SIES: BASIC C PROGRAMMING

L #15: STRUCTURES, FILE INPUT & OUTPUT

Seung Beop Lee

School of International Engineering and Science

CHONBUK NATIONAL UNIVERSITY

Outline

- **Structures**
- **File Input & Output**

Structures

Structures

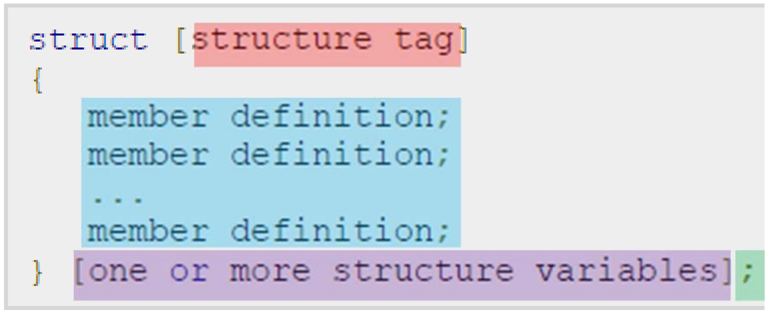
- Structures
  - **C arrays** allow you to define type of variables that can hold several data items of the **same kind**.
  - **C structure** is another user defined data type available in C programming, which allows you to combine data items of **different kinds**.
  - Generally, structures are used to represent a record.
  - Suppose you want to keep track of your books in a library.
  - You might want to track the following attributes about each book:

- Title
- Author
- Subject
- Book ID

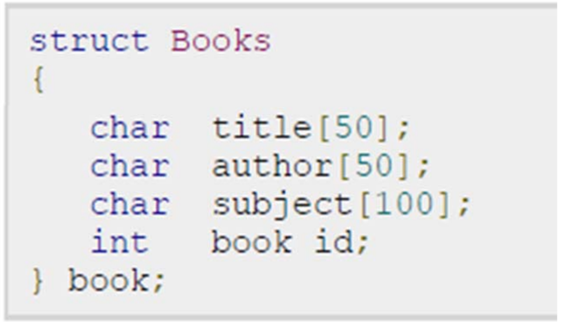
```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book id;
} book;
```

Structures

- Defining a structure
  - To define a structure, you must use the **struct** statement.
  - The **struct statement** defines a new data type (structure type), with more than one member for your program.
  - The format of the **struct statement** is this:



```
struct [structure tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```



```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

  - The structure tag is optional.
  - Each member definition is a normal variable or array definition.
  - One or more structure variables is optional.
  - Note that there is the final semicolon at the end of the structure's definition.

Structures

- Defining a structure
  - To define a structure, you must use the **struct** statement.
  - The **struct statement** defines a <u>new data type (structure type)</u>, with more than one member for your program.
  - The format of the **struct statement** is this:

```
struct [structure tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

- Declaration of the structure in functions
  - The syntax of the declaration of the structure is the same to that of the variable except the **struct**.

