6.1 Concept of Structure

◇ Definition:

A **structure** in C is a user-defined data type that allows grouping of different types of variables under one name.

It is used to represent a record.

◇ Syntax:

```
struct structure_name
{
    data_type member1;
    data_type member2;
    ...
};
```

☑ Example:

```
#include <stdio.h>

struct Student
{
    int roll;
    char name[20];
    float marks;
};

int main()
{
    struct Student s1;

    s1.roll = 1;
    s1.marks = 85.5;

    printf("Roll = %d\n", s1.roll);
    printf("Marks = %.2f\n", s1.marks);
```

```
    return 0;
}
```

6.2 Initializing and Accessing Members of Structure

◇ Initialization at Declaration

```
struct Student
{
    int roll;
    float marks;
};

int main()
{
    struct Student s1 = {1, 90.5};
}
```

◇ Accessing Members

Use **dot (.) operator**

```
variable.member_name
```

Example:

```
printf("%d", s1.roll);
```

6.3 Array of Structure

Used when storing multiple records.

- **Syntax:**

```
struct Student s[5];
```

☑ Example Program

```
#include <stdio.h>

struct Student
{
    int roll;
    float marks;
};

int main()
{
    struct Student s[2];
    int i;

    for(i = 0; i < 2; i++)
    {
        printf("Enter roll and marks: ");
        scanf("%d %f", &s[i].roll, &s[i].marks);
    }

    for(i = 0; i < 2; i++)
    {
        printf("Roll = %d, Marks = %.2f\n", s[i].roll, s[i].marks);
    }

    return 0;
}
```

6.4 Pointer to Structure

A pointer can store the address of a structure variable.

◇ Declaration:

```
struct Student *ptr;
```

◇ Access Members Using Pointer

Use **arrow (->) operator**

```
ptr->member_name
```

☑ Example

```
#include <stdio.h>

struct Student
{
    int roll;
    float marks;
};

int main()
{
    struct Student s1 = {1, 88.5};
    struct Student *ptr = &s1;

    printf("Roll = %d\n", ptr->roll);
    printf("Marks = %.2f\n", ptr->marks);

    return 0;
}
```

6.5 Union

◇ Definition:

A **union** is similar to a structure, but all members share the same memory location.

Syntax:

```
union union_name
{
    data_type member1;
    data_type member2;
};
```

Example

```
#include <stdio.h>

union Data
{
    int i;
    float f;
};

int main()
{
    union Data d;

    d.i = 10;
    printf("Integer = %d\n", d.i);

    d.f = 5.5;
    printf("Float = %.2f\n", d.f);

    return 0;
}
```

When one member is updated, other members may lose their values because they share memory.

6.6 Difference Between Structure and Union

Feature	Structure	Union
Memory	Separate memory for each member	Shared memory

Size	Sum of all members	Size of largest member
Value storage	All members can store values simultaneously	Only one member at a time
Keyword	struct	union
Memory usage	More	Less