Introduction to C Programming

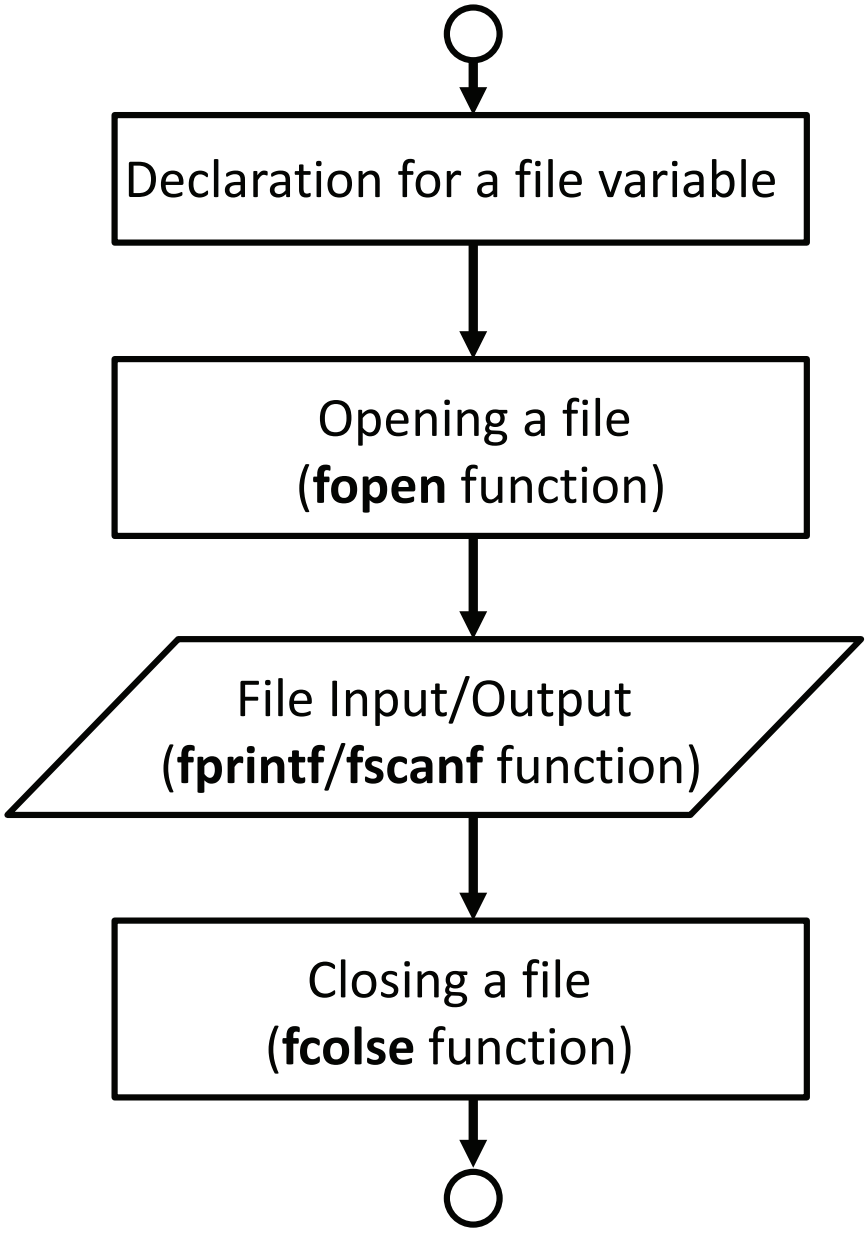
— File Input/Output —

Waseda University

Today's Topics

- Usage of the reading and writing of files.
  - Opening a file (**fopen** function).
  - Closing a file (**fclose** function).
  - How to write to a file (**fprintf** function).
  - How to read from a file (**fscanf** function).
  - Special characters and Conversions characters.
- Error handling
  - **exit** function

File Handling



FILE **file-variable*

fopen("*filename*","*mode*")

fprintf(*file-variable*,*format*,*parameter*)

fscanf(*file-variable*,*format*,&*parameter*)

fclose(*file-variable*)

File pointer

File pointer

```
FILE *file-variable;
```

- *file-variable* is a file variable which is declared by **FILE ***.
- *file-variable* is called the file pointer.
- **FILE *** and file functions are stored in standard include file **stdio.h**.

Opening/Closing a file

fopen function fopen("filename","mode")

```
FILE *file;    /* Declaration for a file variable */
file = fopen("filename","mode");
```

- Before a file can be used, a file must be opened using **fopen** function.
- *mode* indicates if the file is to be read or written. Example

mode	Description
w	Create an empty file for writing. Deletes content and overwrites if the file with the same name already exists.
r	Open for reading (The file must exist).

- If there is an I/O error, then the value NULL is returned.

fclose function

- **fclose** function close the file.

```
fclose(file pointer);
```

How to write to a file

fprintf function

`fprintf(`*file pointer*`,` *format*`,` *parameter1*`,` *parameter2*`, ...);`

- **fprintf** function converts date and writes it to a file of *file pointer*.
- Example:

```
int a=10;
FILE *file=fopen("test","w"); /* Creating a file */
fprintf(file, "%d¥t",a);      /* Writing ‘‘10 Tab’’ to the test file */
fprintf(file, "%d",a+1);      /* Writing ‘‘11’’ to the test file */
fprintf(file, "%d¥n",a*2);    /* Writing ‘‘20 New line’’ to the test file */
fclose(file);
```

Special characters

- In Japan, Yen sign ¥ is equivalent to the backslash character \.
- ¥ is called the escape character.
- ¥ is used to signal that a special character follows.