```
int len ;
```

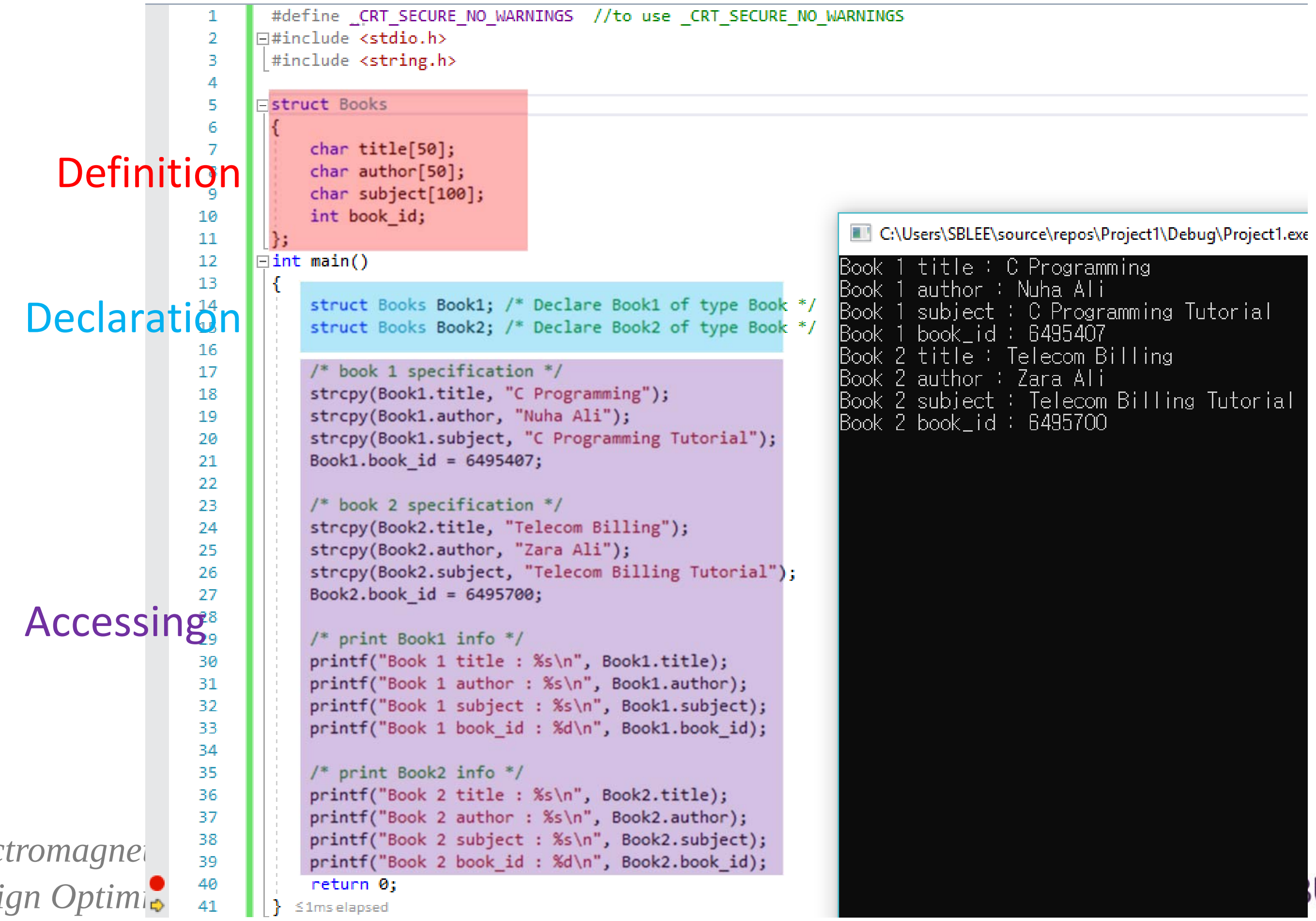
Data type | Variable name

```
struct Books Book1;        /* Declare Book1 of type Book */
struct Books Book2;        /* Declare Book2 of type Book */
```

Struct Structure type | Variable name

Structures

- Accessing structure members
  - To access any member of a structure, you must use the **member access operator** (**.**).
  - The **member access operator** is coded as a **period** (**.**) between the <u>structure variable name</u> and the <u>structure member</u> that you wish to access.



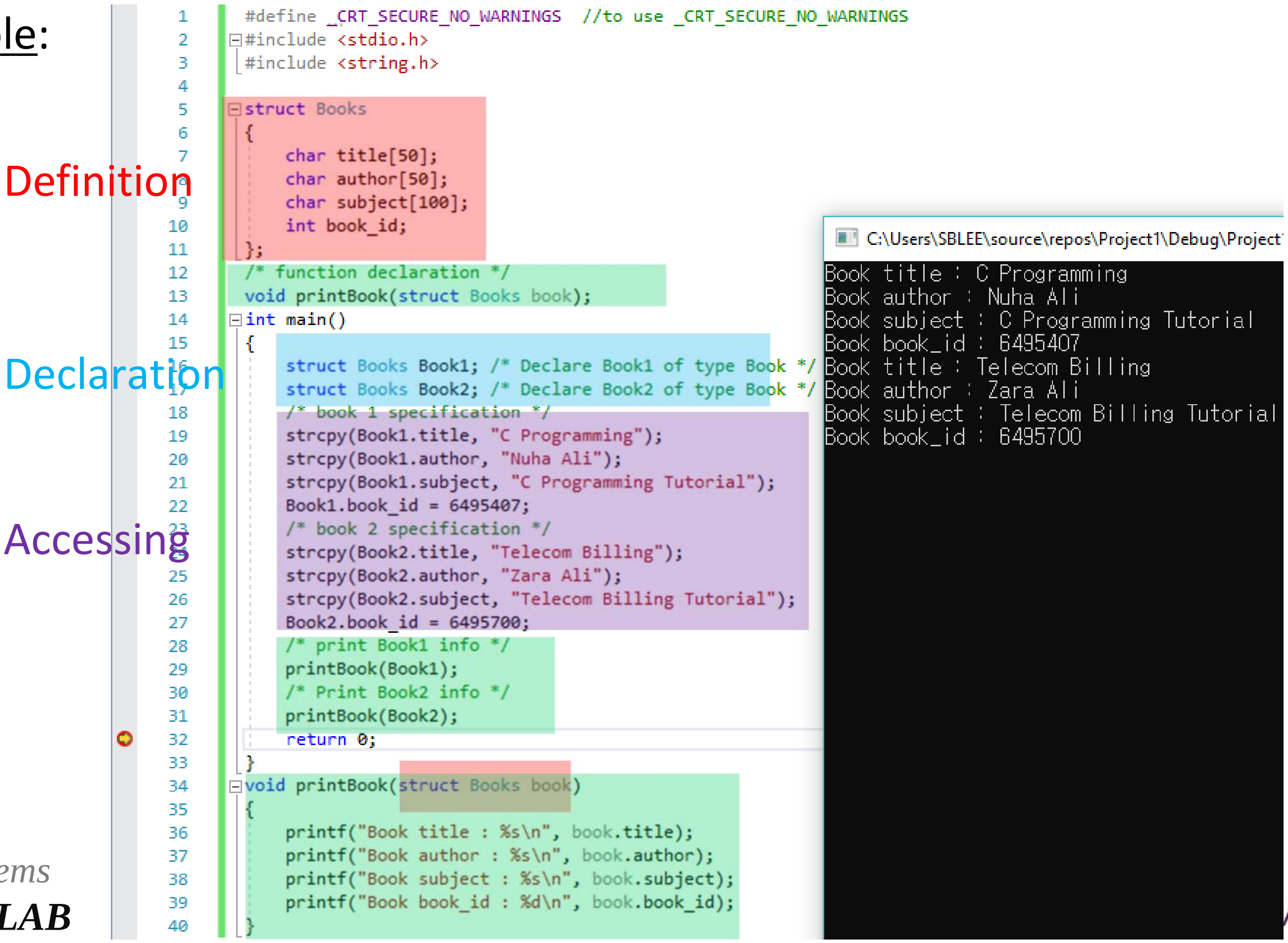
```c
#define _CRT_SECURE_NO_WARNINGS  //to use _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

