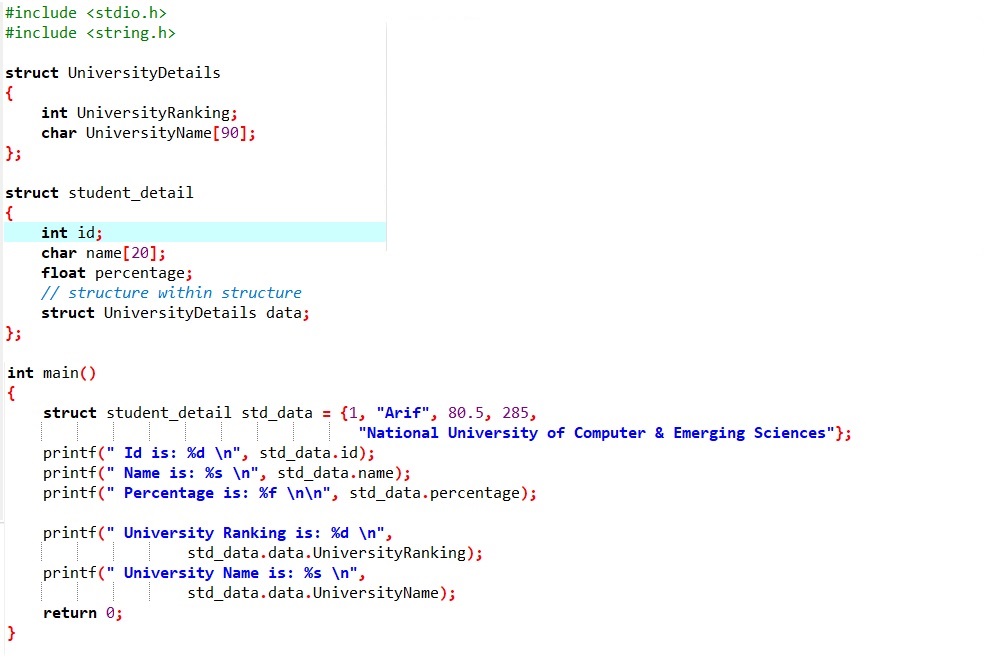
|  |  |  |
| --- | --- | --- |
| **CL1002 Programming Fundamentals** | **LAB 11**  **Nested Structure,**  **Introduction to Pointers,**  **Accessing Arrays using Pointer,**  **Dynamic Memory Management** | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |  |

**Learning Objectives**

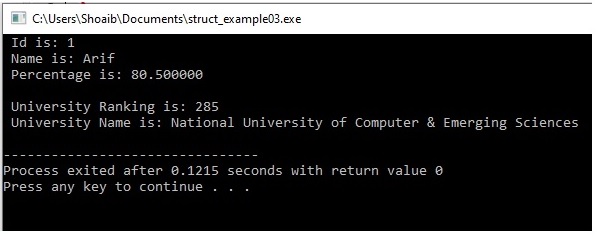
1. Nested Structures
2. Introduction to Pointers
3. Accessing Array using Pointers
4. Dynamic Memory

**2.0 Nested Structures**

Nested structure in C is nothing but structure within structure. One structure can be declared inside other structure as we declare structure members inside a structure. The structure variables can be a normal structure variable ,array or a pointer variable to access the data. You can learn below concepts in this section.

