SIES: BASIC C PROGRAMMING

L #13: STRUCTURES

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 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;
```

- 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 structure variable name and the structure member that you wish to access.

Definition

Declaration

Accessing

```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:

Definition

Declaration

Accessing

```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
```

Thank You