配列

いくつもの変数につづけてアクセスするときには配列を使うと便利である．数学のベクトルや行列のようなものと考えればいい．

配列の宣言と初期代入

配列を使用する場合には，最初に配列名と次元と要素数を宣言しておく必要がある．int 型で 1 次元で要素数 5 の配列を，初期化してプリントすると

```
int a[5]={1,2,3,4,5};
for (i=0;i<5;i++){
   printf("%3d",a[i]);
}
```

普通のベクトルや行列の記述と異なり，C 言語の配列では添字が 0 から始まる．したがって最大添え字は 4 である．C 言語の慣例では，例えば 2 次元の配列は

```
double b[3][3];
for (i=0;i<3;i++){
   for (j=0;j<3;j++){
      scanf("%lf",&b[i][j]);
   }
}
```

である．しかし，

```
double b[N+1][N+1];
for (i=1;i<=N;i++){
   for (j=1;j<=N;j++){
      scanf("%lf",&b[i][j]);
   }
}
```

として b[1][1] から始めるのも「あり」．実際，数値計算のバイブルとされている”Numerical Recipe in C”という本ではこのように 1 から始まる添え字が使われている (1 オフセットと称する) [1]．0 オフセットを使うか，1 オフセットを使うかは問題やアルゴリズムにあったものを選べばよい．ただし，0 オフセット以外を使うときは，どういう意図でルールを破るかをコメントしておく必要はある．他人のプログラムを読むときは注意．

練習問題

1. `double a[5]={1,2,3,4,5};`

   として，a[0] から a[4] までの和を求めよ．

[1]ただし，ここで例示した b[N+1][N+1] というのは必要メモリーの観点から無駄が多い．プログラミング技能が上がれば「動的メモリー割当」というのを使うが，今は気にしなくても良い．

2. 100 までの素数をすべて求めて出力するプログラムを作れ．

   前に作った素数判定プログラムに整数を順に代入して調べるのはロスが大きい．**エラストテネスのふるい**が比較的簡単．初めに 0 から 100 までの配列を用意し 0 で初期化する．これを番兵と見なして，まず 2 の倍数 (2 自身は除く) に全部印をつける．次に，3 の倍数 (3 自身は除く) に全部印をつける．こうして，100 まで繰り返し，印がつかないで残っている数は「どの数の倍数でもない数」であるから素数である．

表 1: 番兵 (配列) の変化の様子．

要素	1	2	3	4	5	6	7	8	9	10	···	98	99	100
初期値	0	0	0	0	0	0	0	0	0	0	···	0	0	0
2 が終了	0	[0]	0	1	0	1	0	1	0	1	···	1	0	1
3 が終了	0	0	[0]	1	0	1	0	1	1	1	···	1	1	1

3. **双子の素数** (余力があれば) p が素数で $p+2$ も素数のとき，これらは双子の素数と呼ばれる．10 以上，1000 以下の双子の素数を全部見つけて出力せよ．

4. **ゴールドバッハの予想** (余力があれば) 「6 以上の偶数は二つの素数の和として表わされる」という予想を 100 以下の偶数について検証せよ．

練習問題 1 解答例

要素数 (境界，bound とも呼ばれる) を定義する方法として enum を用いている．

```
#include <stdio.h>
enum {N=5}:
//const int N=5;

int main(void){
  double a[N]={1,2,3,4,5};
  double total;
  int i;
  total=0;
  for (i=0;i<N;i++){
    total += a[i];
  }
// output
  printf("\nSum a[i] = %f \n",total);
  return 0;
}
```

要素数を定義する方法として#define もよく用いられてきた．1 オフセットを意図している．

```
#include <stdio.h>
#define N 5
```

```
int main(void){
  double a[N+1]={0,1,2,3,4,5};//unit offset
  double total;
  int i;
  total=0;
  for (i=1;i<=N;i++){
    total += a[i];
  }
// output
  printf("\nSum a[i] = %f \n",total);
  return 0;
}
```

#define はプリプロセッサ (前処理) を使っており，バグりやすい．上に示した const あるいは enum が「ただしい」書き方らしい．このような「ただしい」書き方が気になりだしたら「プログラミング作法」(Kernighan & Pike 著，アスキー 2000) を参照せよ．