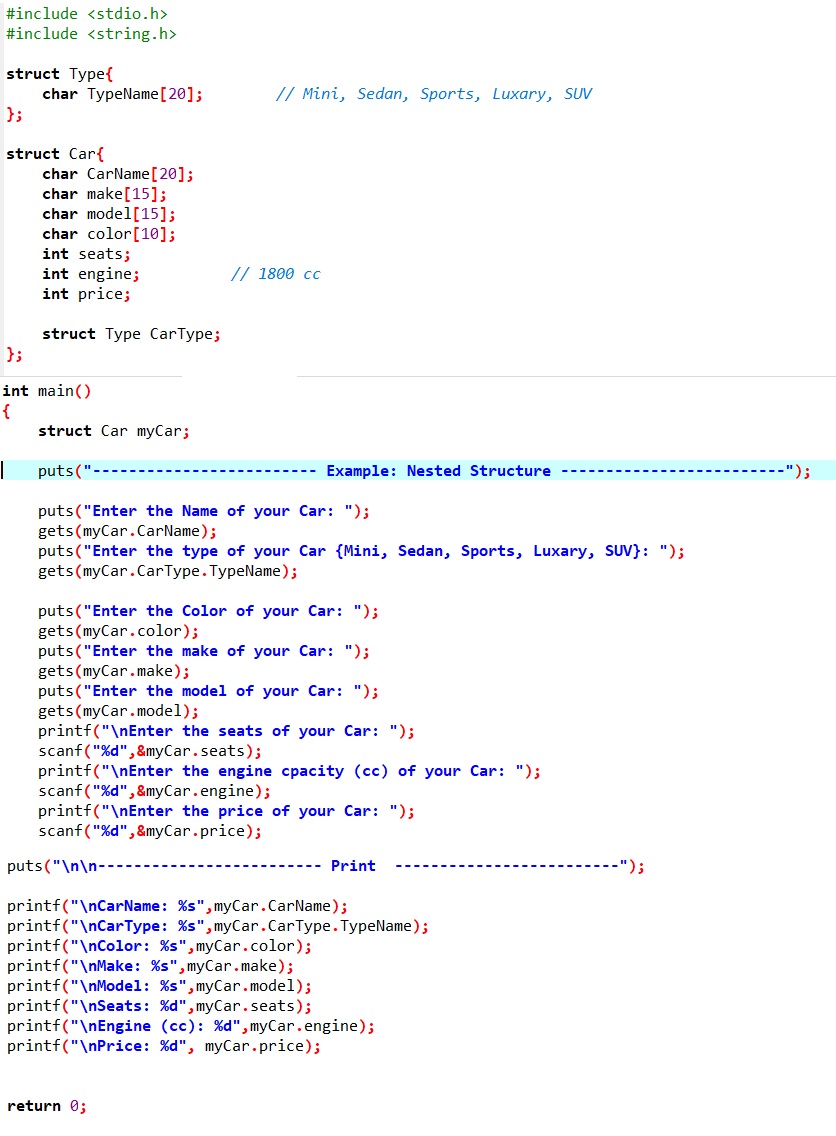


**OUTPUT:**

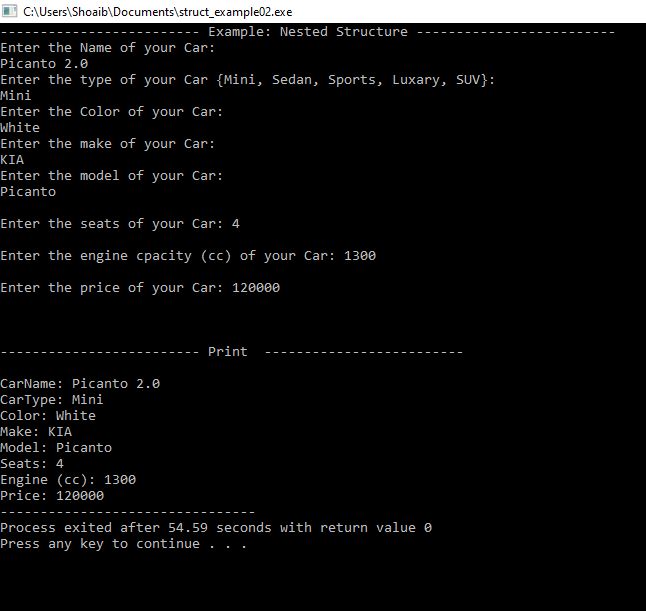
****

Another example of Nested Structure:

**Sample Code:**



**OUTPUT:**

****

**3.0 Introduction to Pointers**

Pointer is a variable whose value is a memory address. Normally, a variable directly contains a specific value. A pointer contains the memory address of a variable that, in turn, contains a specific value. In this sense, a variable name directly references a value, and a pointer indirectly references a value.

**3.1 Pointer Declaration & Initialization**

**Syntax:** type \* variable;

**Code:**  int \*ptr = 0; // Pointer Declaration

int var= 10;

ptr = &var; // Pointer Initialization

The value of the pointer variable ptr is a memory address. A data item whose address is stored in this variable must be of the specified type.

