Unit :IV

Array of pointers to structures

Structure Pointers

- C allows pointers to structures just as it allows pointers to any other type of object.

Declaring a Structure Pointer

- structure pointers are declared by placing * in front of a structure variable's name.
- For example

```
struct addr *addr_pointer;

struct bal {
float balance;
char name[80];
} person;
struct bal *p; /* declare a structure pointer */
```

Using Structure Pointers

- There are two primary uses for structure pointers:
- To pass a structure to a function using call by reference
- To create linked lists and other dynamic data structures that rely on dynamic allocation.

Advantages

- When a pointer to a structure is passed to a function, only the address of the structure is passed
- This makes for very fast function calls.
- A second advantage, in some cases, is that passing a pointer makes it possible for the function to modify the contents of the structure used as the argument.

address of a structure variable

- Syntax

ptrname=&structurevariablename

```
struct bal {
float balance;
char name[80];
} person;
struct bal *p; /* declare a structure pointer */
```

This places the address of the structure **person into the pointer p:**

```
p = &person;
```

- There are two ways of accessing members of structure using pointer:
- Using indirection (*) operator and dot (.) operator.
- Using arrow (->) operator or membership operator.

Access the members of a structure using a pointer

- Example for accessing the members of a structure using a pointer to that structure, using the arrow operator ->

p–>balance

- Example for accessing the members of a structure using a pointer to that structure, using the indirection * and dot operator .

(*p).balance

- Parentheses around *p are necessary because the precedence of dot(.) operator is greater than that of indirection (*) operator.

example1:

```
#include <stdio.h>
 struct person { int age; float weight; };
int main()
 { struct person *personPtr, person1;
   personPtr = &person1;
   printf("Enter age:");
   scanf("%d", &personPtr->age);
    printf("Enter weight:");
    scanf("%f", &personPtr->weight);
     printf("Displaying:\n");
      printf("Age: %d\n", personPtr->age);
      printf("weight: %f", personPtr->weight);
return 0; }
```

- Example 2:

```
#include <stdio.h>
typedef struct Complex
{
     float real;
     float imag;
} complex;

void addNumbers(complex c1, complex c2, complex *result);
```

```c
int main()
{
    complex c1, c2, result;

    printf("For first number,\n");
    printf("Enter real part: ");
    scanf("%f", &c1.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c1.imag);

    printf("For second number, \n");
    printf("Enter real part: ");
    scanf("%f", &c2.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c2.imag);

    addNumbers(c1, c2, &result);
    printf("\nresult.real = %.1f\n", result.real);
    printf("result.imag = %.1f", result.imag);

    return 0;
}
```

```
void addNumbers(complex c1, complex c2, complex *result)
{
        result->real = c1.real + c2.real;
        result->imag = c1.imag + c2.imag;
}
```