struct Books
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};
int main()
{
    struct Books Book1; /* Declare Book1 of type Book */
    struct Books Book2; /* Declare Book2 of type Book */

    /* book 1 specification */
    strcpy(Book1.title, "C Programming");
    strcpy(Book1.author, "Nuha Ali");
    strcpy(Book1.subject, "C Programming Tutorial");
    Book1.book_id = 6495407;

    /* book 2 specification */
    strcpy(Book2.title, "Telecom Billing");
    strcpy(Book2.author, "Zara Ali");
    strcpy(Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;

    /* print Book1 info */
    printf("Book 1 title : %s\n", Book1.title);
    printf("Book 1 author : %s\n", Book1.author);
    printf("Book 1 subject : %s\n", Book1.subject);
    printf("Book 1 book_id : %d\n", Book1.book_id);

    /* print Book2 info */
    printf("Book 2 title : %s\n", Book2.title);
    printf("Book 2 author : %s\n", Book2.author);
    printf("Book 2 subject : %s\n", Book2.subject);
    printf("Book 2 book_id : %d\n", Book2.book_id);
    return 0;
}  ≤1ms elapsed
```

C:\Users\SBLEE\source\repos\Project1\Debug\Project1.exe

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

Structures

- Structures as Functions Arguments
  - You can pass a structure as a function argument in very similar way as you pass any other variable or pointer.
  - You would access structure variables in the similar way as you have accessed in the above example:



```c
#define _CRT_SECURE_NO_WARNINGS  //to use _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

