4 Input and Output Functions

4.1 printf() and scanf() statements

- `printf()` functions

```
printf("control string", arguments);
 cf. Arguments can be omitted.

eg.

printf("output");
        output
printf("%c %d\n", 'A', 10);
        A 10
  %c, %d - numeric conversion specifier
```

variable-type	specifier	meaning
character	%c	character
	%s	string
integer	%d	decimal no.
	%o	octal no.
	%x	hexadecimal no.
	%u	unsigned no.
floating-point values	%f	floating point type
	%e	exponent type
	%g	shorter between %f and %e

cf. qualifiers - short : h, long : l, double : l, long double : L

- The number of conversion specifiers should be the number of outputs.

eg.

```
printf("%c %d\n", 65, 65);
        A 65
printf("%d", ‘A’-‘8’+‘5’);
        4
printf("%d %o %x", 10, 10, 10);
        10, 12, a
printf("%f %e\n", 2.4, 2.4);
        2.400000 2.400000e+00
printf("%s", ‘‘string’’);
        string
```

- user defined output forms

integer	real number	string
%nd or %-nd	%n.mf or %-n.mf	%n.ms or %-n.ms

```
n                  : no. of spaces
-                  : starting from left
m(in real number) : no of floating points
m(in string)       : first m characters
```

eg.

```
printf("%d\n", 126);              : 126
printf("%10d\n", 126);            : #######126
printf("%-10d\n", 126);           : 126#######
```

eg.

```
printf("%f\n", 1234.56);          : 1234.56####
printf("%e\n", 1234.56);          : 1.23456#e+03
printf("%3.1f\n", 1234.56);       : 1234.6
printf("%10.3f\n", 1234.56);      : ##1234.560
printf("%10.3e\n", 1234.56);      : #1.235e+03
```

eg.

```
printf("%2s\n", "string");        : string
printf("%10s\n", "string");       : ####string
printf("%-10.5s\n", "string");    : strin#####
```

- `scanf()` functions

```
scanf("control string", arguments);
  - Arguments should be memory addresses of variables.
  - Conversion specifiers are same as printf().
```

eg.

```
char ch;
scanf("%c %d", &ch, &i);
```

cf. `&` operator: returns the memory address of assigned variable.

4.2 File Input and Output

- unix command :

```
a.out <in.file> out.file
```

- declaration of data type of file :

```
FILE *fp; (*fp : file pointer - the variable that indicate
                                the memory address of file)
```

- `fopen()` and `fclose()` functions

```
fopen("filename", "r");
                   r : read
                   w : write
                   a : append
```

- `fopen()` returns the memory address of file.

eg.

```
FILE *in;
in = fopen("test". r");
fclose(in)              : fclose(file-pointer)
                         - notifying the end of a task to the file
```

- Input, Output function related to files.

- character

```
char ch; FILE *fp;

ch = getchar();
ch = getc(fp);        : read one character in the FILE
                        which is indicated by fp.
ch = getc(stdin)      : 'stdin' - keyboard

putchar(ch);
putc(ch, fp);         : write 'ch' to the 'fp'
putc(ch, stdout);     : 'stdout' - screen
```

- string

```
char str[100]; FILE *fp;

gets(str);
fgets(str, max, fp);    : read the first max characters in
                          a string of the FILE indicated by fp
                          and assign to str.
puts(str);
fputs(str, fp);         : write the string to the FILE
                          indicated by fp
```

- printf & scanf

```
scanf("control-string", argument_list);
fscanf(fp, "control-string", argument_list);   : read from the FILE
                                                  indicated by fp
printf("control-string", argument_list);
fprintf(fp, "control-string", argument_list)   : write to the FILE
                                                  indicated by fp
```

eg.

```
main()
{
     FILE *fp;
     int age;
     fp=fopen("sam", "r");
     fscanf(fp, "%d", &age);
     fclose(fp);

     fp=fopen("sam", "w");
     fprintf(fp, "sam is %d years old\n", age);
     fclose(fp)
}
```

eg.

```
if(fp==NULL)
   printf("error");
else{
  ...
```

- Programming Linear Modeling (Regression)

the process that determines the linear equation that is the best fit to a set of data points in terms of minimizing the sum of the squared distances between the line and the data points.

