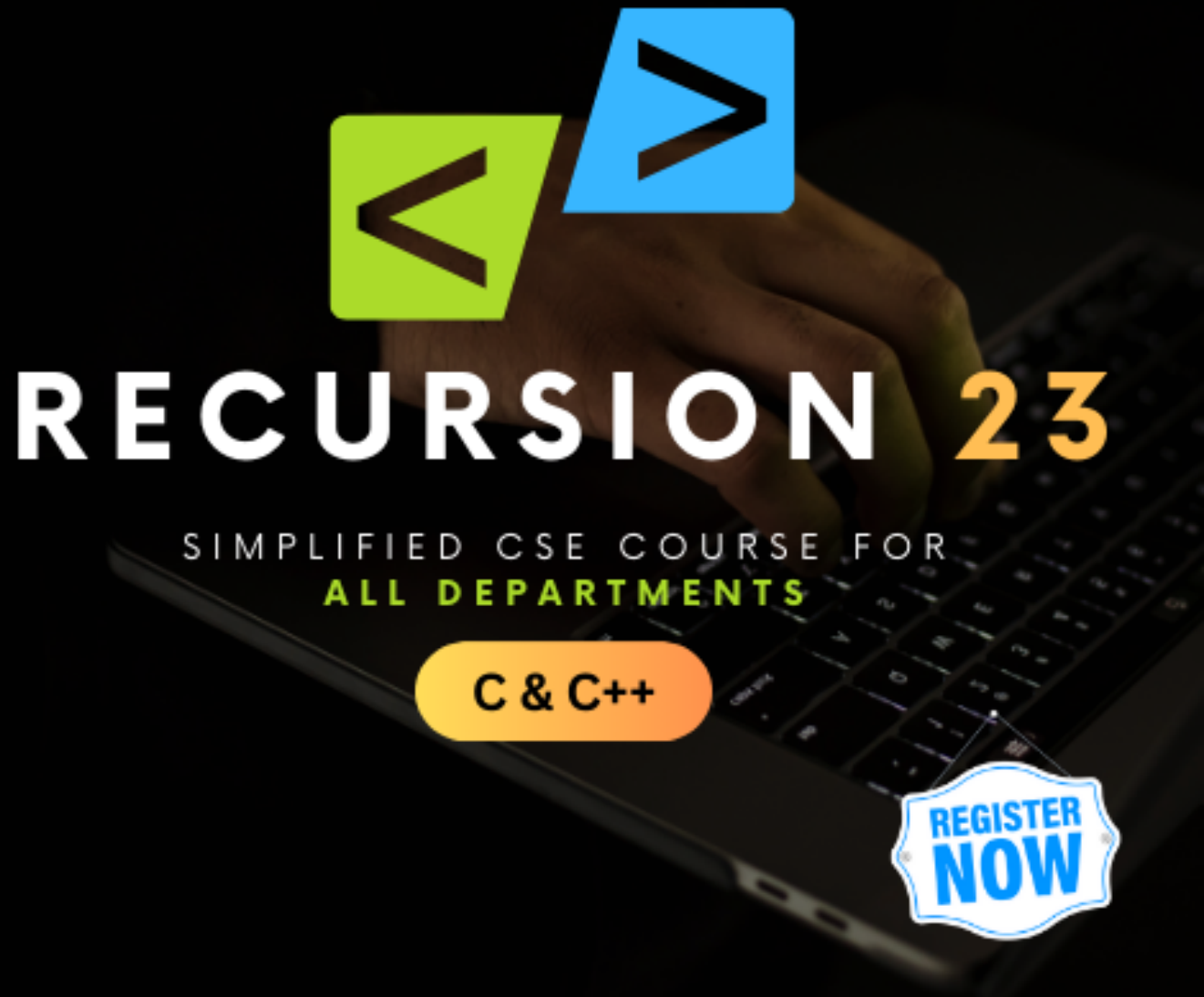
Lecture 25

Structure & Union



What is a Structure?

A structure is a user-defined data type that groups related variables of different data types into a single entity. Each variable within a structure is called a member.

