**Defining a Structure**

struct StructureName {

data\_type member1;

data\_type member2;

};

### ****Example: Defining and Using a Structure****

#include <stdio.h>

// Define a structure

struct Student {

char name[50];

int age;

float marks;

};

int main() {

// Declare a structure variable

struct Student student1;

// Input data into the structure

printf("Enter name: ");

fgets(student1.name, sizeof(student1.name), stdin);

printf("Enter age: ");

scanf("%d", &student1.age);

printf("Enter marks: ");

scanf("%f", &student1.marks);

// Output the data

printf("\nStudent Information:\n");

printf("Name: %s", student1.name);

printf("Age: %d\n", student1.age);

printf("Marks: %.2f\n", student1.marks)

return 0;

}

### 1. Array of Structures

#include <stdio.h>

struct Student {

char name[50];

int age;

float marks;

};

int main() {

struct Student students[3]; // Array of 3 students

// Input data for each student

for (int i = 0; i < 3; i++) {

printf("Enter details for student %d:\n", i + 1);

printf("Name: ");

getchar(); // Clear input buffer

fgets(students[i].name, sizeof(students[i].name), stdin);

printf("Age: ");

scanf("%d", &students[i].age);

printf("Marks: ");

scanf("%f", &students[i].marks);

}

// Output data for each student

printf("\nStudent Details:\n");

for (int i = 0; i < 3; i++) {

printf("\nStudent %d:\n", i + 1);

printf("Name: %s", students[i].name);

printf("Age: %d\n", students[i].age);

printf("Marks: %.2f\n", students[i].marks);

}

return 0;

}

#### **2. Nested Structures**

#include <stdio.h>

struct Address {

char city[50];

int zip;

};

struct Employee {

char name[50];

int id;

struct Address address; // Nested structure

};

int main() {

struct Employee emp;

// Input data

printf("Enter employee name: ");

fgets(emp.name, sizeof(emp.name), stdin);

printf("Enter employee ID: ");

scanf("%d", &emp.id);

printf("Enter city: ");

getchar(); // Clear buffer

fgets(emp.address.city, sizeof(emp.address.city), stdin);

printf("Enter ZIP code: ");

scanf("%d", &emp.address.zip);

// Output data

printf("\nEmployee Details:\n");

printf("Name: %s", emp.name);

printf("ID: %d\n", emp.id);

printf("City: %s", emp.address.city);

printf("ZIP: %d\n", emp.address.zip);

return 0;

}

#### **3. Passing Structures to Functions**

#include <stdio.h>

struct Rectangle {

int length;

int width;

};

int calculateArea(struct Rectangle rect) {

return rect.length \* rect.width;

}

int main() {

struct Rectangle rect = {10, 5}; // Initialize structure

int area = calculateArea(rect); // Pass structure to function

printf("Area of rectangle: %d\n", area);

return 0;

}

### ****Problem: Manage Student Grades****

You need to create a program to manage and analyze student grades using structures. The program should:

1. Define a structure Student with the following attributes:
   * name (string)
   * rollNumber (integer)
   * marks (array of 3 integers for 3 subjects)
2. Use an array of structures to handle multiple students.
3. Implement the following functions:
   * inputDetails(): Inputs the details of all students (loop through the array of structures).
   * calculateAverage(): Calculates the average marks of a student (passed as an argument).
   * displayDetails(): Displays the name, roll number, marks, and average of all students.
   * findTopper(): Identifies and displays the details of the student with the highest average marks.
4. Use a loop to iterate through the students' data in all the above functions.

***INPUT*** Enter the number of students: 3

Enter details for student 1:

Name: Alice

Roll Number: 1

Marks (3 subjects): 85 90 78

Enter details for student 2:

Name: Bob

Roll Number: 2

Marks (3 subjects): 70 75 80

Enter details for student 3:

Name: Charlie

Roll Number: 3

Marks (3 subjects): 92 88 95

***OUTPUT*** Student Details:

Name: Alice, Roll Number: 1, Marks: 85, 90, 78, Average: 84.33

Name: Bob, Roll Number: 2, Marks: 70, 75, 80, Average: 75.00

Name: Charlie, Roll Number: 3, Marks: 92, 88, 95, Average: 91.67

Topper:

Name: Charlie, Roll Number: 3, Average: 91.67

#include <stdio.h>

// Define the Student structure

struct Student {

char name[50];

int rollNumber;

int marks[3];

};

// Function to input student details

void inputDetails(struct Student students[], int n) {

for (int i = 0; i < n; i++) {

printf("Enter details for student %d:\n", i + 1);

printf("Name: ");

getchar(); // Clear input buffer

fgets(students[i].name, sizeof(students[i].name), stdin);

students[i].name[strcspn(students[i].name, "\n")] = '\0'; // Remove newline

printf("Roll Number: ");

scanf("%d", &students[i].rollNumber);

printf("Marks (3 subjects): ");

for (int j = 0; j < 3; j++) {

scanf("%d", &students[i].marks[j]);

}

}

}

// Function to calculate the average marks of a student

float calculateAverage(struct Student student) {

int total = 0;

for (int i = 0; i < 3; i++) {

total += student.marks[i];

}

return total / 3.0;

}

// Function to display all student details

void displayDetails(struct Student students[], int n) {

printf("\nStudent Details:\n");

for (int i = 0; i < n; i++) {

float avg = calculateAverage(students[i]);

printf("Name: %s, Roll Number: %d, Marks: ", students[i].name, students[i].rollNumber);

for (int j = 0; j < 3; j++) {

printf("%d ", students[i].marks[j]);