struct Books
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};
/* function declaration */
void printBook(struct Books book);
int main()
{
    struct Books Book1; /* Declare Book1 of type Book */
    struct Books Book2; /* Declare Book2 of type Book */
    /* book 1 specification */
    strcpy(Book1.title, "C Programming");
    strcpy(Book1.author, "Nuha Ali");
    strcpy(Book1.subject, "C Programming Tutorial");
    Book1.book_id = 6495407;
    /* book 2 specification */
    strcpy(Book2.title, "Telecom Billing");
    strcpy(Book2.author, "Zara Ali");
    strcpy(Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;
    /* print Book1 info */
    printBook(Book1);
    /* Print Book2 info */
    printBook(Book2);
    return 0;
}
void printBook(struct Books book)
{
    printf("Book title : %s\n", book.title);
    printf("Book author : %s\n", book.author);
    printf("Book subject : %s\n", book.subject);
    printf("Book book_id : %d\n", book.book_id);
}
```

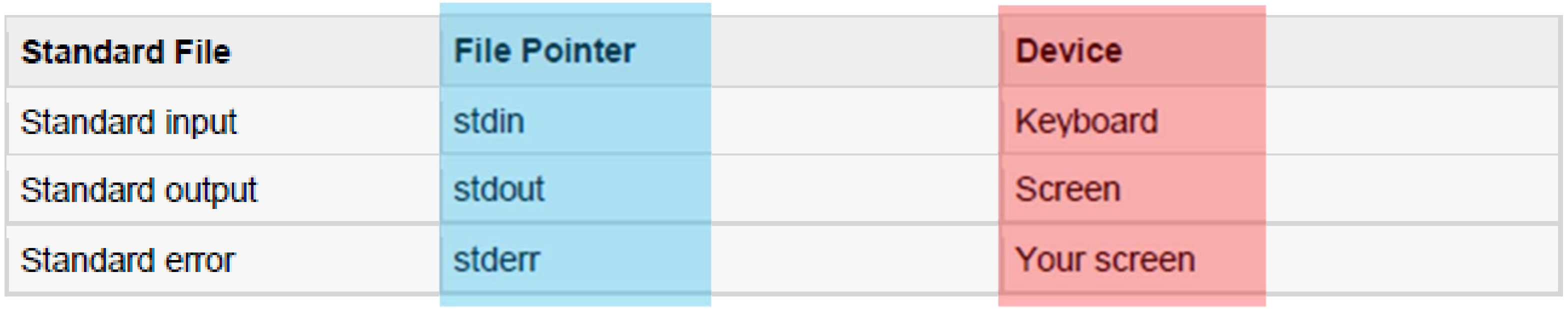
C:\Users\SBLEE\source\repos\Project1\Debug\Project

```
Book title : C Programming
Book author : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Telecom Billing
Book author : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700
```

Input & Output

Standard Files

- Standard files
  - C programming language treats all the devices as files.
  - So devices such as the display are addressed in the same way as files.
  - The following three files are automatically opened when a program executes to provide access to the keyboard and screen.
  - To do these work in C programming, we use the file pointers.



Standard File	File Pointer	Device
Standard input	stdin	Keyboard
Standard output	stdout	Screen
Standard error	stderr	Your screen

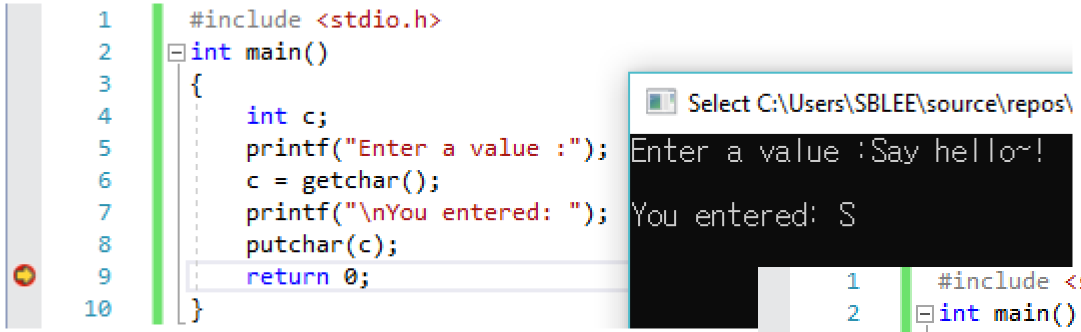
Input & Output

- Input
  - The **Input** means to **feed some data into a program**.
  - C programming language provides a set of built-in functions to read given input and feed it to the program as per requirement.

- Output
  - The **Output** means to **display some data on a screen, printer or in any file**.
  - C programming language provides a set of built-in functions to output the data on the computer screen as well as you can save that data in text or binary files.

getchar() & putchar() functions

- int getchar() functions
  - The int getchar(void) function **reads** the next available character from the screen and returns it as an integer.
  - This function reads **only single character at a time**.
- int putchar() functions
  - The int putchar(int c) function **puts** the passed character on the screen and returns the same character.
  - This function puts **only single character at a time**.

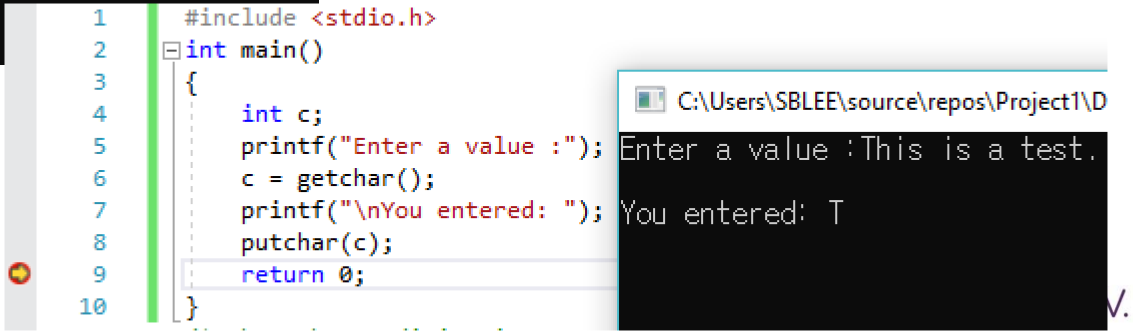