¥n	¥t	¥"	¥¥
Newline	Tab	Double quote	Yen (Backslash)

Conversions characters

- In *format* of **fprintf** function, % is used to signal that a conversions characters follows.

Example	Argument type / Displays
%d	int / decimal number
%f	double / [-]*m.dddddd* (, where *d*'s is given by the precision (default 6))
%e	double / [-]*m.dddddd* E ± *xx* (, where *d*'s is given by the precision (default 6))
%p	* / pointer
%c	int / single characters
%s	char * / string of characters
%x	int / unsigned hexadecimal number
%%	no / print a %
%5d	int / at least 5 characters wide, with leading blank space as necessary.
%05d	int /at least 5 characters wide, with leading 0 as necessary (e.g. 000mm).
%.2f	double / [-]*m.dd* 2 characters after decimal point.
%5.2f	double / at least 5 wide including decimal point and 2 after decimal point.
%+5.2f	double / includes sign, whether positive or negative.

Example 1

Example 1 : Write the following program

Write the program that write the multiplication table to a file.

```
1   2   3   4   5   6   7   8   9
2   4   6   8  10  12  14  16  18
3   6   9  12  15  18  21  24  27
4   8  12  16  20  24  28  32  36
5  10  15  20  25  30  35  40  45
6  12  18  24  30  36  42  48  54
7  14  21  28  35  42  49  56  63
8  16  24  32  40  48  56  64  72
9  18  27  36  45  54  63  72  81
```

- Let filename be “kukuout.c”.
- Make a text file “kuku” and write the multiplication table using Tab-delimited to “kuku” file.
- Use ”%2d¥ t” and ”¥ n” in the **fprintf** function.
- You should first write the program to display the multiplication table.

How to read from a file

fscanf function

fcanf(*file pointer*, *format*, *pointer list*)

- **fscanf** function is an input routine similar to **scanf** function that reads from a file.
- *file pointer* is a file opened for reading.
- **fscanf** function ignore any whitespace characters (space, tab, newline).
- Return value is the number of parameters successfully converted.
- You shuld open the file with read only mode ("r") using **fopen** function.

```
int a; double b;
FILE *file=fopen("test","r");  /*Opening the test file with read only mode/
fscanf(file, "%d",&a);  /*Reading an integer number and substituting to a*/
fscanf(file, "%lf",&b); /*Reading a real number and substituting to b*/
fclose(file);
```

Supplement

- C provide three pre-opened files
- **stdin** $\rightarrow$ Standard input (Keyboard)
  fprintf(stdin,"$\cdots$",$\cdots$) $\Leftrightarrow$ scanf("$\cdots$",$\cdots$)
- **stdout** $\rightarrow$ Standard output (Display)
  fprintf(stdout,"$\cdots$",$\cdots$) $\Leftrightarrow$ printf("$\cdots$",$\cdots$)
- **stderr** $\rightarrow$ Standard error (Display)
- Difference between **stdout** and **stderr** is related to the redirect of UNIX-base OS.

Error handing

exit function

- Program execution should be terminated if an error occurs.
- The **exit** function terminates program execution when it is called.
- The **exit** function requires `#include<stdlib.h>`
- Argument : Normally you write 1 or higher if something went wrong and 0 if everything went ok.

```
exit(1);
```

- `You should write an error message just before the execution of` **exit** `function using stderr with fprintf function.`

```
fprintf(stderr,"Error message") ;
```

Error handing

Fail to open a file

```
FILE *file = fopen("test","r");
if(file==NULL){                                    /*Cannot open file*/
    fprintf(stderr, "cannot open file 'test'");      /*Error message*/
    exit(1);                    /*Terminating of the program execution*/
}
```

Fail to read from file

```
double a; int scannum;
FILE *file ;
...
scannum = fscanf(file,"%lf",&a);              /*Reading a real number*/
if(scannum!=1){                                    /*Cannot read file*/
    fprintf(stderr, "cannot read file");             /*Error message*/
    exit(1);                    /*Terminating of the program execution*/
}
```

- You don't have to write the **fclose** function if the **exit** function execute.

Example 2

Example 2 : Write the following program

- Let filename be “kukuin.c”.
- Read a “kuku” file and display the multiplication table with true/false.
- If true/false, display -/ X after the number.

```
1-   2-   3-   4-   5-   6-   7-   8-   9-
2-   4-   6-   8-  10-  12-  14-  16-  18-
3-   6-   9-  12-  15-  18-  21-  24-  27-
4-   8-  12-  16-  20-  24-  28-  32-  36-
5-  10-  15-  20-  25-  30-  36X  40-  45-
6-  12-  18-  24-  30-  36-  42-  48-  54-
7-  14-  11X  28-  35-  42-  49-  56-  63-
8-  16-  24-  32-  40-  48-  56-  64-  72-
9-  18-  27-  36-  45-  54-  63-  72-  81-
```

- Change some numbers in “kuku” file to check your program.
- If the program cannot open the file or cannot read a value, then display an error message and **exit**.

Summary

- Usage of the reading and writing of files.
  - Opening a file (**fopen** function).
  - Closing a file (**fclose** function).
  - How to write to a file (**fprintf** function).
  - Special characters and Conversions characters.
  - How to read from a file (**fscanf** function).
- Error handling
  - **exit** function