Fundamental Coding with C

Structures

Fundamental
Coding with C

Structures

- Structures are the optimal way to represent data as a whole.
- 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 some attributes about each book. You can represent this information in a structure.

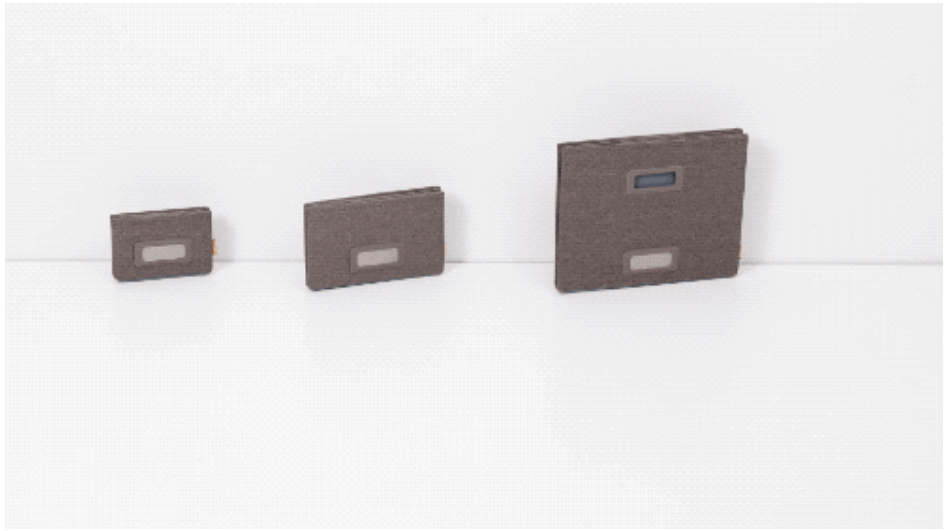
Book1
• Title • Author • Subject • Book ID

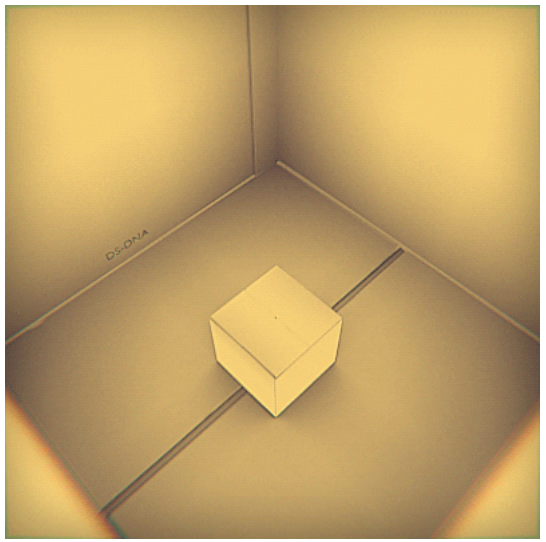
Book2
• Title • Author • Subject • Book ID

Fundamental
Coding with C

Structures

- Individual components of a struct type are called members (or fields).
- Members can be of different types (simple, array or struct).
- A struct is named as a whole while individual members are named using field identifiers.
- Complex data structures can be formed by defining arrays of structs.





Fundamental
Coding with C

Structures

- To define a structure in C, you must use the **struct** statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows:

```
struct [structure tag] {

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

- The structure tag is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition.
- At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.

Fundamental
Coding with C

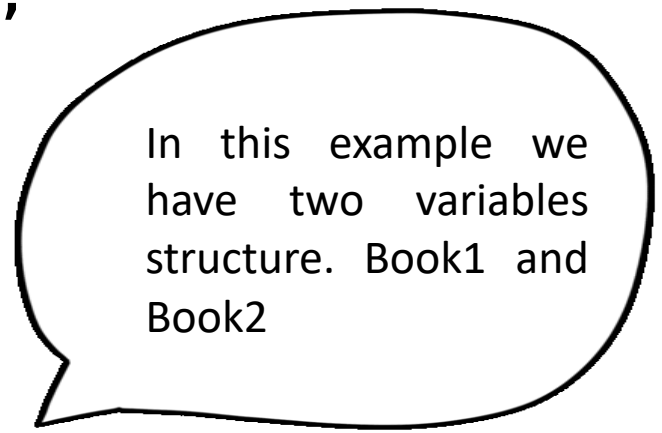
Structures

- Structure in C definition example:

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

- Structure in C declaration example:

```
struct Books Book1;
struct Books Book2;
```



Fundamental
Coding with C

Structures

- Structure in C. Accessing Structure Members :
- To access any member of a structure, we use **the member access operator (.)**.
- The member access operator is coded as a period between the structure variable name and the structure member that we wish to access.

```
strcpy( Book1.title, "C Programming");
strcpy( Book1.author, "Nuha Ali");
strcpy( Book1.subject, "C Programming Tutorial");
Book1.book_id = 6495407;
```

The member access operator (.)

Remember, to assign a string to a variable you need to use `strcpy`

Fundamental
Coding with C

Is time to Try

https://repl.it/languages/c

Fundamental
Coding with C

Structures

```c
#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, "Juan Perez");
       strcpy( Book1.subject, "C Programming Tutorial");
       Book1.book_id = 6495407;

       /* book 2 specification */
       strcpy( Book2.title, "Java");
       strcpy( Book2.author, "Jhon smith");
       strcpy( Book2.subject, "Java 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;
}
```