**Structures in C**

**C Structures :**Structure is a user-defined datatype in [C language](https://www.studytonight.com/c/overview-of-c.php) which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful. It is somewhat similar to an [Array](https://www.studytonight.com/c/arrays-in-c.php), but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical more useful. In structure, data is stored in form of **records**.

**For example:** If I have to write a program to store Student information, which will have Student's name, age, branch, permanent address, father's name etc, which included string values, integer values etc, how can I use arrays for this problem, I will require something which can hold data of different types together.

Syntax to Define a Structure in C

struct structName

{

   // structure definition

   Data\_type1 member\_name1;

   Data\_type2 member\_name2;

   Data\_type2 member\_name2;

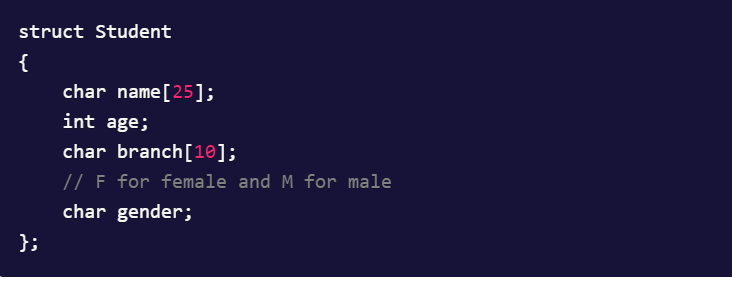
};

As you can see in the syntax above, we start with the struct keyword, then it's optional to provide your structure a name, we suggest you to give it a name, then inside the curly braces, we have to mention all the member variables, which are nothing but normal C language variables of different types like int, float, array etc.

After the closing curly brace, we can specify one or more structure variables, again this is optional.

**Note:** The closing curly brace in the structure type declaration must be followed by a semicolon(;).

Example:



Here **struct Student** declares a structure to hold the details of a student which consists of 4 data fields, namely name, age, branch and gender. These fields are called **structure elements or members**.

Each member can have different data type, like in this case, name is an array of char type and age is of int type etc. **Student** is the name of the structure and is called as the **structure tag**.

### **Declaring Structure Variables :**It is possible to declare variables of a **structure**, either along with structure definition or after the structure is defined. **Structure** [variable](https://www.studytonight.com/c/variables-in-c.php) declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

#### **Declaring Structure variables separately**

struct Student

{

char name[25];

int age;

char branch[10];

//F for female and M for male

char gender;

};

struct Student S1, S2; //declaring variables of struct Student

#### **2) Declaring Structure variables with structure definition**

struct Student

{

char name[25];

int age;

char branch[10];

//F for female and M for male

char gender;

}S1, S2;

### **Accessing Structure Members:**Structure members can be accessed and assigned values in a number of ways. Structure members have no meaning individually without the structure. In order to assign a value to any structure member, the member name must be linked with the **structure** variable using a dot . operator also called **period** or **member access** operator.

**For example:**

**#include<stdio.h>**

**#include<string.h>**

**struct Student**

**{**

**char name[25];**

**int age;**

**char branch[10];**

**//F for female and M for male**

**char gender;**

**};**

**void main()**

**{**

**struct Student s1; // s1 is a variable of Student type and age is a member of Student**

**s1.age = 18; // using string function to add name**

**strcpy(s1.name, "Viraaj"); // displaying the stored values**

**printf("Name of Student 1: %s\n", s1.name);**

**printf("Age of Student 1: %d\n", s1.age);**

**}**

**Output:**

Name of Student 1: Viraaj

Age of Student 1: 18

We can also use scanf() to give values to structure members through terminal.

scanf(" %s ", s1.name);

scanf(" %d ", &s1.age);

### **Structure Initialization :** Like a variable of any other datatype, structure variable can also be initialized at compile time.

struct Patient

{

float height;

int weight;

int age;

};

struct Patient p1 = { 180.75 , 73, 23 }; //initialization

OR

**struct Patient p1;**

**p1.height = 180.75; //initialization of each member separately**

**p1.weight = 73;**

**p1.age = 23;**

### **Array of Structure:** We can also declare an array of **structure** variables. in which each element of the [array](https://www.studytonight.com/c/arrays-in-c.php) will represent a **structure** variable. **Example :** struct employee emp[5];

The below program defines an array emp of size 5. Each element of the array emp is of type Employee.

**struct Employee**

**{**

**char ename[10];**

**int sal;**

**};**

**struct Employee emp[5];**

**int i, j;**

**void ask()**

**{**

**for(i = 0; i < 3; i++)**

**{**

**printf("\nEnter %dst Employee record:\n", i+1);**

**printf("\nEmployee name:\t");**

**scanf("%s", emp[i].ename);**

**printf("\nEnter Salary:\t");**

**scanf("%d", &emp[i].sal);**

**}**

**printf("\nDisplaying Employee record:\n");**

**for(i = 0; i < 3; i++)**

**{**

**printf("\nEmployee name is %s", emp[i].ename);**

**printf("\nSlary is %d", emp[i].sal);**

**}**

**}**

**void main()**

**{**

**ask();**

**}**

### **Structure as Function Arguments :**We can pass a structure as a function argument just like we pass any other variable or an array as a function argument.

**struct Student**

**{**

**char name[10];**

**int roll;**

**};**

**void main()**

**{**

**void show(struct Student st);**

**struct Student std;**

**printf("\nEnter Student record:\n");**

**printf("\nStudent name:\t");**

**scanf("%s", std.name);**

**printf("\nEnter Student rollno.:\t");**

**scanf("%d", &std.roll);**

**show(std);**

**}**

**void show(struct Student st)**

**{**

**printf("\nstudent name is %s", st.name);**

**printf("\nroll is %d", st.roll);**

**}**

# **typedef in C :** typedef is a [keyword](https://www.studytonight.com/c/keywords-and-identifier.php) used in C language to assign alternative names to existing [datatypes](https://www.studytonight.com/c/datatype-in-c.php" \t "_blank). Its mostly used with user defined datatypes, when names of the datatypes become slightly complicated to use in programs. Following is the general syntax for using typedef,

Lets take an example and see how typedef actually works.

**typedef unsigned long ulong;**

The above statement define a term ulong for an unsigned long datatype. Now this ulong identifier can be used to define unsigned long type variables.

**ulong i, j;**

### **Application of typedef :**typedef can be used to give a name to user defined data type as well. Lets see its use with [structures](https://www.studytonight.com/c/structures-in-c.php).

**typedef struct**

**{**

**type member1;**

**type member2;**

**} type\_name;**

Here **type\_name** represents the stucture definition associated with it. Now this **type\_name** can be used to declare a variable of this stucture type.

**type\_name t1, t2;**

### **Structure definition using typedef:** Let's take a simple code example to understand how we can define a structure in C using typedef keyword.

**typedef struct employee**

**{**

**char name[50];**

**int salary;**

**}emp;**

**void main( ) {**

**{**

**emp e1;**

**}**