問題提起その 1

一文字ずつ文字(数字)を読み込み、それぞれの文字が何回入力されたかを数えて出力するプログラム。

```
int code, count_0=0, count_1=0, count_2=0,
 count_3=0, ...., count_9=0;

while( (code=getchar()) != EOF ){

  switch(code){
    case '0': count_0++; break;
    case '1': count_1++; break;
    ....
    case '9': count_9++; break;
  }
}

printf("0 は %d 個入力された\n",count_0);
printf("1 は %d 個入力された\n",count_1);
....
```

同じ型の変数を 10 個用意してプログラムするのは大変!

問題提起その 2

20 個の整数値(もしくは実数値)を読み込んで、昇順に並び替えたい。

考え方

- 20 個の変数を宣言して数値を格納(記憶)する。
- 変数の並び替え(ソート)

```
int a0, a1, a2, ..., a19;

scanf("%d", &a0);
scanf("%d", &a1);
....
```

20 個の値の読み込み

変数を 20 個宣言するのは大変である。また、記憶する数値の個数がさらに増えると値を格納する変数を用意するだけでも大変。

配列

同じ型のデータの集まりを配列という。

10 個の要素からなる
整数型の配列 a を宣言

```
int a[10];
```

配列名の後の [ ] の中に添え字が入る。添え字は整数型。
配列の要素は添え字で指定する。

添え字は 0 から始まる。i 番目の要素は a[i] で表す(i = 0, 1, 2, ... , 9)。

同じ型で構成される複数のデータをまとめたものが配列。

これまで扱ってきた変数:

整数型の変数 a を宣言　　`int a;`　　a

配列 2

`int a[10];`　は、10 個の整数型の要素を持つ配列を宣言

10 個の要素を持つ配列の場合、許される添え字は 0 から 9 までの 10 個である。

```
a[0] = 3;
a[3] = 7;
```

0 番目の要素に 3 を代入。
3 番目の要素に 7 を代入。

上の配列 a の場合、a[10] や a[11] という添え字は範囲外参照となり、おかしな結果を招く(コンパイルエラーにならないので注意が必要)。
当然 a[-1] という参照も範囲外参照。

```
a[10] = -4;
```

a[10] は存在しないので不可!(コンパイルはできるので注意)

配列 3

配列の要素数は、コンパイル時に確定していなければならない。

```
int size, a[size], i;

scanf("%d", &size);

/* 配列の初期化 */
for(i=0; i<size; i++)
  a[i] = 0;
```

配列要素数がコンパイル時に確定していないので不可（コンパイルエラー）

配列サイズ（要素数）は静的。動的にはとれない。

コンパイル時に、コンパイラは配列要素を格納するメモリ領域を確保するため。

配列 4

配列を宣言しただけでは、配列要素の中身は不確定（変数と同じ）。

```
int a[10], i;

/* 配列の初期化 */
for(i=0; i<10; i++)
  a[i] = 0;
```

要素を指定した配列は、今まで通りの変数と同じ。a[i] は i 番目の要素。

もしくは、下のように、宣言と同時に初期化も可能。

```
int a[10] = {1,2,3,4,5,6,7,8,9,10};
```

配列は、int 型以外にも float, double, char 等の型について使用できる。

```
double x[100], y[100];
char code[256];
```

例 1

文字定数 '0' ~ '9' の出現回数を数える。

```
int code, count[10], i;

for(i=0; i<10; i++) /* 配列要素の初期化 */
  count[i]=0;

while( (code=getchar()) != EOF ){

  switch(code){
    case ‘0’: count[0]++; break;
    case ‘1’: count[1]++; break;
    ....
    case ‘9’: count[9]++; break;
  }
}

for(i=0; i<10; i++)
  printf(“%c は %d 個入力された\n”, i+’0’, count[i]);
```

count[0] は ‘0’ の入力回数を数える
count[1] は ‘1’ の入力回数を数える
…

例 2

switch 文を使わない例

```
int code, count[10], i;

for(i=0; i<10; i++) /* 配列要素の初期化 */
  count[i]=0;

while( (code=getchar()) != EOF ){

  if( code >= ‘0’ && code <= ‘9’ )
    count[ code-’0’ ]++;

}

for(i=0; i<10; i++)
  printf(“%c は %d 個入力された\n”, i+’0’, count[i]);
```