Usage:

Structures are used when you need to store data items of different types together, representing a complex data item. For instance, a structure can represent a record in a database, a point in a coordinate system, or a date.

Structure Example

```
struct Person {
    char name[50];
    int age;
    float salary;
};
```

Structure Sytax

```
struct MyStrcuture{ //Declaration
    int number; //Member
    char letter; //Member
}; //End With semicolon
```

Use case

Problem: Awesh's Employee Management System

```
Awesh is managing a small company and wants to keep track of his employees' information. He needs a
program that can store and display the following details for each employee:

Employee ID (integer)
Name (string)
Age (integer)
Salary (float)
Your task is to create a structure to represent an employee and write a program to input and display the
details of N employees.
```

Defining a struct

```c
// Define the Employee structure
struct Employee {
    int id;
    char name[50];
    int age;
    float salary;
};
```

Declaring and Accessing

```c
#include <stdio.h>
#include <string.h> // For strcpy
// Define the Employee structure
struct Employee {
    int id;
    char name[50];
    int age;
    float salary;
};
int main() {
    // Declare a single Employee structure
variistruct Employee employee1;

    // Assign values to the employee
    employee1.id = 101;
    strcpy(employee1.name, "John");
    employee1.age = 28;
    employee1.salary = 50000.00;

    // Print details of the employee
    printf("Employee ID: %d\n", employee1.id);
    printf("Name: %s\n", employee1.name);
    printf("Age: %d\n", employee1.age);
    printf("Salary: %.2f\n", employee1.salary);

    return 0;
}
```

Another Example

```c
#include <stdio.h>

// Define the Point structure
struct Point {
    int x;
    int y;
};

int main() {
    // Declare and initialize a Point structure
vari
    struct Point p1 = {1, 2};

    // Print the coordinates of the point
    printf("X-coordinate: %d\n", p1.x);
    printf("Y-coordinate: %d\n", p1.y);

    return 0;
}
```

Typedef

```c
#include <stdio.h>

// Define the Point structure
typdef struct {
    int x;
    int y;
} Point;

int main() {
    // Declare and initialize a Point structure
variPoint p1 = {1, 2};

    // Print the coordinates of the point
    printf("X-coordinate: %d\n", p1.x);
    printf("Y-coordinate: %d\n", p1.y);

    return 0;
}
```

Nested Structure

```c
struct Address {
    char city[30];
    int zip;
};

struct Person {
    char name[50];
    int age;
    struct Address addr;
};
```

Example code

```c
#include <stdio.h>
#include <string.h>

// Define the Address structure
struct Address {
    char city[30];
    int zip;
};

// Define the Person structure with a nested Address structure
struct Person {
    char name[50];
    int age;
    struct Address addr;
};

int main() {
    // Declare and initialize a Person structure variable
    struct Person person1;

    // Assign values to the Person structure members
    strcpy(person1.name, "Alice");
    person1.age = 28;

    // Assign values to the nested Address structure members
    strcpy(person1.addr.city, "New York");
    person1.addr.zip = 10001;

    // Print the Person details including the nested Address
    printf("Name: %s\n", person1.name);
    printf("Age: %d\n", person1.age);
    printf("City: %s\n", person1.addr.city);
    printf("ZIP Code: %d\n", person1.addr.zip);

    return 0;
}
```

Structure Pointer

```c
// C program to illustrate the structure pointer
#include <stdio.h>

// structure declaration
struct Point {
    int x, y;
};

int main()
{
    struct Point str = { 1, 2 };

    // p2 is a pointer to structure p1
    struct Point* ptr = &str;

    // Accessing structure members using structure pointer
    printf("%d %d", ptr->x, ptr->y);

    return 0;
}
```

