添字があること以外は普通の変数と同様に使用出来る
- 代入（配列aの6個目の要素a[5]の値を3にする）

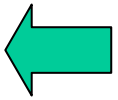
```
a[5] = 3;
```

- 値の参照（変数bの値を配列cの11個目の要素c[10]の値にする）

```
b = c[10];
```

- 配列まるごとの代入、参照は出来ない
（x,yが共に配列の時、 以下のような代入は出来ない）

```
x = y;
```



出来ない！

落とし穴(p.215)

- 「やってみよう」をやってみて、「おや？？」と思ったことはないだろうか？
- それは多分以下のことだろう

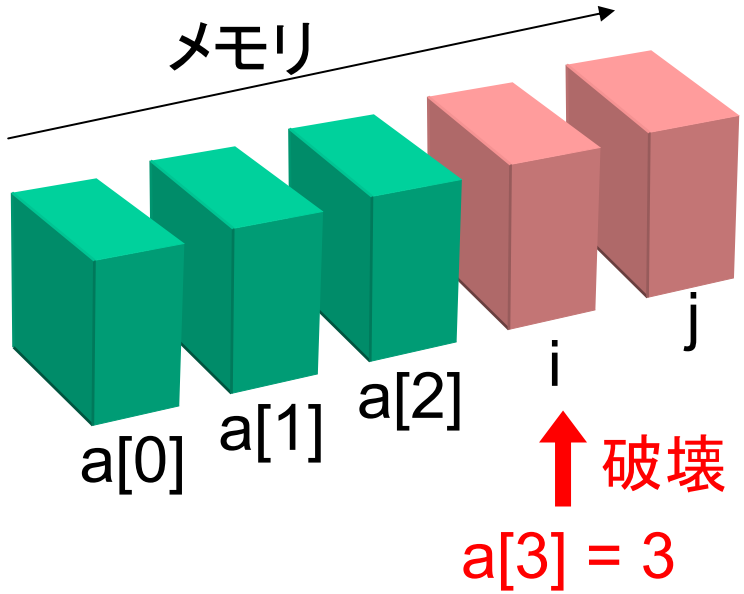
`int man[10];`と宣言した時に要素`man[10]`は存在しない

落とし穴（続き）

- `int man[10]` と宣言した時に存在する要素は
  `man[0],man[1],....man[8],man[9]`
  の10個になる。（最後の添字は　配列の大きさ−1）
- 決して`man[10]`という要素は存在しないので注意が必要。

落とし穴（続き）

- `int a[3],i,j;`
  とメモリの中に順に領域が取られたとする



プログラムが異常動作する事をプログラムが「暴走する」と言います。

- 間違って存在しないa[3]に数値を代入すると、実は変数iを破壊することになり、プログラムが暴走するかもしれない。

落とし穴（実験）

```
#include <stdio.h>

main()
{
  int i , j = 333;
  int test[3];

  test[0] = 0;
  test[1] = 1;
  test[2] = 2;

  for (i = 0 ; i < 5 ; i++){
    printf("test[%d] : %d\n",i,test[i]);
  }
}
```

```
std1dc1{s1000000}1: gcc lec07-2.c
std1dc1{s1000000}2: ./a.out
test[0] : 0
test[1] : 1
test[2] : 2
test[3] : -785519980
test[4] : -785580071
std1dc1{s1000000}3:
```

test[3],test[4]の値は不確定。
必ずしもこの実行結果と同じになるとは限らない。これはコンピュータの機種や使用しているOS(**Operating System**)の差に起因する

/home/course/prog0/public_html/2006/lec/source/lec07-2.c

配列の初期化(p.236)

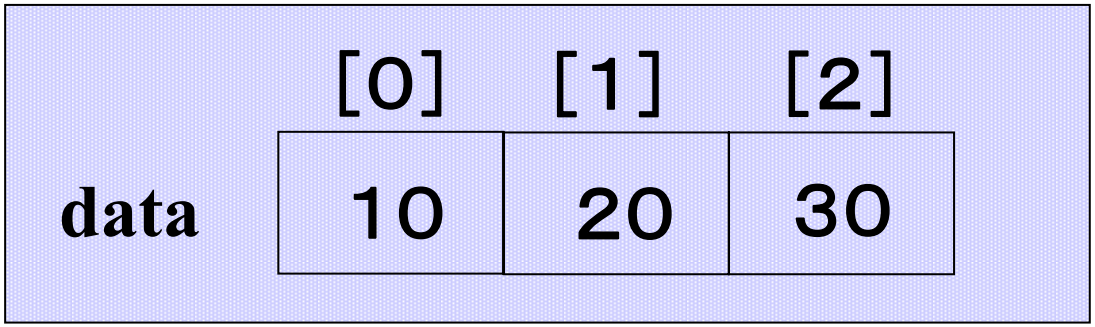
配列の初期化:

