

## File Handling I: (Pure theoretical material)

Until now, all the input and output (I/O) process are done on screen.  
But, every time when we need to get the result, we must input the data again.  
It may be annoying when the data size is very large.

So, apart from screen, we can use other channel for I/O.  
For example, input from file, input from scanner, output to file, output to printer.  
We only focus on file I/O, which frequently occurs in OI questions.

Though, there are also several common file types such as text file, data file, bitmap ...,  
we will discuss text file and data file only.

Before we begin our discussion, I would like to raise some common misconceptions first.

First, to computer, **File extension** is not important and may be ignored. It only gives  
human beings a more convenience way to recognize its type.

Second, 1 byte is equal to 8 bits. Kb is not the same as KB.  
While Kb means kilo-bit, KB represents kilo-byte.

Third, in Physics, the SI-unit K, M, G mean  $10^3, 10^6, 10^9$  respectively.  
However, for computer, K is equal to  $2^{10}$ , M is  $2^{20}$ , G is  $2^{30}$ .  
Since Hz is a measurement of for number of revolution in 1 second, which is physics unit  
1GHz means  $10^9$  hertz.  
But, 1MB means  $2^{20}$  bytes because bit and byte is both computational unit.

Besides, unlike tape cassette which stores sound sequentially, computers stores a file  
parts by parts randomly.  
For example, for a 1MB-sized file, it may store 256KB on the top of hard-disk, and the  
remaining on the bottom of the hard-disk.

Finally, since a computer can only store binary numbers, there is no concept of file type  
to a computer. The major difference between a image file and a text file depends on the  
ways how a computer handles the file.

Therefore, it is necessary to use different handler for each different type.

*You may skip this page if you find difficulties.*

## File Handling II:

If you don't use screen I/O, the statement **uses wincrt**; is useless.

### Text file:

A text file is a file consists of characters only.  
It is readable for human.  
The most common extension is **.txt**

To handle a text file in Pascal, we use the text file handler type.

```
Var identifier : text;
```

**Example:**

```
Var f: text;
```

This statement declares *f* as a text file handler, which is prepared for I/O with text file.

### File Handling Procedure I:

```
Var f : text ; filename : string ; expr1 , expr2 : <any value> ;  
var1 , var2 : <any type>;
```

Procedure	Meaning
Assign(f, filename)	Assigns the name of an external file to a text file variable.
Rewrite(f)	Forced-creates and opens a new text file for output.
Reset(f)	Opens the text file <i>f</i> , put the cursor at beginning for input.
Append(f)	Open the file <i>f</i> , put the cursor at the end for output.
Write(f, expr1 ,expr2)	Output <i>expr1</i> and <i>expr2</i> to text file <i>f</i> .
Writeln(f, expr1 ,expr2)	Output <i>expr1</i> , <i>expr2</i> and a new line to text file <i>f</i> .
Read(f, var1 ,var2)	Input <i>var1</i> and <i>var2</i> from text file <i>f</i> .
Readln(f, var1 ,var2)	Input <i>var1</i> , <i>var2</i> and go to next line to text file <i>f</i> .
Close(f)	Finish using the file <i>f</i> . (Must use, otherwise data will lose)

Actually, the **write**, **writeln**, **read** and **readln** statement is almost the same as you learnt before, except that they *operate on a file instead of the screen*.

Here is example demonstrating the use of the above procedures.

```
var f:text;  
s:string;  
begin  
assign(f,'ex1.txt'); { Tell the computer that we use the text handler f to handle ex1.txt}  
rewrite(f);         { Create a new file called ex1.txt }  
writeln('Please enter some words :');  
readln(s);  
writeln(f,s);       { Store the word you typed to ex1.txt}  
close(f);           { Similar to save }  
writeln('Please check ex1.txt, it should store the words you just typed');  
end.
```

## File Handling III:

### Example:

```
var f:text;
    s:string;
begin
assign(f,'ex1.txt'); { Tell the computer that we use the text handler f to handle ex1.txt}
reset(f);           { Open the file ex1.txt for input}
readln(f,s);        {Read s from ex1.txt}
close(f);           { Similar to exit}
writeln("The words store in the file :");
writeln(s);         { Output the first line of ex1.txt}
end.
```

```
var f:text;
    s:string;
begin
assign(f,'ex1.txt'); { Tell the computer that we use the text handler f to handle ex1.txt}
rewrite(f);          { Create a new file called ex1.txt }
writeln('Please enter some words :');
readln(s);
writeln(f,s);        { Store the word you typed to ex1.txt}
writeln('Please check ex1.txt, nothing is there because I forgot to close(f)');
end.
```

```
var f1,f2:text;
    s:string;
begin
assign(f1,'ex1.txt');
reset(f1);
assign(f2,'ex2.txt');
rewrite(f2);
readln(f1,s);
writeln(f2,s);
close(f1);
close(f2);
end.
{You can compile and run this program without the statement uses winCRT; }
{Although the content of ex2.txt changed there is not any apparent result from the screen.}
{It's because this program does not use screen I/O}
```

