

Chapter 3 Input and Output

Ground Rules

- Switch off your handphone and pager
- Switch off your laptop computer and keep it
- No talking while lecture is going on
- No gossiping while the lecture is going on
- Raise your hand if you have question to ask
- Be on time for lecture
- Be on time to come back from the recess break to continue the lecture
- Bring your lecture notes to lecture

1

3.1 Disk File

- Interactive mode of operation is fine for small program
- Use disk files input and output for large program as it is less costly to re-run

Basic Operations

- 1) naming a file
- 2) opening a file
- 3) reading data from a file
- 4) writing data on a file
- 5) closing a file

2

3.2 High Level File I/O Functions

fopen()	create a new file, or open an existing file
fclose()	close a file
fprintf()	write a set of data values to a file
fscanf()	read a set of data values to a file

3

3.3 Defining and Opening a File

A stream (flow of data) is declared as a pointer to an object of type **FILE**.
format:

FILE *stream;

- a) The file pointer stream communicates between the program and the system.
 - b) The pointer contains all the information about the file.
 - c) Mode can be one of the following:
 - "r" : open the file for reading only
 - "w" : open the file for writing only
 - "a" : open the file for appending only
 - "t" : text file
 - "b" : binary file
- Note the double quotation marks, they are string, not character.
 - What is string and what is character ?
 - We are concerned with "w", "r", "t", only.

4

Example:

```
FILE *indata, *outdata;
```

```
indata = fopen ("a:newdata.inf", "r");
```

Opens the file **newdata.inf** stored in **drive A** for input and stores its stream pointer under the name **indata**.

```
outdata = fopen ("a:olddata.ouf", "w");
```

Opens the file **olddata.ouf** stored in **drive A** for output and stores its stream pointer under the name **outdata**.

3.4 Closing a file

```
fclose (filePtr);
```

Example:

```
FILE *indata, *outdata;
```

```
/* file processing */
```

```
.
```

```
fclose (indata);
```

```
fclose (outdata);
```

The closing of an output file causes any data remaining in the buffer to be written to the file. A file that has been closed can be reopened for further processing with the **fopen** function.

5

3.5 Input and Output for Arbitrary Streams

- **scanf()** reads from the standard input stream **stdin** (usually refer to keyboard).
- **printf()** writes to the standard output stream **stdout** (usually refer to monitor screen).
- **fscanf()** and **fprintf()** read from or write to the stream specified by the first argument of the function respectively.
- Stream can from any I/O device. We assume disk file.

3.6 EOF

How does a computer operating system remember where each file begins and ends?

- EOF is a constant and its value is -1 in most implementation. This is basically a file marker.
- **fscanf** reads the contents of a file. It returns - 1 (EOF) if it attempts to read at the end of the file.

6

/* demo33.c */

```
/* copy from indata file to outdata file */
#include <stdio.h>
void main ()
{
    FILE *indata, *outdata;
    char this1;
    if ((indata = fopen ("f:demo33.inf", "r")) == NULL)
    {
        printf ("Can't open demo33.inf");
        exit (1);
    }
    if ((outdata = fopen ("f:demo33.ouf", "w")) == NULL)
    {
        printf ("Can't open demo33.ouf");
        exit (1);
    }
    while (fscanf (indata, "%c", &this1) != EOF)
        fprintf (outdata, "%c", this1);
    fclose (indata);
    fclose (outdata);
}
```

Input file:

I have problem to remember the alphabetical order where m is before n or n is before m until a student tells me that monkey comes first. Now I can remember that m is before n.

Output file:

I have problem to remember the alphabetical order where m is before n or n is before m until a student tells me that monkey comes first. Now I can remember that m is before n.

7

/* demo33a.c */

```
/* copy from indata file to outdata file. Use of sub-directory */
#include <stdio.h>
void main ()
{
    FILE *indata, *outdata;
    char this1;
    if ((indata = fopen ("f:\\demo33.inf", "r")) == NULL)
    {
        printf ("Can't open demo33.inf");
        exit (1);
    }
    if ((outdata = fopen ("f:\\Lecture\\chapter3\\demo33a.ouf", "w")) == NULL)
    {
        printf ("Can't open demo33c.ouf");
        exit (1);
    }
    while (fscanf (indata, "%c", &this1) != EOF)
        fprintf (outdata, "%c", this1);
    fclose (indata);
    fclose (outdata);
}
```

8

Case Study:

Ah Huat sells three types of durian in NTUC Fair Price. The description of each type and profit per durian is given in the table below.

Type	Description	Profit/Durian
D11	Sweet	\$2.5
D24	Bitter	\$3.5
XO	Seedless	\$3.2

The durians sold are returnable by customers, and Ah Huat will have to refund them the same price they have paid. To computerize his durian business he has stored the daily sales in a text file named durian.inf as follow:

Each row of data is for one day of transactions. The text file contains all the business records in a month. The number of business days in each month is not predictable. Write a program to read the input file from drive a and sum up the profit for each day in the month.