配列宣言時に初期値を同時に代入しておく

方法: 中カッコにカンマ区切りで添字の若番順に列挙する

例: 以下のように宣言すると、各要素が順に10，20，30という値で予め初期化される。

```
int data[3] = {10, 20, 30};
```



配列の使い方(1)
普通の変数と同じように

- 下例では
  - data[0]を国語の試験の点数
  - data[1]を数学の試験の点数
  - data[2]を英語の試験の点数

として、独立した変数のように使用している。

```
int data[3];
printf("国語、数学、英語の点数を空白で分けて入力して下さい＼n");
scanf("%d%d%d",&data[0],&data[1],&data[2]);
printf("3教科の平均は %f点です＼n",
       (float)(data[0]+data[1]+data[2])/3);
```

配列の使い方(2)
for文と一緒に

/home/course/prog0/public_html/2006/lec/source/lec07-3.c

```
#include <stdio.h>
main()
{
  int kokugo[10];
  int i , sum = 0;
  float average;

  for (i = 0 ; i < 10 ; i++){ /*データ入力、合計計算*/
    scanf("%d",&kokugo[i]);
    sum = sum + kokugo[i];
  }

  average = (float)sum / 10.0; /*平均計算*/

  for (i = 0 ; i < 10 ; i++){  /*データ、平均からの差出力*/
    printf("[%d] %3d %+6.2f\n",i,kokugo[i],kokugo[i] - average);
  }
}
```

10人分の成績を読み込み、その平均値を出し、各々の人の成績の平均からの差を表示するプログラム

全体(小数点も含む)で6桁、小数点以下2桁で符号つきの書式(例+34.56)

配列の使い方(3)
個数分(人数分)のデータ保管庫として

```
#include <stdio.h>
main()
{
  int eigo[4] = {89, 65, 78, 92};
  int i , sum = 0;
  float average;

  for (i = 0 ; i < 4 ; i++){ /*合計計算*/
    sum = sum + eigo[i];
  }
  average = (float)sum / 4.0; /*平均計算*/
  for (i = 0 ; i < 4 ; i++){  /*データ、平均からの差出力*/
    printf("[%d] %3d %+6.2f\n",i,eigo[i],eigo[i] - average);
  }
}
```

配列の初期化

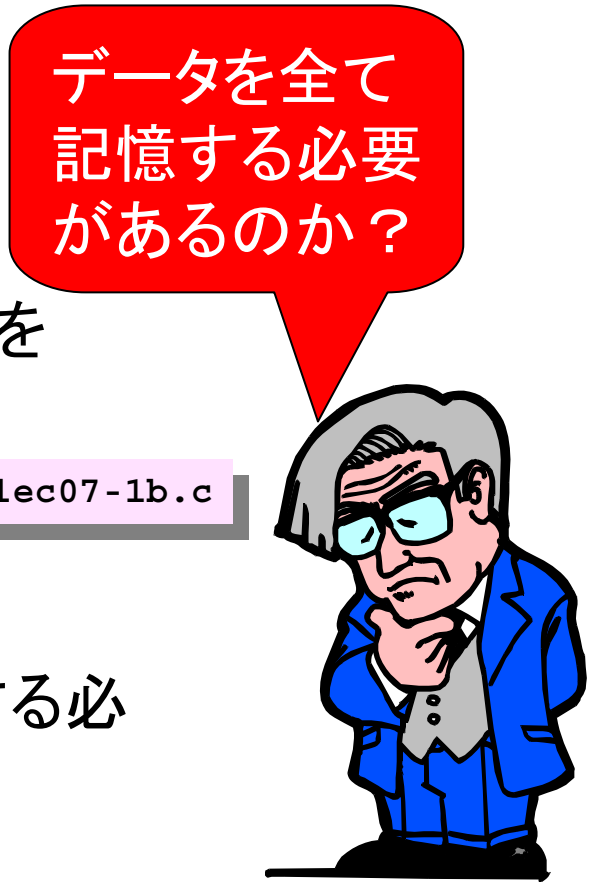
/home/course/prog0/public_html/2006/lec/source/lec07-4.c

配列を使用する基準

- 平均を求める
  - 記憶する必要なし
  - ループ中ですぐ加えていき、最後に個数で割る
  - 配列は必要ない（つまり lec07-1.c は配列を使う必要はなかった ⇨ lec07-1b.c参照）

```
/home/course/prog0/public_html/2006/lec/source/lec07-1b.c
```

- 各個人の平均からの差を求める
  - 後で個々のデータを使用するので、記憶する必要がある ⇨ 配列が必要



サンプル(1) 逆順に表示

```
#include <stdio.h>

main()
{
  int data[3];
  int i;

  for (i = 0 ; i < 3 ; i++){ /* forループによるデータ入力 */
    scanf("%d",&data[i]);
  }

  i = 2; /* iは3になっているはずなので i--(又は--i)でも良い */
  while (i >= 0){                /* while ループによる逆順出力 */
    printf("data[%d] : %d¥n",i,data[i]);
    i--;
  }
}
```

```
std1dc1{s1000000}1: gcc lec07-5.c
std1dc1{s1000000}2: ./a.out
3 4 5
data[2] : 5
data[1] : 4
data[0] : 3
std1dc1{s1000000}3:
```

