Chapter 5: Structure and Union

6.1 Basic Concept:
A structure is a user-defined data type in C that allows you to group together variables of different types under a single name. It is used to represent a collection of related data items.
Example:

```c
#include <stdio.h>

// Structure declaration
struct student {
    int rollNumber;
    char name[50];
    int age;
};

int main() {
    // Structure variable declaration and initialization
    struct student s1 = { 1, "John", 20 };

    // Accessing structure members
    printf("Roll Number: %d\n", s1.rollNumber);
    printf("Name: %s\n", s1.name);
    printf("Age: %d\n", s1.age);

    return 0;
}
```

6.2 Structure declaration, initialization:
Structure declaration involves defining the structure type and its members, specifying the data type and name of each member. Structure initialization is the process of assigning values to the members of a structure variable.
Example:

```c
#include <stdio.h>

// Structure declaration
struct point {
    int x;
    int y;
};

int main() {
    // Structure variable declaration and initialization
    struct point p1 = { 3, 5 };

    // Accessing structure members
    printf("x-coordinate: %d\n", p1.x);
    printf("y-coordinate: %d\n", p1.y);

    return 0;
}
```

6.3 Structure within structure:
A structure can contain another structure as one of its members. This is known as a nested structure or a structure within a structure.
Example:
```c
#include <stdio.h>

// Structure declaration
struct address {
    char street[50];
    char city[50];
};

// Structure declaration with nested structure
struct person {
    char name[50];
    int age;
    struct address addr;
};

int main() {
    // Structure variable declaration and initialization
    struct person p1 = { "John", 25, { "123 Main St", "New York" } };

    // Accessing structure members
    printf("Name: %s\n", p1.name);
    printf("Age: %d\n", p1.age);
    printf("Address: %s, %s\n", p1.addr.street, p1.addr.city);

    return 0;
}
```

6.4 Nested Structures:
Nested structures are structures that are defined within another structure. They allow for hierarchical data representation, where the inner structures are accessed through the outer structure.
Example:
```c
#include <stdio.h>

// Structure declaration
struct date {
    int day;
    int month;
    int year;
};

// Structure declaration with nested structures
struct employee {
    char name[50];
    int empId;
    struct date dob;
};

int main() {
    // Structure variable declaration and initialization
    struct employee emp1 = { "John Doe", 1234, { 10, 5, 1990 } };
    // Accessing structure members
    printf("Name: %s\n", emp1.name);
    printf("Employee ID: %d\n", emp1.empId);
    printf("Date of Birth: %d/%d/%d\n", emp1.dob.day, emp1.dob.month,
emp1.dob.year);

    return 0;
}
```

6.5 Array of Structure:
An array of structures is a collection of multiple structures of the same type. It allows you to store and manipulate multiple instances of a structure together.
Example:

```
#include <stdio.h>

// Structure declaration
struct student {
    int rollNumber;
    char name[50];
    int age;
};

int main() {
    // Array of structures declaration and initialization
    struct student students[3] = {
        { 1, "John", 20 },
        { 2, "Alice", 21 },
        { 3, "Bob", 19 }
    };

    // Accessing structure members
    for (int i = 0; i < 3; i++) {
        printf("Student %d:\n", i + 1);
        printf("Roll Number: %d\n", students[i].rollNumber);
        printf("Name: %s\n", students[i].name);
        printf("Age: %d\n", students[i].age);
        printf("\n");
    }

    return 0;
}
```

In this example, an array of structures named `students` is declared and initialized with three instances of the `student` structure. The members of each structure within the array can be accessed using the dot operator (`.`).