```c
#include <stdio.h>
int main()
{
	int c;
	printf("Enter a value :");
	c = getchar();
	printf("\nYou entered: ");
	putchar(c);
	return 0;
}
```

```
Select C:\Users\SBLEE\source\repos\
Enter a value :Say hello~!

You entered: S
```

Case 1

Case 2

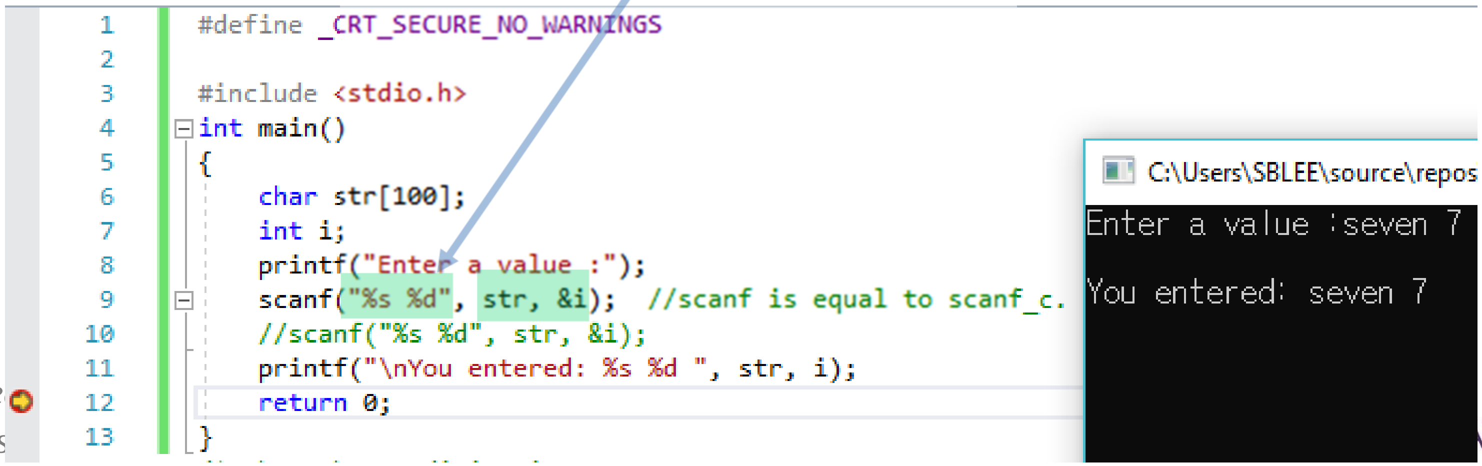


```c
#include <stdio.h>
int main()
{
	int c;
	printf("Enter a value :");
	c = getchar();
	printf("\nYou entered: ");
	putchar(c);
	return 0;
}
```

```
C:\Users\SBLEE\source\repos\Project1\D
Enter a value :This is a test.

You entered: T
```

scanf() & printf() functions

- int scanf(const char* format, …) functions
  - The int scanf(const char* format, ...) function reads input from the standard input stream **stdin** and scans that input according to format provided.
- int printf(const char* format, …) functions
  - The int printf(const char *format, ...) function writes output to the standard output stream **stdout** and produces output according to a format provided.
- format
  - The **format** can be a simple constant string, but you can specify %s, %d, %c, %f, etc., to print or read strings (%s), integer (%d), character(%c) or float(%f) respectively.