入力データは一旦配列**data**に蓄えられ、逆の順で出力される

/home/course/prog0/public_html/2006/lec/source/lec07-5.c

サンプル(2) ヒストグラム

```
#include <stdio.h>
#include <stdlib.h> /*乱数を使用するため必要*/
main()
{
  int histogram[11],i,ransu;

  for(i = 1 ; i <= 10 ; i++){
    histogram[i] = 0;/* ヒストグラム初期化 */
  }

  for(i = 0 ; i < 100000 ; i++){
    ransu = rand()*10/(RAND_MAX + 1) + 1;
                    /* 1から10までの乱数発生 */
    histogram[ransu]++;     /*ヒストグラム加算 */
  }

  for(i = 1 ; i <= 10 ; i++){/* ヒストグラム表示 */
    printf("%2d : %6d\n",i,histogram[i]);
  }
}
```

要素0は使わないので初期化しない

10進整数を6桁で表示

乱数の出現頻度を調べる

```
std1dc1{s1000000}1: gcc lec07-6.c
std1dc1{s1000000}2: ./a.out
 1 :   9939
 2 :   9878
 3 :   9902
 4 :  10031
 5 :   9916
 6 :  10115
 7 :   9987
 8 :  10077
 9 :  10166
10 :   9989
std1dc1{s1000000}3:
```

出現確率9.939%

例えば発生した乱数が**6**なら**histogram[6]**という要素をインクリメントすることで、出現回数をカウントする

/home/course/prog0/public_html/2006/lec/source/lec07-6.c

サンプル(3) 日数計算

```
#include <stdio.h>
main()
{
  int mdata[13];
  int month,day,i,total_days = 0;

  /* 各月の日数 */
  mdata[0]  = 0;  /* 要素0のデータは使わない */
  mdata[1]  = 31; mdata[2]  = 28; mdata[3]  = 31; mdata[4]  = 30;
  mdata[5]  = 31; mdata[6]  = 30; mdata[7]  = 31; mdata[8]  = 31;
  mdata[9]  = 30; mdata[10] = 31; mdata[11] = 30; mdata[12] = 31;

  while(1){/* 無限ループ */
    printf("知りたい日の月日を空白で分けて入力してください -> ");
    scanf("%d%d",&month,&day);
    /* 日付チェック */
    if((day > 0)&&(day <= mdata[month])&&(month > 0)&&(month <= 12)) break;
    printf("日付がおかしいので、再度入力してください\n");
  }
  for (i = 1 ; i < month ; i++) total_days += mdata[i]; /* 前月までの日数計算 */
  total_days += day;          /* 前月までの日数に今月の日数を加算して合計の日数を算出 */
  printf("\n%d月%d日は1年のうちで%d番目の日です\n",month,day,total_days);
}
```

月日を入力したら、1年のうち何番目の日か出力する

```
std1dc1{s1000000}2: ./a.out
知りたい日の月日を空白で分けて入力してください -> 1 35
日付がおかしいので、再度入力してください
知りたい日の月日を空白で分けて入力してください -> 13 1
日付がおかしいので、再度入力してください
知りたい日の月日を空白で分けて入力してください -> 6 2

6月2日は1年のうちで153番目の日です
std1dc1{s1000000}3:
```

このプログラムは簡単のためうるう年は考えない

月が1から12の範囲、日付が各月の日数の範囲じゃないと再入力

/home/course/prog0/public_html/2006/lec/source/lec07-7.c

サンプル(4) database

```c
#include <stdio.h>
main()
{
  int  id[3],result[3],i,man,found;
  /* 登録されている学生番号と成績 */
  id[0] = 1002345; result[0] = 89;
  id[1] = 1002498; result[1] = 64;
  id[2] = 1002984; result[2] = 93;

  while(1){ /* 無限ループ */
    printf("Enter student ID ->");
    scanf("%d",&man);
    if(man == 0) break; /* 0を入力されるとループ終了 */
    found = 0;   /* 発見したかどうか記憶する変数初期化 */

    for (i = 0 ; i < 3 ; i++){  /* 全データと照合 */
      if(man == id[i]){         /* 発見した！ */
        found = 1; /* 発見したことを記憶してforループ脱出 */
        break;
      }
    }
    if(found == 0) printf("   Not found\n");     /* 発見出来ない場合 */
    else printf("   Found score : %d\n",result[i]);  /* 発見した場合 */
  }
  printf("   Bye!\n"); /* 終了 */
}
```

```
std1dc1{s1000000}2: ./a.out
Enter student ID ->10000
   Not found
Enter student ID ->1002498
   Found score : 64
Enter student ID ->1002984
   Found score : 93
Enter student ID ->0
   Bye!
std1dc1{s1000000}3:
```

入力された学生番号がデータにあれば表示、なければその旨を表示する

/home/course/prog0/public_html/2006/lec/source/lec07-8.c