

Lab No. 05

Nested Structure and Pointer to Structure

OBJECTIVE:

Things that will be covered in today's lab:

- Nested Structures (A struct within a struct)
- Pointer to Structure

THEORY:

A nested structure in C++ is nothing but a structure within a structure. One structure can be declared inside the other structure as we declare structure members inside a structure. A structure variable can be a normal structure variable or a pointer variable to access the data.

Declare two separate struct **Declare a nested struct:**

```
struct xyz{  
    // xyz member variables  
};  
struct abc  
{  
    xyz var;  
    // abc member variables  
}a;  
  
// Access member variables  
a.abc_member  
a.var.xyz_member
```

```
struct abc{  
    // abc member variables  
    struct xyz  
    {  
        // xyz member variables  
    }var;  
}a;  
  
// Access member variables  
a.abc_member  
a.var.xyz_member
```

Access nested members:

- You can access the members of “*abc*” struct using dot operator.
- A “*xyz*” structure is nested within the “*abc*” struct so the members of “*xyz*” can be accessed using “*abc*” struct object.

Pointer to structure

You can define pointers to structures in a very similar way as you define a pointer to any other variable. 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 “&” operator before the name of the structure.

Here variable is an object of structure type *struct_name* and *p_var* is a pointer to point to objects of structure type variable. Therefore, the following code would also be valid:

```
struct_name * p_var;
struct_name variable;
p_var = & variable;
p_var->struct_members;
```

The value of the pointer *p_var* would be assigned the address of the object variable. The arrow operator (*->*) is a **dereference** operator that is used exclusively with pointers to objects that have members. This operator serves to access the member of an object directly from its address.

Expression	What is evaluated	Equivalent
a.b	Member b of object a	
a->b	Member b of object pointed to by a	(*a).b

Example: What should be the output of this program?

```
struct movies_t
{
    char * title ;
    int year;
};
int main ()
{
    movies_t amovie;

    movies_t * pmovie;
    pmovie = & amovie;

    pmovie->title= "MATRIX";
    pmovie->year = 1999;
    cout << pmovie->title;
+   cout <<" ("<< pmovie->year <<") \n";
}
```

Exercise 1:

Write a program to help a university system to store records for its employee. You have to perform the following tasks:

1. Define a struct **facultyMember** with the following attributes:
 - ID number (int)
 - First Name (string)
 - Last Name (string)
 - Designation (string) e.g. Assistant professor, Lecturer etc.
2. Implement a function **“void newrecord(facultyMember & fm)”**, which will take an argument of *FacultyMember* type, input values for all the attributes from user, and store it in the argument variable.
3. Implement a function **“void printdetails(facultyMember fm)”**, which will print the values of the variable fm passed as an argument.
4. Implement your main function. Declare a variable of *FacultyMember* type. Assign values to it using *NewRecord* function. Print its values using *PrintDetails* function.
5. Now, declare a *FacultyMember* type array of size 3 in *main()*. Fill the values using *newRecord* function. (Note that your *newRecord* can assign value to a single *FacultyMember* type variable and you cannot change the prototype).
6. Print the values of the above array using *PrintDetails* Function without changing the prototype.
7. Implement a function **“void sortid(facultyMember fm [], int size)”**, which takes a *FacultyMember* type array as an argument and its maximum size. You have to sort this array in ascending order with respect to the ID number (use any sorting algorithms). Be careful while swapping the two locations of array.

```
// Point 5
facultyMember f1[3];
for( int i=0; i<3; i++)
    newrecord(f1[i]);
for( int j=0; j<3; j++)
    printdetails(f1[j]);

// Point 7
sortid(f1,3);
for( int k=0; k<3; k++)
    printdetails(f1[k]);
```

Exercise 2:

Implement another struct **University** with the following details:

- Name of University (string)
- Address (string)
- List of faculty Members (*FacultyMember* type array of size 3)

In `main ()`, declare a variable *myUni* of the *University* type. Set any values to name and address of *myUni*. Copy the values of the array you declared in step5 into the *FacultyMember* type array of *myUni*.

Print the details of the *myUni*. You should use the *PrintDetails* function for printing the details of Faculty member list of *myUni*.

```

        university uni;
        uni.name="National University";
        uni.address="Block-B,Faisal Town";
        for( int b=0; b<3; b++) {
// Assign value university faculty member
        }
        for( int a=0; a<3; a++) {
            cout<<uni.name<<"    "<<uni.address<<"    ";
            printdetails(uni.FM[a]);
        }
    
```

Post Lab:

Write a C++ program to read data and display the author's record. The structure definition of authors should include the following details:

- Author's ID
- Author's Name
- Book List
 - Book Code
 - Book name
 - Subject
 - Book Price
 - Edition

The record of each author should contain details of three books.