**int \*ptr = 0; or int \*ptr = NULL;**

**int var = 10;**

**var**

**ptr**

var = 10

&var = 6024

ptr = 0

&ptr = 6000

**NULL**

**10**

**6000**

**6024**

**ptr = &var;**

**var**

**ptr**

var = 10

&var = 6024

ptr = 6024

&ptr = 6000

**10**

**\*ptr**

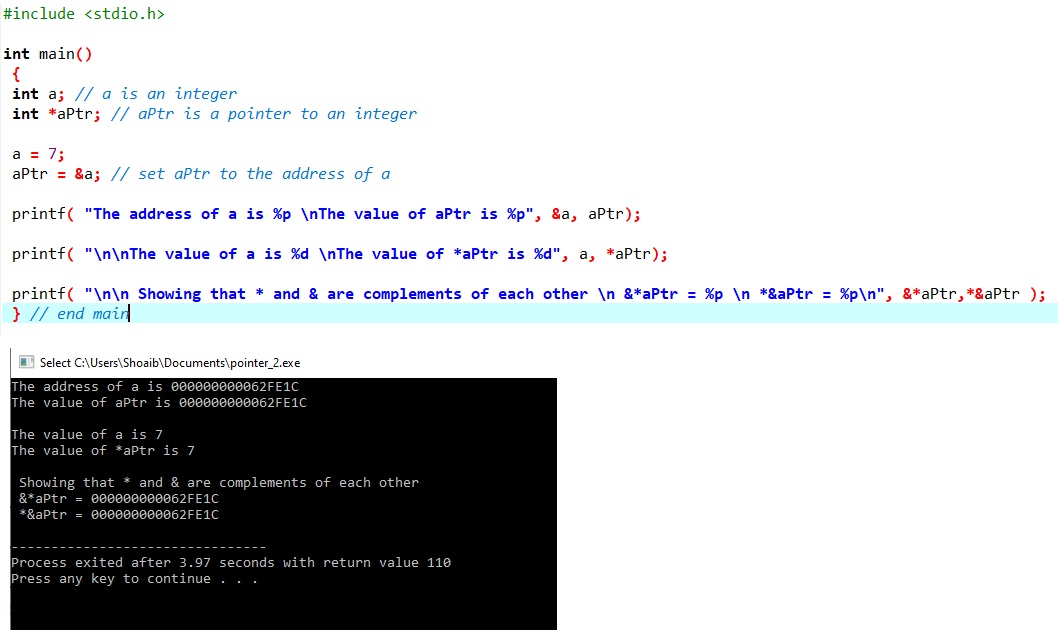
**6024**

\*ptr=10

**6000**

**6024**

**Sample Code:**



**3.2 POINTER ARITHMETICS**

* A limited set of arithmetic operations may be performed on pointers. A pointer may be incremented (++) or decremented (--), an integer may be added to a pointer (+ or +=), an integer may be subtracted from a pointer (- or -=) and one pointer may be subtracted from another.
* When an integer is added to or subtracted from a pointer, the pointer is incremented or decremented by that integer times the size of the object to which the pointer refers.
* Two pointers to elements of the same array may be subtracted from one another to determine the number of elements between them.

**Pointers with Functions**

* **Pass by reference:**

Normally when passing a variable to a function, the compiler makes a COPY of the variable in the function. Hence changing the value of the argument in the function does not change the original value. This is called pass by value. Sometimes, like in scanf(), we want to change the variable inside the function. To do this, we pass a pointer as input argument to the function this is called pass by reference.

**Example:**

#include <stdio.h>

void swap(int \*n1, int \*n2);

int main()

{

int num1 = 5, num2 = 10;

printf("Before swape \n");

printf("num1 = %d\n", num1);

printf("num2 = %d", num2);

// address of num1 and num2 is passed

swap( &num1, &num2);

printf("After swape \n");

printf("num1 = %d\n", num1);

printf("num2 = %d", num2);

return 0;

}

void swap(int\* n1, int\* n2)

{

int temp;

temp = \*n1;

\*n1 = \*n2;

\*n2 = temp;

}