}

printf(", Average: %.2f\n", avg);

}

}

// Function to find and display the topper

void findTopper(struct Student students[], int n) {

int topperIndex = 0;

float highestAvg = calculateAverage(students[0]);

for (int i = 1; i < n; i++) {

float avg = calculateAverage(students[i]);

if (avg > highestAvg) {

highestAvg = avg;

topperIndex = i;

}

}

printf("\nTopper:\n");

printf("Name: %s, Roll Number: %d, Average: %.2f\n",

students[topperIndex].name,

students[topperIndex].rollNumber,

highestAvg);

}

int main() {

int n;

// Input the number of students

printf("Enter the number of students: ");

scanf("%d", &n);

struct Student students[n];

// Input student details

inputDetails(students, n);

// Display all details

displayDetails(students, n);

// Find and display the topper

findTopper(students, n);

return 0;

}

### ****Key Concepts in the Code****

1. **Hierarchical Data Organization**:
   * Students are assigned to teachers.
   * Teachers are assigned to staff.
2. **Attendance-Based Evaluation**:
   * **Student Attendance**: Used to identify the top student.
   * **Teacher Attendance**: Average attendance of students assigned to them.
   * **Staff Attendance**: Average attendance of teachers they assist.
3. **Functions for Calculation**:
   * Each level uses a function to find the top-performing entity.
4. **Predefined Input**:
   * Data is hard-coded for simplicity but can be modified to use dynamic input (scanf).
5. **Scalability**:
   * The solution can be extended to handle more students, teachers, and staff dynamically.

This program demonstrates efficient data management and real-world hierarchical relationships using **structures** and **nested logic**.

4o

#include <stdio.h>

#include <string.h>

#define NUM\_STUDENTS 5

#define NUM\_TEACHERS 3

#define NUM\_STAFF 2

// Structure Definitions

typedef struct {

char name[50];

int roll;

float attendance; // Percentage attendance

} Student;

typedef struct {

char name[50];

int id;

Student students[NUM\_STUDENTS];

int numStudents; // Number of students assigned

} Teacher;

typedef struct {

char name[50];

int id;

Teacher teachers[NUM\_TEACHERS];

int numTeachers; // Number of teachers assigned

} Staff;

// Function Prototypes

Student findTopStudent(Student students[], int n);

Teacher findTopTeacher(Teacher teachers[], int n);

Staff findTopStaff(Staff staff[], int n);

int main() {

// Input Students

Student students[NUM\_STUDENTS] = {

{"Alice", 1, 85.5},

{"Bob", 2, 90.0},

{"Charlie", 3, 88.2},

{"David", 4, 92.5},

{"Eve", 5, 89.7}

};

// Input Teachers

Teacher teachers[NUM\_TEACHERS] = {

{"Mr. Smith", 101, {students[0], students[1]}, 2},

{"Ms. Johnson", 102, {students[2], students[3]}, 2},

{"Mr. Brown", 103, {students[4]}, 1}

};

// Input Staff

Staff staff[NUM\_STAFF] = {

{"Mr. Green", 201, {teachers[0], teachers[1]}, 2},

{"Ms. White", 202, {teachers[2]}, 1}

};

// Find Top Entities

Student topStudent = findTopStudent(students, NUM\_STUDENTS);

Teacher topTeacher = findTopTeacher(teachers, NUM\_TEACHERS);

Staff topStaff = findTopStaff(staff, NUM\_STAFF);

// Display Results

printf("Top Student: %s (Roll: %d, Attendance: %.2f%%)\n", topStudent.name, topStudent.roll, topStudent.attendance);

printf("Top Teacher: %s (ID: %d, Avg Student Attendance: %.2f%%)\n", topTeacher.name, topTeacher.id, topTeacher.students[0].attendance);

printf("Top Staff: %s (ID: %d, Avg Teacher Attendance: %.2f%%)\n", topStaff.name, topStaff.id, topTeacher.students[0].attendance);

return 0;

}

// Function Definitions

Student findTopStudent(Student students[], int n) {

Student top = students[0];

for (int i = 1; i < n; i++) {

if (students[i].attendance > top.attendance) {

top = students[i];

}

}

return top;

}

Teacher findTopTeacher(Teacher teachers[], int n) {

Teacher top = teachers[0];

float maxAvgAttendance = 0;

for (int i = 0; i < n; i++) {

float sum = 0;

for (int j = 0; j < teachers[i].numStudents; j++) {

sum += teachers[i].students[j].attendance;

}

float avgAttendance = sum / teachers[i].numStudents;

if (avgAttendance > maxAvgAttendance) {

maxAvgAttendance = avgAttendance;

top = teachers[i];

}

}

return top;

}

Staff findTopStaff(Staff staff[], int n) {

Staff top = staff[0];

float maxAvgAttendance = 0;

for (int i = 0; i < n; i++) {

float sum = 0;

for (int j = 0; j < staff[i].numTeachers; j++) {

float teacherAttendanceSum = 0;

for (int k = 0; k < staff[i].teachers[j].numStudents; k++) {

teacherAttendanceSum += staff[i].teachers[j].students[k].attendance;

}

sum += teacherAttendanceSum / staff[i].teachers[j].numStudents;

}

float avgAttendance = sum / staff[i].numTeachers;

if (avgAttendance > maxAvgAttendance) {

maxAvgAttendance = avgAttendance;

top = staff[i];

}

}

return top;

}