

Pointer, Structure, Union

Subject: Programming In C Language

Data Types

C programming language which has the ability to divide the data into different types. The type of a variable determine the what kind of values it may take on. The various data types are

- Simple Data type
 - Integer, Real, Void, Char
- Structured Data type
 - Array, Strings
- User Defined Data type
 - Enum, Structures, Unions

Structure Data Type

- A structure is a user defined data type that groups logically related data items of different data types into a single unit. All the elements of a structure are stored at contiguous memory locations.
- A variable of structure type can store multiple data items of different data types under the one name.
- As the data of employee in company that is name, Employee ID, salary, address, phone number is stored in structure data type.

Defining of Structure

A structure has to be defined, before it can be used. The syntax of defining a structure is

```
struct <struct_name>
{
<data_type> <variable_name>;
<data_type> <variable_name>;
.....
<data_type> <variable_name>;
};
```

Example of Structure

The structure of Employee is declared as

```
struct employee
{
int emp_id;
char name[20];
float salary;
char address[50];
int dept_no;
int age;
};
```

Memory Space Allocation

8000

emp_id

8002

name[20]

8022

salary

8024

address[50]

8074

dept_no

8076

age

8078

employee

Declaring a Structure Variable

- A structure has to declared, after the body of structure has defined.

The syntax of declaring a structure is

```
struct <struct_name> <variable_name>;
```

The example to declare the variable for defined structure “employee”

```
struct employee e1;
```

Here e1 variable contains 6 members that are defined in structure.

Initializing a Structure Members

The members of individual structure variable is initialize one by one or in a single statement. The example to initialize a structure variable is

1) struct employee e1 = {1, "Hemant", 12000, "3 vikas colony new delhi", 10, 35};

2) e1.emp_id=1; e1.dept_no=1

e1.name="Hemant"; e1.age=35;

e1.salary=12000;

e1.address="3 vikas colony new delhi";

Accessing a Structure Members

- The structure members cannot be directly accessed in the expression. They are accessed by using the name of structure variable followed by a dot and then the name of member variable.
- The method used to access the structure variables are `e1.emp_id`, `e1.name`, `e1.salary`, `e1.address`, `e1.dept_no`, `e1.age`. The data with in the structure is stored and printed by this method using `scanf` and `printf` statement in c program.

Structure Assignment

- The value of one structure variable is assigned to another variable of same type using assignment statement. If the e1 and e2 are structure variables of type employee then the statement

$e1 = e2;$

- Assign value of structure variable e2 to e1. The value of each member of e2 is assigned to corresponding members of e1.

Program to implement the Structure

```
#include <stdio.h>
#include <conio.h>
struct employee
{
int emp_id;
char name[20];
float salary;
char address[50];
int dept_no;
int age;
};
```

Program to implement the Structure

```
void main ( )  
{ struct employee e1,e2;  
  printf ("Enter the employee id of employee");  
  scanf ("%d",&e1.emp_id);  
  printf ("Enter the name of employee");  
  scanf ("%s",e1.name);  
  printf ("Enter the salary of employee");  
  scanf ("%f",&e1.salary);  
  printf ("Enter the address of employee");  
  scanf ("%s",e1.address);  
  printf ("Enter the department of employee");  
  scanf ("%d",&e1.dept_no);  
  printf ("Enter the age of employee");
```

Program to implement the Structure

```
scanf("%d",&e1.age);  
printf ("Enter the employee id of employee");  
scanf("%d",&e2.emp_id);  
printf ("Enter the name of employee");  
scanf("%s",e2.name);  
printf ("Enter the salary of employee");  
scanf("%f",&e2.salary);  
printf ("Enter the address of employee");  
scanf("%s",e2.address);  
printf ("Enter the department of employee");  
scanf("%d",&e2.dept_no);  
printf ("Enter the age of employee");  
scanf("%d",&e2.age);
```