添え字が code-’0’ になっていることに注意。if 文の条件判断で、読み込んだ文字が 0 から 9 の間にあることを保証しておかないと範囲外参照になる。

例 3

10 個の数値を配列へ格納（記憶）する。

```
int i, data[10];

for(i=0; i<10; i++) /* 配列へデータを読み込む */
  scanf("%d", &data[i]);

for(i=0; i<10; i++)
  printf("%d 番目のデータは %d\n", i, data[i]);

 /* 配列 data に格納された値を並び替える（ソート）*/
```

配列を使うと、複数のデータの取り扱い（入出力など）が易しくなる。

多次元配列

`int a[5];`　　1 次元配列

a[0]	a[1]	a[2]	a[3]	a[4]

`int b[4][3];`　　2 次元配列

4 行 3 列の行列に相当

12 個の要素からなる 2 次元配列

b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]
b[2][0]	b[2][1]	b[2][2]
b[3][0]	b[3][1]	b[3][2]

一般に、b[M][N] は
M 行 N 列の行列に相当

b[i][j] は (i, j) 成分　　2 つの添え字で要素を指定。添え字の範囲に注意！

$0 \le i < 4, 0 \le j < 3$

多次元配列の初期化

配列の宣言と同時に配列要素を初期化できる。

```
int a[2][3] = {{1,2,3},{2,3,4}};
int b[2][3] = {{0,0,0},{-1,-2,3}};
int c[2][3], i, j;

for(i=0; i<2; i++)
  for(j=0; j<3; j++)
    c[i][j] = a[i][j] + b[i][j];
```

初期化数値が配列サイズを越えるとコンパイルエラー。
足りない分は int 0 で初期化。

```
double x[10][10][10];
```

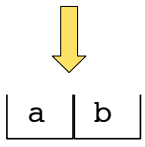
10*10*10=1000 個の要素からなる 3 次元配列

メモリ上の配列および変数の配置

配列の添え字の範囲はプログラマの責任。配列の範囲外参照はおかしな動作・結果を招く。

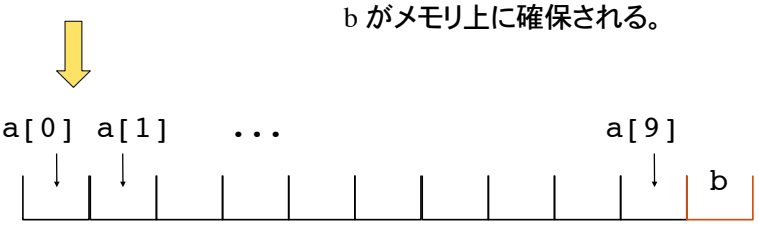
```
int a, b;
```

メモリ上に整数値を格納する 2 つの変数 a, b を確保。メモリ上のどの場所が確保されるかは処理系に依存（たいていは連続した位置に配置される）。




```
int a[10], b;
```

これにより 10 個の要素を持つ整数型の配列 a と整数型の変数 b がメモリ上に確保される。



正しい添え字の範囲は 0 から 9 まで!

ここで、a[10] = 1 の様な範囲外参照および代入をすると、隣に確保された変数 b の領域を侵害して変数 b の値がおかしくなってしまう。

オブジェクト

変数や配列など、計算機のメモリ上に格納されるものをオブジェクト object と呼ぶ。オブジェクトが占めるメモリ領域をオブジェクトサイズという。

オブジェクトサイズを知るには sizeof 演算子を用いる。

sizeof 式 または sizeof(型名)

sizeof 演算子は、式もしくは型名の変数に割り当てられるオブジェクトサイズを返す。単位はバイト。

```
int a, b[5];
double x, y[5];

printf("sizeof a=%d, b=%d\n, sizeof(int), sizeof b);
printf("sizeof x=%d, y=%d\n, sizeof(double), sizeof y);
```

