**Structures Data Types**

A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

**How to define structures?**

Before you can create structure variables, you need to define its data type. To define a struct, the ***struct*** keyword is used.

**Syntax of struct**

*struct* ***structureName***

*{*

*dataType member1;*

*dataType member2;*

*...*

*};*

**Here is an example:**

*struct Person*

*{*

*char name;*

*int cityNo=50;*

*float salary;*

*};*

*main()*

*{*

*typedef struct Person* ***per****;*

*per p;*

*cout<<enter the value of structure elements”<<endl;*

*cin>>p.name>>p.cityno>>p.salary;*

*cout<<”these are the values of structure”;*

*cout<<p.name<<”\n”<<p.cityno<<”\n”<<”salary=”<<p.salary;*

*getch();*

*}*

*Used to store the records*

**Create struct variables**

When a struct type is declared, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Here's how we create structure variables:

*struct Person*

*{*

*char name[50];*

*int citNo;*

*float salary;*

*} ;*

***typedef struct Person per;***

*int main()*

*{*

*per a,a1={“vijay”,750, 20000.45 };*

*scanf(“%s%d%f”, a.name,&a.citNo,&a.salary);*

*printf(“%d”, a1. salary);*

*return 0;*

*}*

**Another way of creating a struct variable is:**

*struct Person*

*{*

*char name[50];*

*int citNo;*

*float salary;*

*} person1, person2, p[20];*

In both cases, two variables **person1, person2**, and an array variable**p** having 20 elements of type struct Person are created.

**Keyword typedef**

We use the typedef keyword to create an alias name for data types. It is commonly used with structures to simplify the syntax of declaring variables.

**This code**

*struct Distance{*

*int feet;*

*float inch;*

*};*

*int main() {*

*struct Distance d1, d2;*

*}*

**Is equivalent to**

*typedef struct Distance{*

*int feet;*

*float inch;*

*} distances;*

*int main() {*

*distances d1, d2;*

*}*

Or

*Typedef struct Distance dis;*

*Dis d1,d2;*

**How to initialize structure members?**

Structure members **cannot be** initialized with declaration. For example the following C program fails in compilation.

|  |
| --- |
| *struct Point*  *{*  *int x = 0;  // COMPILER ERROR:  cannot initialize members here*  *int y = 0;  // COMPILER ERROR:  cannot initialize members here*  *};* |

The reason for above error is simple, when a datatype is declared, no memory is allocated for it. Memory is allocated only when variables are created.

Structure members **can be** initialized using curly braces ‘{}’. For example, following is a valid initialization.

|  |
| --- |
| struct Point  {     int x, y;  };    int main()  {     // A valid initialization. member x gets value 0 and y     // gets value 1.  The order of declaration is followed.     struct Point p1 = {0, 1};  //sequence should be same  } |

**Access members of a structure**

There are two types of operators used for accessing members of a structure.

1. .(Member operator)/ member access operator
2. -> (Structure pointer operator)/pointer to member access operator

Suppose, you want to access the salary of person2. Here's how you can do it.

***person2.salary***

**Structure example:**

*#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, "Nuha Ali");*

*strcpy( Book1.subject, "C Programming Tutorial");*

*Book1.book\_id = 6495407;*

*/\* book 2 specification \*/*

*strcpy( Book2.title, "Telecom Billing");*

*strcpy( Book2.author, "Zara Ali");*

*strcpy( Book2.subject, "Telecom Billing 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;*

*}*

When the above code is compiled and executed, it produces the following result −

*Book 1 title : C Programming*

*Book 1 author : Nuha Ali*

*Book 1 subject : C Programming Tutorial*

*Book 1 book\_id : 6495407*

*Book 2 title : Telecom Billing*

*Book 2 author : Zara Ali*

*Book 2 subject : Telecom Billing Tutorial*

*Book 2 book\_id : 6495700*

**What is an array of structures?**

Like other primitive data types, we can create an array of structures.

|  |
| --- |
| #include<stdio.h>    struct Point  {     int x, y;  };    int main()  {     // Create an array of structures     struct Point arr[10];       // Access array members     arr[0].x = 10;     arr[0].y = 20;       printf("%d %d", arr[0].x, arr[0].y);     return 0;  } |

**Output:**

10 20

***What is a structure pointer?***

Like primitive types, we can have pointer to a structure. If we have a pointer to structure, members are accessed using arrow ( -> ) operator.

|  |
| --- |
| #include<stdio.h>  struct Point  {     int x, y;  };    int main()  {     struct Point p1 = {1, 2};       // p2 is a pointer to structure p1     struct Point \*p2 = &p1;       // Accessing structure members using structure pointer     printf("%d %d", p2->x, p2->y);     return 0;  } |

**Output:**

1 2

[**union**](https://www.geeksforgeeks.org/union-c/)

A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes. 

**Defining a Union:** To define a union, you must use the **union** statement in the same way as you did while defining a structure. The union statement defines a new data type with more than one member for your program. The format of the union statement is as follows:

union [union name]

{

member definition;

member definition;

...

member definition;

};

**Similarities between Structure and Union**

1. Both are user-defined data types used to store data of different types as a single unit.
2. Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
3. Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.
4. A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
5. **‘.’** operator is used for accessing members.

**Differences:**