```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
int main()
{
    char str[100];
    int i;
    printf("Enter a value :");
    scanf("%s %d", str, &i);  //scanf is equal to scanf_c.
    //scanf("%s %d", str, &i);
    printf("\nYou entered: %s %d ", str, i);
    return 0;
}
```

C:\Users\SBLEE\source\repos

```
Enter a value :seven 7

You entered: seven 7
```

File input & output

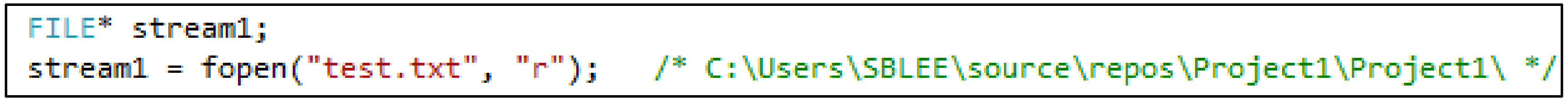
File Input & Output

- File I/O
  - This chapter we will see how C programmers can create, open, close text or binary files for their data storage.

Opening Files

- Syntax
  - The general syntax is the following:

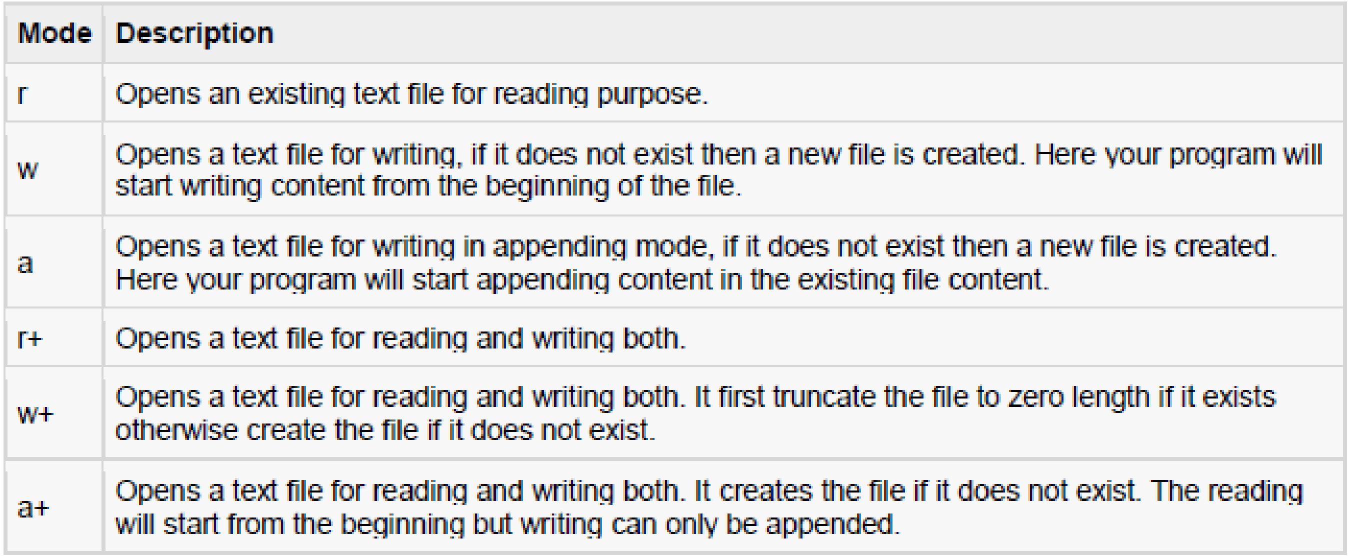
```
FILE* name;