## File Handling IV:

### File Handling Function I:

Var *f*: text;

Procedure	Meaning
Eoln	Check if the cursor is at the <b>end of line</b> of the screen.
Eoln( <i>f</i> )	Check if the cursor is at the <b>end of line</b> of the file <i>f</i> .
Eof	Check if the cursor is at the <b>end of file</b> of the screen.
Eof( <i>f</i> )	Check if the cursor is at the <b>end of file</b> of the file <i>f</i> .

### Example:

Count number of characters of 1<sup>st</sup> line in file **eoln.txt**

```
var f:text;    i:integer;    c:char;
begin
assign(f,'eoln.txt');
reset(f);
i:=0;
while not eoln(f) do
begin
read(f,c);
inc(i);
end;
writeln('There are total ',i,' characters in the first line.');
```

Count number of lines in file **eof.txt**

```
var f:text;    i:integer;
begin
assign(f,'eof.txt'); reset(f);
i:=0;
while not eof(f) do
begin
readln(f);
inc(i);
end;
writeln('There are total ',i,' lines in eof.txt.');
```

Count number of characters of each line.

```
var f:text;    count,i:integer;    c:char;
begin
assign(f,'eoln.txt'); reset(f);
i:=0;
while not eof(f) do
begin
begin
inc(i);
count :=0;
while not eoln(f) do
begin
read(f,c);
inc(count);
end;
writeln('Line ',i,' : ',count);
end;
end.
```

## File Handling V:

(All the data files can be found on the sub-folder **data**)

### Practical Exercises:

Input: pex1.txt  
Output: Screen

**Description:**

The first line of the file stores an integer N.  
The following n lines store an integer.  
Output the total sum of those N integers.

Input: pex5.txt  
Output: pex5o.txt

**Description:**

Output the sum of all integers in the file.

Input: pex2.txt  
Output: pex2o.txt

**Description:**

The first line of the file stores an integer N.  
The following n lines store an integer.  
Output the average of those N integers.

Input: pex6.txt  
Output: pex6o.txt

**Description:**

Remove all the lines starting with 'A'  
Store the result to pex6o.txt

Input: pex3.txt  
Output: pex3o.txt

**Description:**

The first line of the file stores an integer N.  
The following n lines store an integer.  
Output the HCF of N integers.

Input: pex7\_1.txt , pex7\_2.txt  
Output: pex7o.txt

**Description:**

First line of each file stores an integer N.  
Means total N integers stored in asc. order.  
Store those integer to pex7o.txt in asc.  
order. ( $N \leq 1000$ )

Input: pex4.txt  
Output: pex4o.txt

**Description:**

The first line of the file stores an integer N.  
The following n lines store an integer.  
Output the LCM of N integers.

Input: pex8.txt  
Output: pex8o.txt

**Description:**

Check each integer in the file is prime or not.

## Exercise for File Handling:

(Hand in on 13-Jul)

### Exercise I:

2000hje.doc	32 - 34
2001hje.doc	1
2003hje.doc	B6

### Exercise II:

1999hje.doc	34
1999hse.doc	14 , 22 – 23* , 30
2000hse.doc	10

### Exercise III: (Hand in on 18-Jul)

2003fse.doc	Question 0 Enumeration
2003fje.doc	Question 3 Goldbach's Conjecture

*Please study all notes before next lesson.*

### Time-table:

Date	Time
13/7, 20/7, 27/7, 3/8,10/8,17/8,24/8	10:00 ~ 13:00
18/7, 25/7, 1/8, 8/8, 15/8, 22/8	14:00 ~ 17:00

### Trainers:

Name	Contact	Job
Wu Kai Chiu, Keith (7s)		Teaching and making notes
Kwok Yuk Hang, Sonic (6s)		Teaching, Helpers
Lo chi yip, Curtis (Graduated)		Helpers