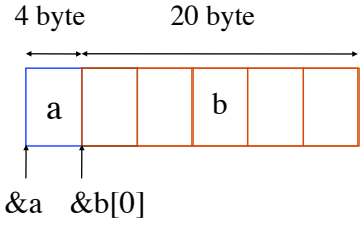
一般的な型のオブジェクトサイズ（処理系により異なる場合がある）

sizeof(int)	4 バイト = 256 * 256 * 256 * 256 = 4,297,967,296
sizeof(double)	8 バイト
sizeof(char)	1 バイト

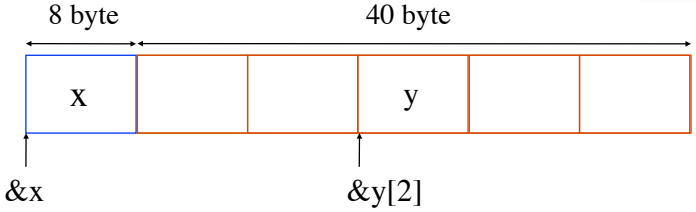
```
int a, b[5];
double x, y[5];
```

左のように宣言された変数・配列は下図のメモリ領域を占める。具体的な格納場所（アドレス）は処理系に依存。

オブジェクトの格納場所はアドレス演算子 & を用いて知ることができる。




```
printf("%x\n", &a);
printf("%x\n", &b[0]);
printf("%x\n", y);
```

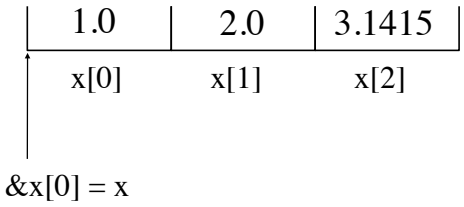


変換指定 %x : 16 進数表示

配列のアドレス

```
double x[]={1.0, 2.0, 3.1415};
```

配列を宣言と同時に初期化。要素数を省いて宣言すると、初期値の個数（この場合 3）が要素数となる。



配列名は、その配列が格納されているアドレスを表す。
（0 番目の要素のアドレスに等しい。配列名は配列へのポインタ）

```
printf("Addres of score[0] is %x\n", &x[0]);
printf("Addres of score is %x\n", x);
```

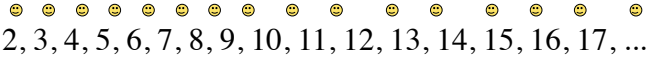
& は不要！

エラトステネスのふるい

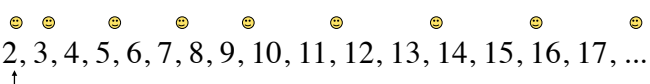
N 以下の素数を求めるアルゴリズムとして、エラトステネスのふるいがある。

アルゴリズム

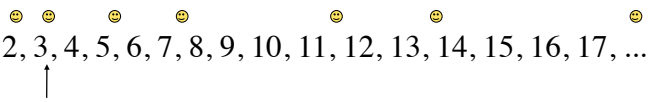
◎ 2 から *N* までの数を並べる。全ての数に印（例えば int 1 ）を割り振っておく。
◎ 2 の倍数（2 自身は除く）に印（例えば int 0）を割り振る。
◎ 印 1 が着いている最小の数は 3 である。3 の倍数（3 自身は除く）に印（0）を割り振る。
◎ 印 1 が着いている最小の数は 5 である。5 の倍数（5 自身は除く）に印（0）を割り振る。
◎ 印 1 が着いている最小の数は 7 である。7 の倍数（7 自身は除く）に印（0）を割り振る。
...
◎ 以上の操作を繰り返す。最後に印 1 が着いている数が素数である。



全ての数に印を付ける



2 を除く偶数（2の倍数）を候補から除く



奇数かつ 3 の倍数は素数ではないので 3 の倍数を除く

問題 1

キーボードから 10 人の成績（100 点満点の整数値）を配列に読み込み、1) 平均点、2) 分散、及び 3) 標準偏差を計算・表示するプログラム

```
% ./a.out
学籍 No. 0 の成績：80
学籍 No. 1 の成績：70
学籍 No. 2 の成績：95
...
学籍 No. 9 の成績：40
平均点は 84.3 点です。
分散は 237.16 です。
標準偏差は 15.4 点です。
%
```

