**Science College for Women/ Second Class/System programming/Lec2**

**Lecture: Asraa A.H.**

**DERIVED TYPES::**

Array and struct are derived types in C. They are more complex than primary. An array is defined as **finite ordered collection of homogenous** data, stored in contiguous memory locations.

* **finite** *means* data range must be defined.
* **ordered** *means* data must be stored in continuous memory addresses.
* **homogenous** *means* data must be of similar data type.

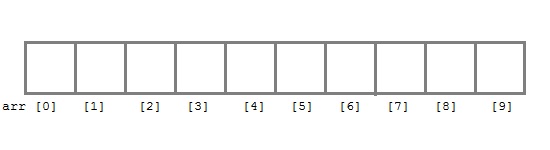
### Declaring an Array

Like any other variable, arrays must be declared before they are used. General form of array declaration is,

data-type array-name[size];

/\* Example of array declaration \*/

int arr[10];



Here int is the data type, arr is the name of the array and 10 is the size of array. It means array arr can only contain 10 elements of int type.

**Index** of an array starts from 0 to **size-1** i.e first element of arr array will be stored at arr[0]address and the last element will occupy arr[9].

Example 1::

#include<stdio.h>

void main()

{ int i;

int arr[5]={2,3,4};// Compile time array initialization

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

printf("%d",arr[i]);

getch();

}

#include<stdio.h>

void main()

{ int arr[4];

int i, j;

printf("Enter array element");

for(i =0; i <4; i++)

scanf("%d",&arr[i]);//Run time array initialization

for(j =0; j <4; j++)

printf("%d\n", arr[j]);

getch();

}

Example 2:: Find summation for array elements::

#include<stdio.h>

void main()

{ int arr[4];

int i, j;

int s=0;

printf("Enter array element");

for(i = 0; i < 4; i++)

scanf("%d", &arr[i]);

for(j = 0; j < 4; j++)

printf("%d\n", arr[j]);

for(j = 0; j < 4; j++)

s=s+arr[j];

printf("%d\n", s);

getch();

}

Output

Enter array element2 3 4 5

2

3

4

5

14

Example 3:: add value 10 to every value for array elements:

#include<stdio.h>

void main()

{ int arr[4];

int i, j;

int s=0;

printf("Enter array element");

for(i = 0; i < 4; i++)

scanf("%d", &arr[i]);

for(j = 0; j < 4; j++)

printf("%d\n", arr[j]);

for(j = 0; j < 4; j++)

arr[j]=arr[j]+10;

for(j = 0; j < 4; j++)

printf("%d\n", arr[j]);

getch();

}

Output

Enter array element2 3 4 5

2

3

4

5

12

13

14

15

### Struct

Structure is allow us to combine data of different types together in C language. It is somewhat similar to an Array, but an array holds data of similar type only. Structure on the other hand, can store data of any type which is practical more useful.

**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. In structure, data is stored in form of **records**.

### Defining a structure

struct keyword is used to define a structure. struct defines a new data type which is a collection of primary and derived data types.

**Syntax:**

struct [structure\_name]

{

//member variable 1

//member variable 2

//member variable 3

...

}[structure\_variables];

As you can see in the syntax above, we start with the struct keyword, then provide your structure a name. 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.

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

#### Example of Structure

struct Student

{

char name[25];

int age;

char branch[10];

// F for female and M for male

char gender;

};

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 name**.

### 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 declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

#### 1) 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 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. We can also declare an array of **structure** variables in which each element of the array will represent a **structure** variable.

**For example:**

#include <stdio.h>

struct Distance

{ int feet;

float inch;

} dist1, dist2, sum;

int main()

{ printf("1st distance\n");

// Input of feet for structure variable dist1

printf("Enter feet: ");

scanf("%d", &dist1.feet);

// Input of inch for structure variable dist1

printf("Enter inch: ");

scanf("%f", &dist1.inch);

printf("2nd distance\n");

// Input of feet for structure variable dist2

printf("Enter feet: ");

scanf("%d", &dist2.feet);

// Input of feet for structure variable dist2

printf("Enter inch: ");

scanf("%f", &dist2.inch);

sum.feet = dist1.feet + dist2.feet;

sum.inch = dist1.inch + dist2.inch;

// printing sum of distance dist1 and dist2

printf("Sum of distances = %d", sum.feet);

printf("Sum of distances = %f", sum.inch);

getch();

return 0;

}