Program to implement the Structure

```
printf ("The employee id of employee is : %d", e1.emp_id);  
printf ("The name of employee is : %s",  
e1.name);  
printf ("The salary of employee is : %f",  
e1.salary);  
printf ("The address of employee is : %s",  
e1.address);  
printf ("The department of employee is : %d",  
e1.dept_no);  
printf ("The age of employee is : %d",  
e1.age);
```

Program to implement the Structure

```
printf ("The employee id of employee is : %d", e2.emp_id);  
printf ("The name of employee is : %s",  
e2.name);  
printf ("The salary of employee is : %f",  
e2.salary);  
printf ("The address of employee is : %s",  
e2.address);  
printf ("The department of employee is : %d",  
e2.dept_no);  
printf ("The age of employee is : %d",e2.age);  
getch();  
}
```

Output of Program

Enter the employee id of employee 1

Enter the name of employee Rahul

Enter the salary of employee 15000

Enter the address of employee 4,villa area, Delhi

Enter the department of employee 3

Enter the age of employee 35

Enter the employee id of employee 2

Enter the name of employee Rajeev

Enter the salary of employee 14500

Enter the address of employee flat 56H, Mumbai

Enter the department of employee 5

Enter the age of employee 30

Output of Program

The employee id of employee is : 1

The name of employee is : Rahul

The salary of employee is : 15000

The address of employee is : 4, villa area, Delhi

The department of employee is : 3

The age of employee is : 35

The employee id of employee is : 2

The name of employee is : Rajeev

The salary of employee is : 14500

The address of employee is : flat 56H, Mumbai

The department of employee is : 5

The age of employee is : 30

Array of Structure

- C language allows to create an array of variables of structure. The array of structure is used to store the large number of similar records.
- For example to store the record of 100 employees then array of structure is used. The method to define and access the array element of array of structure is similar to other array. The syntax to define the array of structure is

Struct <struct_name> <var_name> <array_name> [<value>];

For Example:-

Struct employee e1[100];

Program to implement the Array of Structure

```
#include <stdio.h>
#include <conio.h>
struct employee
{
int emp_id;
char name[20];
float salary;
char address[50];
int dept_no;
int age;
};
```

Program to implement the Array of Structure

```
void main ( )
{
    struct employee e1[5];
    int i;
    for (i=1; i<=100; i++)
    {
        printf ("Enter the employee id of employee");
        scanf ("%d",&e[i].emp_id);
        printf ("Enter the name of employee");
        scanf ("%s",e[i].name);
        printf ("Enter the salary of employee");
        scanf ("%f",&e[i].salary);
```

Program to implement the Array of Structure

```
printf ("Enter the address of employee");
scanf ("%s", e[i].address);
printf ("Enter the department of employee");
scanf ("%d",&e[i].dept_no);
printf ("Enter the age of employee");
scanf ("%d",&e[i].age);
}
for (i=1; i<=100; i++)
{
    printf ("The employee id of employee is : %d",
e[i].emp_id);
    printf ("The name of employee is: %s",e[i].name);
```

Program to implement the Array of Structure

```
printf ("The salary of employee is: %f",  
e[i].salary);  
printf ("The address of employee is : %s",  
e[i].address);  
printf ("The department of employee is : %d",  
e[i].dept_no);  
printf ("The age of employee is : %d", e[i].age);  
}  
getch();  
}
```

Structures within Structures

C language define a variable of structure type as a member of other structure type. The syntax to define the structure within structure is

```
struct <struct_name>{  
    <data_type> <variable_name>;  
        struct <struct_name>  
            { <data_type> <variable_name>;  
              .....}<struct_variable>;  
<data_type> <variable_name>;  
};
```

Example of Structure within Structure

The structure of Employee is declared as

```
struct employee
{
    int emp_id;
    char name[20];
    float salary;
    int dept_no;
    struct date
    {
        int day;
        int month;
        int year;
    }doj;
};
```


Accessing Structures within Structures

The data member of structure within structure is accessed by using two period (.) symbol. The syntax to access the structure within structure is

```
struct _var. nested_struct_var. struct_member;
```

For Example:-

```
e1.doj.day;
```

```
e1.doj.month;
```

```
e1.doj.year;
```

Pointers and Structures

C language can define a pointer variable of structure type. The pointer variable to structure variable is declared by using same syntax to define a pointer variable of data type. The syntax to define the pointer to structure

```
struct <struct_name> *<pointer_var_name>;
```

For Example:

```
struct employee *emp;
```

It declare a pointer variable “emp” of employee type.

