BAKI ALİ NEFT MƏKTƏBİ
BAKU HIGHER OIL SCHOOL

Programming and Computer Applications-1

Structures

Instructor : PhD, Associate Professor Leyla Muradkhanli

Introduction

- **Structures**—are collections of related variables under one name.
- Structures may contain variables of many different data types—in contrast to arrays that contain only elements of the same data type.
- structure is another user defined data type available in C that allows to combine data items of different kinds.

Structure Definitions

- Structures are derived data types—they are constructed using objects of other types.
- 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

Structure Definitions

To define a structure, 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];
```

Structure Definitions

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. Here is the way you would declare the Book structure

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

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. You would use the keyword struct to define variables of structure type.
- The following example shows how to use a structure in a program

Accessing Structure Members

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};
int main( ) {
   struct Books Book1;
   strcpy( Book1.title, "C Programming Language");
   strcpy( Book1.author, "Brain Kernighan");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 1234567;
   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);
   getch();
   return 0;
}
```

Structures as Function Arguments

You can pass a structure as a function argument in the same way as you pass any other variable or pointer.

```
void printBook( struct Books book );
```

Structures as Function Arguments

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>

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

void printBook( struct Books book );
int main( ) {

   struct Books Book1;

   strcpy( Book1.title, "C Programming Language");
   strcpy( Book1.author, "Brain Kernighan");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 1234567;
```

Structures as Function Arguments

```
   printBook( Book1 );
   getch();
   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);
}
```

Pointers to Structures

You can pass a structure as a function argument in the same way as you pass any other variable or pointer.

```
struct Books *struct_pointer;
```

Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the '&'; operator before the structure's name as follows

```
struct_pointer = &Book1;
```

To access the members of a structure using a pointer to that structure, you must use the → operator as follows

```
struct_pointer->title;
```

Pointers to Structures

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};
void printBook( struct Books *book );
int main( ) {
   struct Books Book1;
   strcpy( Book1.title, "C Programming Language");
   strcpy( Book1.author, "Brain Kernighan");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 1234567;
```

Pointers to Structures

```
printBook( &Book1 );
   getch();
   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);
}
```

Structure

- For example, the following `struct` contains character array members for an employee's first and last names, an `int` member for the employee's age, a `char` member that would contain `'M'` or `'F'` for the employee's gender and a `double` member for the employee's salary:,
    - ```
      struct employee {
          char firstName[ 20 ];
          char lastName[ 20 ];
          int age;
          char gender;
          double Salary;
      };
      ```
      ```