Structure Array

```c
#include <stdio.h>
#include <string.h>
// Define the Employee structure
struct Employee {
    int id;
    char name[50];
    int age;
    float salary;
};
int main() {
    // Declare an array of Employee structures
    struct Employee employees[3];
    // Assign values to the first employee
    employees[0].id = 101;
    strcpy(employees[0].name, "John");
    employees[0].age = 28;
    employees[0].salary = 50000.00;
    // Assign values to the second employee
    employees[1].id = 102;
    strcpy(employees[1].name, "Alice");
    employees[1].age = 30;
    employees[1].salary = 55000.00;
    // Assign values to the third employee
    employees[2].id = 103;
    strcpy(employees[2].name, "Bob");
    employees[2].age = 25;
    employees[2].salary = 48000.00;
    // Print details of all employees
    for(int i = 0; i < 3; i++) {
        printf("Employee ID: %d, Name: %s, Age: %d, Salary: %.2f\n",
               employees[i].id, employees[i].name, employees[i].age,
employees[i].salary);
    }
    return 0;
}
```

Structure Array

```c
#include <stdio.h>
#include <string.h>
// Define the Employee structure
struct Employee {
    int id;
    char name[50];
    int age;
    float salary;
};
int main() {
    // Declare an array of Employee structures
    struct Employee employees[3];
    // Assign values to the first employee
    employees[0].id = 101;
    strcpy(employees[0].name, "John");
    employees[0].age = 28;
    employees[0].salary = 50000.00;
    // Assign values to the second employee
    employees[1].id = 102;
    strcpy(employees[1].name, "Alice");
    employees[1].age = 30;
    employees[1].salary = 55000.00;
    // Assign values to the third employee
    employees[2].id = 103;
    strcpy(employees[2].name, "Bob");
    employees[2].age = 25;
    employees[2].salary = 48000.00;
    // Print details of all employees
    for(int i = 0; i < 3; i++) {
        printf("Employee ID: %d, Name: %s, Age: %d, Salary: %.2f\n",
               employees[i].id, employees[i].name, employees[i].age,
employees[i].salary);
    }
    return 0;
}
```

Problem:Structuring and Grading Multiple Students

```
Problem Statement:
Awesh, a diligent teacher, needs to manage the information of multiple students and calculate their
grades based on their exam performance. He has asked you to write a C program that organizes and
processes this information using structures.

You need to create a Student structure that includes the following:

#A nested Name structure that holds:
first_name: A string representing the student's first name.
last_name: A string representing the student's last name.

#The Student structure should also include:
roll_number: An integer representing the student's roll number.
percentage: A float representing the student's total percentage in the exam.
grade: A character representing the student's grade.

#You must implement the following functionalities:

"Input Function":
A function to input the details of multiple students (first name, last name, roll number, and
percentage).

"Grade Calculation Function":
A separate function to calculate each student grade based on their percentage. The grading criteria are
as follows:
A for percentage ≥ 90
B for percentage ≥ 80 and < 90
C for percentage ≥ 70 and < 80
D for percentage ≥ 60 and < 70
F for percentage < 60

"Output Function":
A function to display each student's full name, roll number, percentage, and grade.
```