name  = fopen(“the path of the file”, “mode”);
```



```
FILE* stream1;
stream1 = fopen("test.txt", "r");    /* C:\Users\SBLEE\source\repos\Project1\Project1\ */
```

Through this code, you can make a path to access the file (e.g. test.txt)

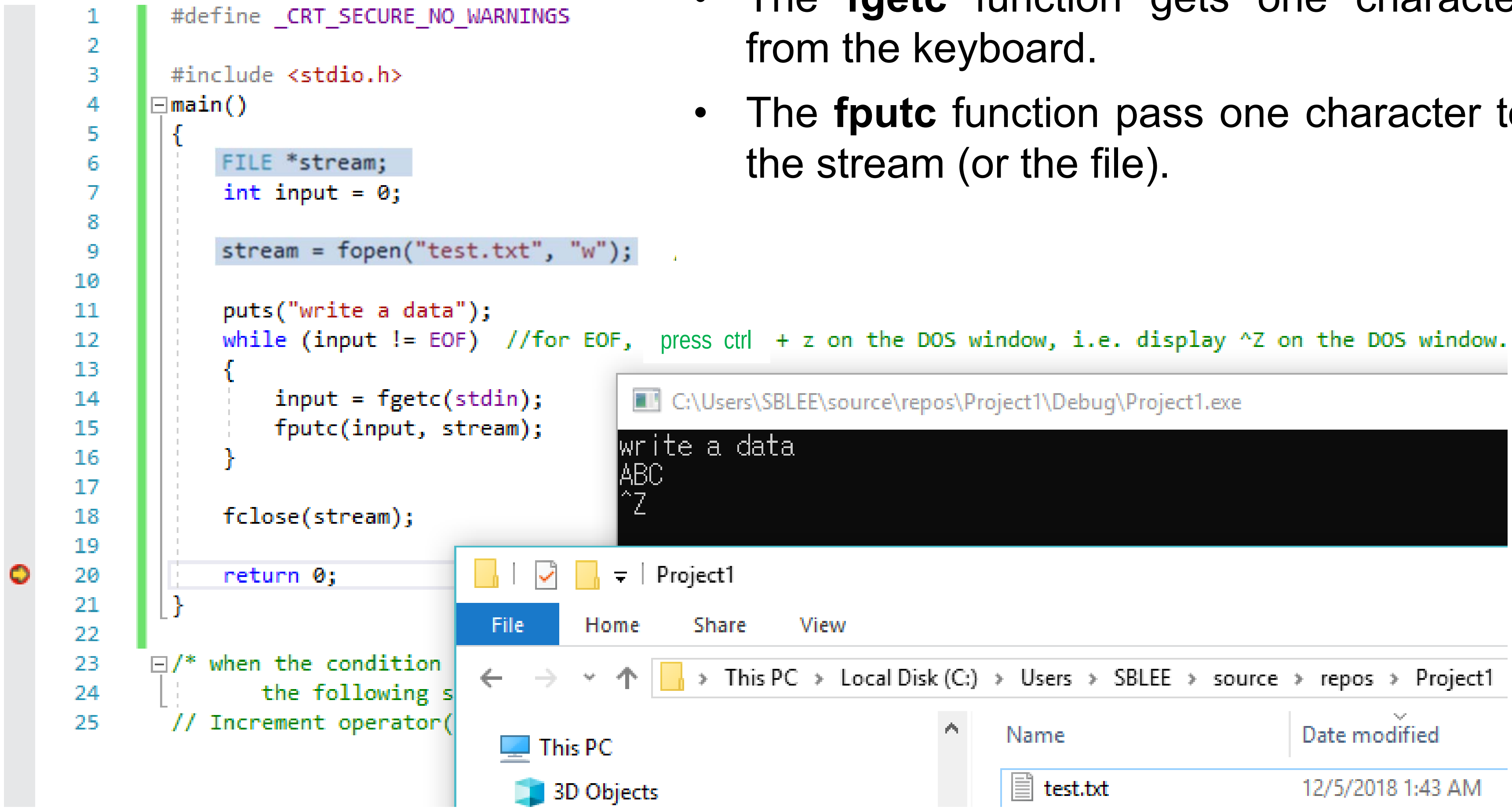
- Access mode
  - The access mode can have one of the following values:



Mode	Description
r	Opens an existing text file for reading purpose.
w	Opens a text file for writing, if it does not exist then a new file is created. Here your program will start writing content from the beginning of the file.
a	Opens a text file for writing in appending mode, if it does not exist then a new file is created. Here your program will start appending content in the existing file content.
r+	Opens a text file for reading and writing both.
w+	Opens a text file for reading and writing both. It first truncate the file to zero length if it exists otherwise create the file if it does not exist.
a+	Opens a text file for reading and writing both. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

Writing a file

- Syntax



```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
main()
{
    FILE *stream;
    int input = 0;

    stream = fopen("test.txt", "w");

    puts("write a data");
    while (input != EOF)  //for EOF,  press ctrl + z on the DOS window, i.e. display ^Z on the DOS window.
    {
        input = fgetc(stdin);
        fputc(input, stream);
    }

    fclose(stream);

    return 0;
}

/* when the condition
     the following s
// Increment operator(
```

C:\Users\SBLEE\source\repos\Project1\Debug\Project1.exe

