

C Programming

Learning Package 8

I/O Operations and Files

Dr Scott Turner

School of Science and Technology

University of Northampton



**Introduction**

In this Learning Package, we are going to look at further functions for handling input and output operations.

**Learning outcomes**

At the end of this Learning Package, the reader should be able to:

* Use further features of the printf() and scanf() functions.
* Develop programs using files.

**Study guide**

Session 1:

Section 8.1

Task 8.1

Section 8.2

Task 8.2

Session 2:

Section 8.3

SAQ 8.1

Section 8.4

SAQ 8.2

Task 8.3

Session 3:

Exercises

* 1. **More on printf()**

Read from pages 206-211 from the sections labelled *9.1 Introduction* to the end of the section labelled *9.3 More on formatting using printf*() of the module textbook.

Task 8.1

* + 1. Type in program 9.1, compile, and execute it.
    2. Alter the program you type in for (a) so it displays the octal, hexadecimal and unsigned integer for any integer between 0 and 255.
    3. What happens if you type in a negative integer? Why do you think this happens?
  1. **More on scanf()**

Read from page 211-213 the section labelled *9.4 input formatting using scanf()* of the module textbook.

Task 8.2

Type in the following programs (a)

#include <stdio.h>

#include <stdlib.h>

int main()

{

char d[20];

printf("\nEnter your full name: "); gets(d);

printf("Hello %s\n\n",d); puts("Welcome");

system("PAUSE"); return 0;

}

(b)

#include <stdio.h>

#include <stdlib.h> int main()

{

char d[20];

printf("\nEnter your full name: "); scanf("%s",&d);;

printf("Hello %s\n\n",d); puts("Welcome");

system("PAUSE"); return 0;

}

Only one line has changed but the program (b) cannot store a string in d with a space within it when scanf is used.

* 1. **Files**

Read pages 213-216 the section labelled *9.5 File I/O* of the module textbook.

SAQ 8.1

Filling in the missing words

* + 1. The **k d** and **s n** are normally identified as the **v s** stdin and stdout respectively. The variable stdin means **s d i t** and the variable stdout means **s d o t**. These are **f e p s**.
    2. The first **p r** in the file versions of the functions fprintf() and fscanf() is a **p r** to a **f e**.
    3. Before **f e** operations can be performed and after a **f e p r** has been **d ed**, the file must be **o ned** for access, using the function

**f n**().

* + 1. The **f n** fopen() returns an **i r** value to indicate if it was successful. If opening a file was **u l** then the function returns **N L** or 0. With this knowledge we can test if the **f e** was opened successfully.
  1. **More file-handling**

Read page 216-226 from the section labelled *9.6 doing some I/O* of the module textbook.

SAQ 8.2

Fill in the following table

|  |  |
| --- | --- |
| **Mode** | **File is opened for** |
|  | Writing to a text file only |
| a |  |
|  | Reading and writing |
|  | Reading from a binary file only |
| wb+ |  |

SAQ 8.3

Fill in the missing letters

A **f e** might not exist when a file is to be **r d** or it might already exits when **w ing** to a file. It would be useful to identify that an **e r** has occur. This can be achieved in a limited way by noting the value of the **f e p ter** used to access the file. If the value **r ed** is a **N L** when

**o ing** a file to read, the file does **n\_t e t**, and the **p a m** can include a routine that would be performed on this value.

Task 8.3

1. Do program 9.2
2. Do program 9.4

**Exercises**

Do 1, 2, 3, 4, 5, 6.

**SAQ Answers**

SAQ 8.1

Filling in the missing words

1. The **keyboard** and **screen** are normally identified as the **variables** stdin and stdout respectively. The variable stdin means **standard input** and the variable stdout means **standard output**. These are **file pointers**.
2. The first **parameter** in the file versions of the functions fprintf() and fscanf() is a **pointer** to a **file**.
3. Before **file** operations can be performed and after a **file pointer** has been **defined**, the file must be **opened** for access, using the function **fopen**().
4. The **function** fopen() returns an **integer** value to indicate if it was successful. If opening a file was **unsuccessful** then the function returns **NULL** or 0. With this knowledge, we can test if the **file** was opened successfully.

SAQ 8.2

Fill the following table

|  |  |
| --- | --- |
| mode | File is opened for |
| w | Writing to a text file only |
| a | Appending |
| r+ | Reading and writing |
| rb | Reading from a binary file only |
| wb+ | Reading and writing operations on a binary file |

SAQ 8.3

Fill in the missing letters

A **file** might not exist when a file is to be **read** or it might already exits when **writing** to a file. It would be useful to identify that an **error** has occur. This can be achieved in a limited way by noting the value of the **file pointer** used to access the file. If the value **returned** is a **NULL** when **opening** a file to read, the file does **not exist**, and the **program** can include a routine that would be performed on this value.