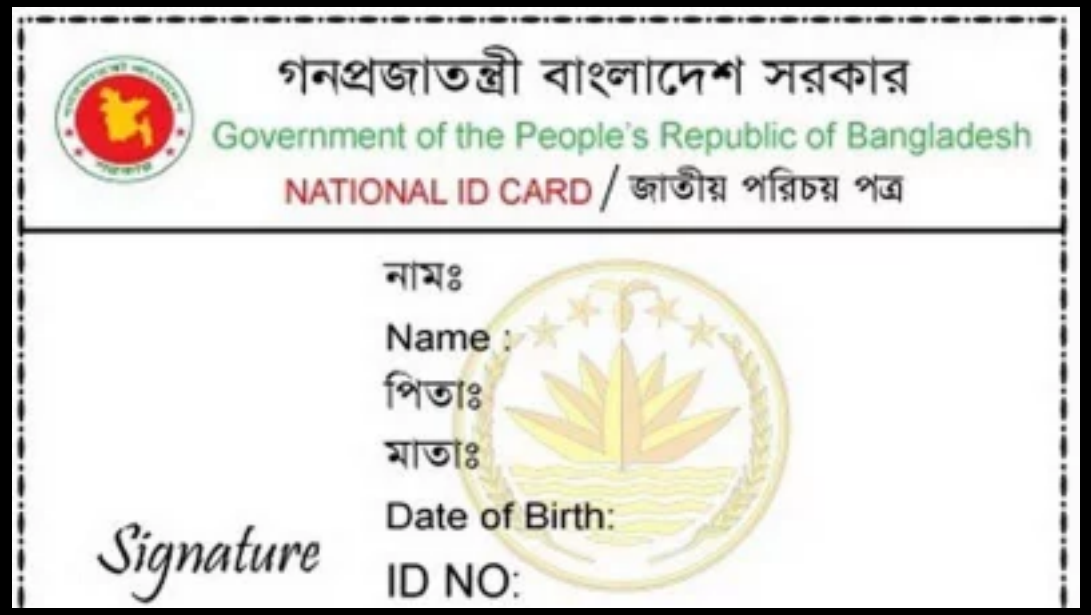
Solution

```c
#include <stdio.h>
#include <string.h>

// Define the Name structure
struct Name {
    char first_name[50];
    char last_name[50];
};

// Define the Student structure with a nested Name structure
struct Student {
    struct Name name;
    int roll_number;
    float percentage;
    char grade;
};

// Function to calculate the grade based on percentage
char calculateGrade(float percentage) {
    if (percentage ≥ 90.0)
        return 'A';
    else if (percentage ≥ 80.0)
        return 'B';
    else if (percentage ≥ 70.0)
        return 'C';
    else if (percentage ≥ 60.0)
        return 'D';
    else
        return 'F';
}
```

```c
// Function to input the details of multiple students
void inputStudents(struct Student students[], int n) {
    for (int i = 0; i < n; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("First Name: ");
        scanf("%s", students[i].name.first_name);
        printf("Last Name: ");
        scanf("%s", students[i].name.last_name);
        printf("Roll Number: ");
        scanf("%d", &students[i].roll_number);
        printf("Percentage: ");
        scanf("%f", &students[i].percentage);

        // Calculate the grade for the student
        students[i].grade = calculateGrade(students[i].percentage);
    }
}

// Function to display the details of all students
void displayStudents(struct Student students[], int n) {
    for (int i = 0; i < n; i++) {
        printf("\nDetails of student %d:\n", i + 1);
        printf("Name: %s %s\n", students[i].name.first_name,
students[i].name.last_name);
        printf("Roll Number: %d\n", students[i].roll_number);
        printf("Percentage: %.2f\n", students[i].percentage);
        printf("Grade: %c\n", students[i].grade);
    }
}

int main() {
    int n;

    // Input the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Declare an array of Student structures
    struct Student students[n];

    // Input the details of all students
    inputStudents(students, n);

    // Display the details of all students
    displayStudents(students, n);

    return 0;
}
```

Your task

Make a structure for NID card.
Take input of 5people.
Calculate their age and show output.



Union

```c
#include <stdio.h>
// Define a union that can store an integer, float, or character
union Data {
    int i;
    float f;
    char c;
};
int main() {
    // Declare a union variable
    union Data data;

    // Store an integer in the union
    data.i = 10;
    printf("data.i: %d\n", data.i);

    // Store a float in the union (overwrites the integer)
    data.f = 220.5;
    printf("data.f: %.2f\n", data.f);

    // Store a character in the union (overwrites the float)
    data.c = 'A';
    printf("data.c: %c\n", data.c);

    return 0;
}
```

Problem of Union

```c
#include <stdio.h>
// Define a union that can store an integer, float, or character
union Data {
    int i;
    float f;
    char c;
};
int main() {
    // Declare a union variable
    union Data data;
    // Store an integer in the union
    data.i = 10;
    printf("data.i: %d\n", data.i);
    // Store a float in the union (overwrites the integer)
    data.f = 220.5;
    printf("data.f: %.2f\n", data.f);
    // Store a character in the union (overwrites the float)
    data.c = 'A';
    printf("data.c: %c\n", data.c);
    // Note: Only the last stored value is valid, as all members share the same memory
locaprintf("After storing char, data.i: %d\n", data.i);
    printf("After storing char, data.f: %.2f\n", data.f);
    return 0;
}
```