No. of D11 durians sold
No. of D11 durians returned
No. of D24 durians sold
No. of D24 durians returned
No. of XO durians sold
No. of XO durians returned

80	5	60	4	90	5
76	4	30	8	75	8
88	12	40	16	50	10
90	10	50	8	70	6
74	7	80	3	60	12
82	12	62	14	90	5
86	8	50	7	79	4
82	15	67	6	67	13
87	8	97	12	93	10
78	16	82	16	80	7
72	17	72	5	78	13
82	12	57	1	85	14
88	9	98	3	79	13
87	2	90	19	91	13
65	5	63	12	56	15

9

```
#include <stdio.h>

main()
{
    FILE *indata;
    float profit, sum=0;
    int a1,a2, b1,b2, c1,c2;
    indata = fopen ("f:\\durian.inf", "r");
    if (indata != NULL)
    {
        while ( fscanf(indata,"%d%d%d%d%d%d",
            &a1,&a2,&b1,&b2,&c1,&c2)==6)
        {
            profit = 2.5*(a1-a2) + 3.5*(b1-b2)
                + 3.2*(c1-c2);

            sum += profit;
        }
        fclose (indata);
        printf ("\n Profit for the month = $%.2f",
            sum);
    }
    return 0;
}
```

80	5	60	4	90	5
76	4	30	8	75	8
88	12	40	16	50	10
90	10	50	8	70	6
74	7	80	3	60	12
82	12	62	14	90	5
86	8	50	7	79	4
82	15	67	6	67	13
87	8	97	12	93	10
78	16	82	16	80	7
72	17	72	5	78	13
82	12	57	1	85	14
88	9	98	3	79	13
87	2	90	19	91	13
65	5	63	12	56	15

durian.inf

Functions printf() and fprintf() return the number of characters written, if all went well, and a negative value if an error occurred.

Functions scanf() and fscanf() return the number of values that were assigned to variables, or EOF if no values were assigned.

Screen Output

Profit for the month = \$8895.50

10

Example. GST (goods and service tax) of 9% has been introduced in Singapore. GST, however, is not applicable to tourists. Therefore, the GSTs paid by tourists are refunded to them. A file named **gst.inf** contains the customer ID, unit price and quantity purchased in an electronic shop. Each row contains the purchase from one customer.

The total payment specified by the authority is calculated as follows:

$\text{payment} = \text{unit_price} \times \text{quantity};$
 $\text{GST} = \text{payment} \times 9\%;$
 $\text{total_payment} = \text{payment} + \text{GST};$

Customer ID beginning with a digit **7** indicates that the ID is for a tourist. Write a program named as **gst.c** to perform the calculation for each customer, and accumulate the sub-totals for payment, GST, and total payment. In **gst.c**, you should also calculate the total GST refundable for the input file, and print it on the last line of the output text file as shown. Your program will produce the text file named as **gst.ouf**.

ID	unit price	quantity
2410	15.20	50
7300	40.45	120
7000	60.20	100
3847	50.30	90
6999	60.20	40
6847	50.30	80
4552	45.70	20
7893	55.20	10

gst.inf

Customer ID	Payment (\$)	GST (\$)	Total Payment (\$)
2410	760.00	68.40	828.40
7300	4854.00	436.86	5290.86
7000	6020.00	541.80	6561.80
3847	4527.00	407.43	4934.43
6999	2408.00	216.72	2624.72
6847	4024.00	362.16	4386.16
4552	914.00	82.26	996.26
7893	552.00	49.68	601.68
Sub-Total: 24059.00 2165.31 26224.31			
Total GST refundable: \$1028.34			

gst.ouf

11

1 | gst.c

```
# include <stdio.h>
# include <stdlib.h>

main()
{
    FILE *indata, *outdata;
    int id, qty;
    float price, payment, total, gst, refund;
    float sump, sumg, sumt;
    indata = fopen ("e:\\lab55\\gst.inf", "r");
    outdata = fopen ("e:\\lab55\\gst.ouf", "w");

    if (indata==NULL)
    {
        printf ("\n gst.inf not exist ...");
        exit(1);
    }

    if (outdata==NULL)
    {
        printf ("\n gst.ouf cannot be opened ...");
        exit(1);
    }
}
```

ID	unit price	quantity
2410	15.20	50
7300	40.45	120
7000	60.20	100
3847	50.30	90
6999	60.20	40
6847	50.30	80
4552	45.70	20
7893	55.20	10

gst.inf

12

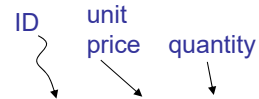
```

refund=sump=sumg=sumt=0;
fprintf(outdata, "\n Customer   Payment   GST   Total ");
fprintf(outdata, "\n   ID       ($)    ($)   Payment ($) ");
fprintf(outdata, "\n=====");

while (fscanf(indata, "%d%f%d", &id, &price, &qty) == 3)
{
    payment = price*qty;
    gst = payment*0.09;
    total = payment + gst;
    sump += payment;
    sumg += gst;
    sumt += total;
    fprintf(outdata, "\n   %d    %8.2f  %6.2f   %8.2f", id, payment, gst, total);
    if (id >= 7000 && id <= 7999) refund += gst;
}

fprintf(outdata, "\n=====");
fprintf(outdata, "\nSub-Total: %8.2f %6.2f   %8.2f", sump, sumg, sumt);
fprintf(outdata, "\n\n      Total GST refundable: $%5.2f \n", refund);
fclose(indata);
fclose(outdata);
return 0;
}

```



ID	unit price	quantity
2410	15.20	50
7300	40.45	120
7000	60.20	100
3847	50.30	90
6999	60.20	40
6847	50.30	80
4552	45.70	20
7893	55.20	10

gst.inf

Customer ID	Payment (\$)	GST (\$)	Total Payment (\$)
=====			
2410	760.00	68.40	828.40
7300	4854.00	436.86	5290.86
7000	6020.00	541.80	6561.80
3847	4527.00	407.43	4934.43
6999	2408.00	216.72	2624.72
6847	4024.00	362.16	4386.16
4552	914.00	82.26	996.26
7893	552.00	49.68	601.68
=====			
Sub-Total:	24059.00	2165.31	26224.31
=====			
Total GST refundable: \$1028.34			

gst.out