For the given sample :

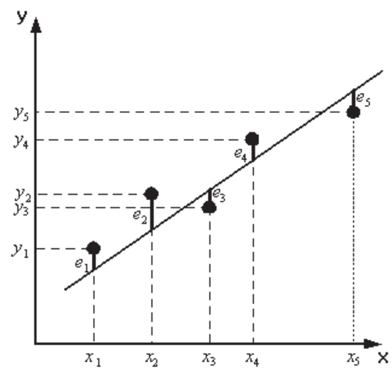
$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

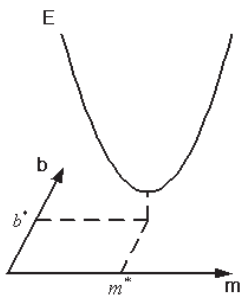
linear model(estimator) :

$$\hat{y}(x) = mx + b$$

problem : for the given sample and estimator, find m and b which minimizes

$$E = \frac{1}{2} \sum_{k=1}^{n} (y_i - \hat{y}(x_i))^2.$$





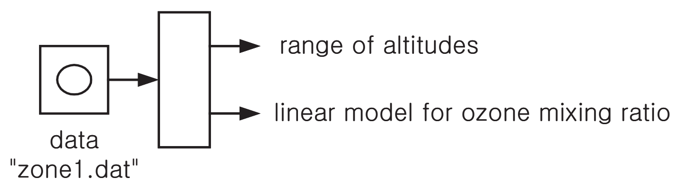
If $\hat{y}$ is linear, E has a quadratic form.

$$\left[\frac{\partial E}{\partial m}\right]_{m=m^*} = 0, \qquad \left[\frac{\partial E}{\partial b}\right]_{b=b^*} = 0$$

$$\begin{aligned} m^* &= \frac{\sum_{k=1}^n x_k \sum_{k=1}^n y_k - n \sum_{k=1}^n x_k y_k}{(\sum_{k=1}^n x_k)^2 - n \sum_{k=1}^n x_k^2} \\ b^* &= \frac{\sum_{k=1}^n x_k \sum_{k=1}^n x_k y_k - \sum_{k=1}^n x_k^2 \sum_{k=1}^n y_k}{(\sum_{k=1}^n x_k)^2 - n \sum_{k=1}^n x_k^2} \end{aligned}$$

- ozone measurements progrmamming

1. problem statement: for a given data (file),
   - determine a linear model for estimating the ozone mixing ratio at a specified altitude and
   - range of altitudes
2. I/O diagram



3. hand example

   - data: (ppmv: parts per million value)

altitude (x_k)	ppmv (y_k)
20	3
24	4
26	5
28	6

$$\sum_{k=1}^{4} x_k = 98, \quad \sum_{k=1}^{4} y_k = 18, \quad \sum_{k=1}^{4} x_k^2 = 2436, \quad \sum_{k=1}^{4} x_k y_k = 464$$

$$\text{denominator} = (\sum_{k=1}^{4} x_k)^2 - 4\sum_{k=1}^{4} x_k^2 = -140$$

$$m^* = (\sum_{k=1}^{4} x_k \sum_{k=1}^{4} y_k - 4\sum_{k=1}^{4} x_k y_k)/\text{denominator} = 0.37$$

$$b^* = (\sum_{k=1}^{4} x_k \sum_{k=1}^{4} x_k y_k - \sum_{k=1}^{4} x_k^2 \sum_{k=1}^{4} y_k)/\text{denominator} = -4.6$$

4. algorithm:
   (a) read data file
   (b) compute range of altitudes, and parameters(m & b) of linear model
   (c) print range of altitudes and linear model

- variables

```
FILE    fp
        x, y        : data file
float   sumx, sumy, sumx2, sumxy, denom, m, b
                    : linear model
        first, last : range
integer i, j, k
```

- C program

```
#include<stdio.h>
main()
{
   int i=0;
   float x, y, sumx=0, sumy=0, sumx2=0, sumxy=0,
         denom, m, b, first, last;
   FILE *fp;

   /*   read data file   */
   fp=fopen("zone1.dat","r");
   while((fscanf(fp,"%f %f", &x, &y))==2)
   {
      i++;
      if(i==1) first=x;
      sumx += x;
      sumy += y;
      sumx2 += x*x;
      sumxy += x*y;
   }
   last=x;
   fclose(fp);

   /*   compute m & b   */
   denom=sumx*sumx-i*sumx2;
   m=(sumx*sumy-i*sumxy)/denom;
   b=(sumx*sumxy-sumx2*sumy)/denom;

   /*   print range, m & b   */
   printf("range of altitudes in km: %.2f to %.2f\n", first, last);
   printf("linear model: m=%.2f, b=%.2f\n", m, b);
}
```