Access the Pointer in Structures

The member of structure variable is accessed by using the pointer variable with arrow operator(\rightarrow) instead of period operator(.). The syntax to access the pointer to structure.

```
pointer_var_name $\rightarrow$ structure_member;
```

For Example:

```
emp $\rightarrow$ name;
```

Here “name” structure member is accessed through pointer variable emp.

Passing Structure to Function

The structure variable can be passed to a function as a parameter. The program to pass a structure variable to a function.

```
#include <stdio.h>
#include <conio.h>
struct employee
{
int emp_id;
char name[20];
float salary;
};
```

Passing Structure to Function

```
void main ( )  
{  
    struct employee e1;  
    printf ("Enter the employee id of employee");  
    scanf ("%d",&e1.emp_id);  
    printf ("Enter the name of employee");  
    scanf ("%s",e1.name);  
    printf ("Enter the salary of employee");  
    scanf ("%f",&e1.salary);  
    printdata (struct employee e1);  
    getch();  
}
```

Passing Structure to Function

```
void printdata( struct employee emp)
{
    printf (“\nThe employee id of employee is :
%d”, emp.emp_id);
    printf (“\nThe name of employee is : %s”,
        emp.name);
    printf (“\nThe salary of employee is : %f”,
        emp.salary);
}
```

Function Returning Structure

The function can return a variable of structure type like a integer and float variable. The program to return a structure from function.

```
#include <stdio.h>
#include <conio.h>
struct employee
{
int emp_id;
char name[20];
float salary;
};
```

Function Returning Structure

```
void main ( )
{
    struct employee emp;
    emp=getdata();
    printf (“\nThe employee id of employee is :%d”, emp.emp_id);
    printf (“\nThe name of employee is : %s”,
emp.name);
    printf (“\nThe salary of employee is : %f”,
emp.salary);
    getch();
}
```


Function Returning Structure

```
struct employee getdata( )  
{  
    struct employee e1;  
    printf ("Enter the employee id of employee");  
    scanf ("%d",&e1.emp_id);  
    printf ("Enter the name of employee");  
    scanf ("%s",e1.name);  
    printf ("Enter the salary of employee");  
    scanf ("%f",&e1.salary);  
    return(e1);  
}
```

Union Data Type

- A union is a user defined data type like structure. The union groups logically related variables into a single unit.
- The union data type allocate the space equal to space need to hold the largest data member of union. The union allows different types of variable to share same space in memory.
- There is no other difference between structure and union than internal difference. The method to declare, use and access the union is same as structure.

Defining of Union

A union has to be defined, before it can be used. The syntax of defining a structure is