```
write a data
ABC
^Z
```

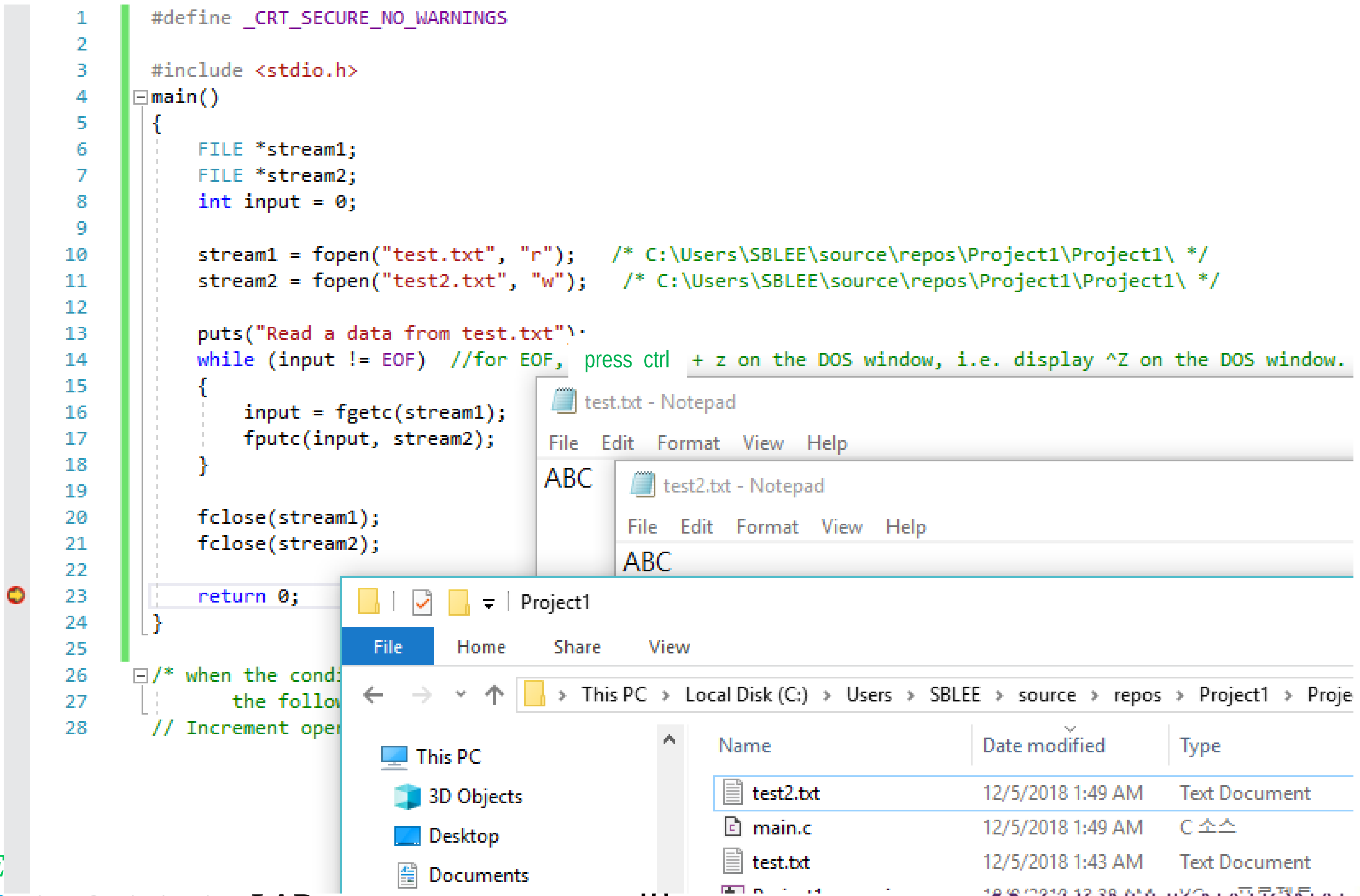
Project1
File
Home
Share
View
This PC › Local Disk (C:) › Users › SBLEE › source › repos › Project1
This PC
3D Objects
Name
Date modified

Name	Date modified
test.txt	12/5/2018 1:43 AM

- The **puts** function prints out the string on DOS window.
- EOF means the end of file.
- The **fgetc** function gets one character from the keyboard.
- The **fputc** function pass one character to the stream (or the file).

Reading a file

- Syntax



```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
main()
{
    FILE *stream1;
    FILE *stream2;
    int input = 0;

    stream1 = fopen("test.txt", "r");   /* C:\Users\SBLEE\source\repos\Project1\Project1\ */
    stream2 = fopen("test2.txt", "w");   /* C:\Users\SBLEE\source\repos\Project1\Project1\ */

    puts("Read a data from test.txt");
    while (input != EOF)  //for EOF, press ctrl + z on the DOS window, i.e. display ^Z on the DOS window.
    {
        input = fgetc(stream1);
        fputc(input, stream2);
    }

    fclose(stream1);
    fclose(stream2);

    return 0;
}

/* when the cond
        the follo
// Increment oper
```

test.txt - Notepad

```
ABC
```

test2.txt - Notepad

```
ABC
```

Project1

This PC › Local Disk (C:) › Users › SBLEE › source › repos › Project1 › Proje

Name	Date modified	Type
test2.txt	12/5/2018 1:49 AM	Text Document
main.c	12/5/2018 1:49 AM	C 소스
test.txt	12/5/2018 1:43 AM	Text Document

Summary

Summary

✓ We considered the **Structures and File Input & Output.**

Thank You