データ $\{n_1, n_2, n_3, n_4, n_5, ..., n_k\}$ の

平均は $\langle n \rangle = \frac{1}{k} \sum_{i=1}^{k} n_k$

分散は $Var[n] = \frac{1}{k} \sum_{i=1}^{k} (n_k - \langle n \rangle)^2$

標準偏差は $\sqrt{Var[n]}$

問題 2

キーボードから 10 人の成績（100 点満点の整数値）を配列に読み込み、1) 最高点、及び 2) 最低点を表示するプログラム

```
% ./a.out
学籍 No. 0 の成績：80
学籍 No. 1 の成績：70
学籍 No. 2 の成績：95
...
学籍 No. 9 の成績：40
最高点は 100 点です。
最低点は 10 点です。
%
```

考え方：
1) 10 人の成績を配列に読み込む。
2) その中で最高点を探し出して表示。
3) 最低点を探し出して表示。

最高点の検索方法

i = 0 として score[i] を仮の最高点 max とする。
i++ として、score[i] と max の大小を比較。
以上を繰り返す。

問題 3

キーボードからアルファベットを 1 文字ずつ読み込み、入力したアルファベットの数を数えるプログラム。入力の中断は Ctrl-D とする。

ただし、大文字と小文字の区別はしない。

```
% ./a.out
アルファベットを入力（Ctrl-D で終了）
We had the world cup last year. [Enter]
[Crtl-D]
a は ? 文字。
b は ? 文字。
...
z は ? 文字。
%
```

26 個のアルファベットの出現回数を数える配列を用いる。

```
int count[26];
```

問題 4

3 行 3 列の行列を入力し、その積を計算するプログラム。
一般に行列の積に関しては AB とBA は等しくないことを示せ。

```
% ./a.out
行列 A の成分を入力せよ。
a[0][0] = 3
a[0][1] = 4
a[0][2] = 5
a[1][0] = -3
a[1][1] = 0
...
行列 B の成分を入力せよ。
b[0][0] = 1
b[0][1] = -6
...

行列 A B は
 3 4 5
-3 0 4
 1 0 3

行列 B A は
 0 3 1
 0 2 0
-1 2 1
%
```

n × n 行列 $A = (a_{ij}), B = (b_{ij})$

について、積 C = AB の (i, j) 成分は

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

考え方：

3 行 3 列（成分は 9 個）を入力して
行列の積を計算する。

行列の積 AB と BA を表示する。

問題 5

下例で使用する変数・配列のオブジェクトサイズ、および、変数と配列要素の値が占めるアドレスを表示せよ。

```
int a, b[5];
double x, y[5];
char c, d[5];

...
```

```
% ./a.out
オブジェクトサイズ: 4, 20
オブジェクトサイズ: 8, 40
オブジェクトサイズ: 1, 5
アドレス: bffffcb0, bffffcb0, ...
アドレス: bffffeb0, bffffeb0, ...
アドレス: bffffbb0, bffffbb0, ...
%
```

プログラムの出力

オブジェクトの格納場所(アドレス)を表示する際に、変換指定子 x を用いよ。
書式の変換指定 %x は、値を 16 進数で表示。

```
printf("%x", &a)
```

オブジェクト a のアドレス表示(16 進数)

問題 6

エラトステネスのふるいを用いて、1000 以下の素数を全て表示せよ。
ただし表示形式として、1 行に 10 個ずつ表示されるようにすること。

```
% ./a.out
  2  3  5  7 11 13 17 19 23 29
31  37 ...


947 953 967 ...  983 991 997
%
```

ヒント

int a[1001] を宣言　素数の候補に印 int 1 を付ける。

a[0] = 0, a[1] = 0, a[2] = 1, a[3] = 1, a[4] = 1, ...

印 1 が着いている最も小さな添え字は 2。
2 の倍数は素数ではないので、4, 6, 8, ... に
印 0 をつける。
a[4] = 0, a[6] = 0, a[8] = 0, a[10] = 0, ...

印 1 が着いている最も小さな添え字は 3。
3 の倍数は素数ではないので、6, 9, 12, ... に
印 0 をつける。
a[6] = 0, a[9] = 0, a[12] = 0, a[15] = 0, ...

印 1 が付いている、値 start 以降で最も小さな添え字の探索方法

```
i = start;
while( a[i]==0 ) i++;
```