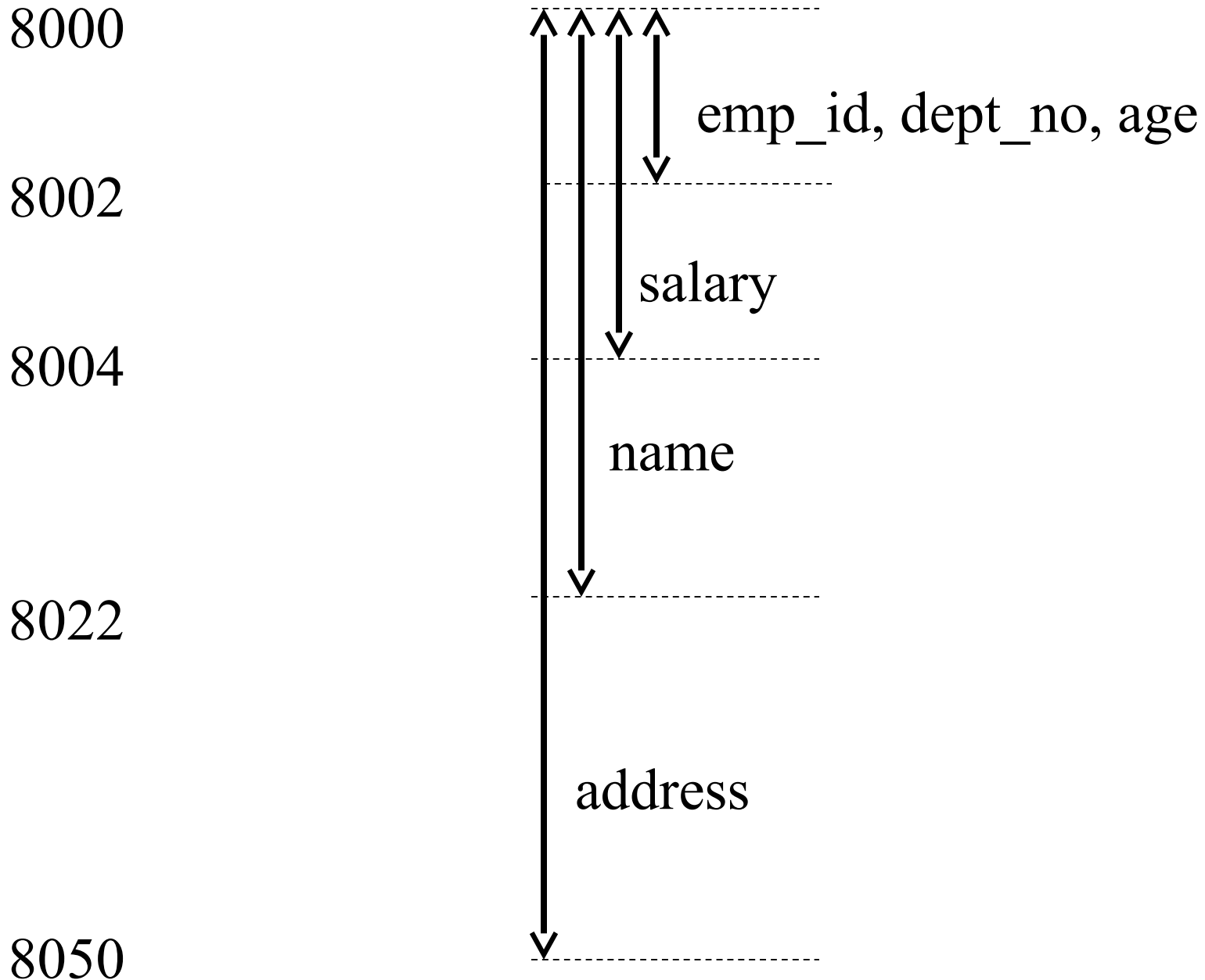
```
union <union_name>
{
    <data_type> <variable_name>;
    <data_type> <variable_name>;
    .....
    <data_type> <variable_name>;
};
```

Example of Union

The union of Employee is declared as

```
union employee
{
int emp_id;
char name[20];
float salary;
char address[50];
int dept_no;
int age;
};
```

Memory Space Allocation



Difference between Structures & Union

1)The memory occupied by structure variable is the sum of sizes of all the members but memory occupied by union variable is equal to space hold by the largest data member of a union.

2)In the structure all the members are accessed at any point of time but in union only one of union member can be accessed at any given time.

Application of Structures

Structure is used in database management to maintain data about books in library, items in store, employees in an organization, financial accounting transaction in company.

Beside that other application are

- 1) Changing the size of cursor.
- 2) Clearing the contents of screen.
- 3) Drawing any graphics shape on screen.
- 4) Receiving the key from the keyboard.

Application of Structures

- 5) Placing cursor at defined position on screen.
- 6) Checking the memory size of the computer.
- 7) Finding out the list of equipments attach to computer.
- 8) Hiding a file from the directory.
- 9) Sending the output to printer.
- 10) Interacting with the mouse.
- 11) Formatting a floppy.
- 12) Displaying the directory of a disk.