Department of Computer Science and Engineering,

PES University, Bangalore, India

Lecture Notes
Problem Solving With C
UE24CS151B

Lecture #12
Problem Solving using Structures and Functions

By,
Prof. Sindhu R Pai,
Theory Anchor, Feb-May, 2025
Assistant Professor
Dept. of CSE, PESU

Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Center, PES University)
Prof. Nitin V Pujari (Dean, Internal Quality Assurance Cell, PES University)

Unit #: 3

Unit Name: Text Processing and User Defined Types

Topic: Problem Solving using Structures and Functions

Course objectives: The objective(s) of this course is to make students

- Acquire knowledge on how to solve relevant and logical problems using computing Machine.
- Map algorithmic solutions to relevant features of C programming language constructs.
- Gain knowledge about C constructs and its associated ecosystem.
- Appreciate and gain knowledge about the issues with C Standards and it's respective behaviours.

Course outcomes: At the end of the course, the student will be able to:

- Understand and Apply algorithmic solutions to counting problems using appropriate C Constructs.
- Understand, Analyze and Apply sorting and Searching techniques.
- Understand, Analyze and Apply text processing and string manipulation methods using Arrays, Pointers and functions.
- Understand user defined type creation and implement the same using C structures, unions and other ways by reading and storing the data in secondary systems which are portable.

Sindhu R Pai

Theory Anchor, Feb - May, 2025

Dept. of CSE,

PES University

Solve the below programs using C Structures and Functions.

Link for Solution: https://drive.google.com/file/d/12SDjjNqSTsvq0_bB-Fimwgya_1hrWURb/view?usp=drive_link

Level-1: Banana

1. In a robotics lab, **Engineers use autonomous drones to inspect large fields**. To visualize their positions, they use a grid system where each **drone's location is marked by an (x, y) coordinate.** A student intern is asked to write a simple C program that records a drone's position using a structure and displays it in coordinate form as shown.
   Sample Execution output:
   Input: x coordinate: 5
   y coordinate: 10
   Output: The Point is (5, 10)

2. In a smart college system, all students log in using biometric devices every day. These **devices record the current date and store them in a server.** The developers were asked to create a small module to accept the date from the user and display it in a standard format. This would later be used in advanced features like attendance tracking, calendar integration, or reminders.
   Example:
   Input: Date: Day (1-31): 6
   Month (1-12): 4
   Year: 2025
   Expected output: Date: Year 2025, Month 4, Day 6

3. In the world of digital education, Universities are moving toward smart grading systems where students' performance is tracked and analyzed using software. One such university wants to develop a module where a **student's marks in three subjects are entered, and the system calculates the average marks.** A team of computer science students is assigned to build this feature using C Structures. The team decided to group the student's name and marks, and a function to calculate the average.
   Sample outputs:
   Input: Student name: Riya
   Marks in 3 subjects: 85 90 88
   Expected output: Student: Riya
   Average Marks: 87.67

4. In a smart village, farmers use **cylindrical water tanks** to store water for their fields. A small device installed on the tank **measures the radius and height, and a program calculates how much water the tank holds.** You're asked to write this program to help farmers monitor water levels using math and programming.
   (Volume of the cylinder)$V = \pi.r^2.h$
   Example #1:
   Input: 3.0(radius) 10.0(height)
   Expected output: Volume of the cylinder: 282.74 cubic units

5. In a smart factory, a robotic arm moves from a base to a height, forming a right-angled triangle. To control its maximum reach without collisions, engineers need to **calculate the diagonal length (hypotenuse) using the Pythagorean Theorem**. You are asked to write a program using structures and functions to help with this.
   Example #1:
   Input: base: 3 height: 4
   Expected output: Diagonal reach = 5.00 units

6. In a modern student management system, **a faculty can update a student's grade(integer) using a digital interface**. Faculty should not be creating a copy of student's record every time a change is made and grade must be updated in original location only directly. This efficient approach makes the code cleaner and faster by modifying the original data in memory.
   Example #1:
   Input: Student name: Alice
   Current grade: 80
   Increase in grade:5
   Expected output: Updated Student grade: 85

7. In a unique school evaluation system, a student must pass any two-subject combination with **at least 70 marks.** If all such combinations pass the threshold, the student is declared **"Pass"**, otherwise, **"Fail".** The three subject scores are neatly grouped in a structure to make processing clean and organized.
   Example #1:
   Input: Enter marks in Subject A, B and C: 40 35 50
   Expected output: Pass
   Example #2:
   Input: Enter marks in Subject A, B and C: 20 35 15
   Expected output: Fail

8. In a computer lab, **three powerful servers—Alpha, Beta, and Gamma**—are available for processing students' large projects. Each server has a different response time (in milliseconds). The college wants to **identify the fastest (smallest response time) among the three so that high-priority student workloads can be directed there**. As a budding CSE student, you're assigned the task to solve these using structures.
   Given,
   ```
   struct ServerTimes {
      int alpha;
      int beta;
      int gamma;
   };
   ```
   Example #1:
   Input:
   time of Server Alpha (ms): 120
   time of Server Beta (ms): 85
   time of Server Gamma (ms): 95
   Expected output:
   Fastest server responds in 85 ms.

9. Guru is building a simple map-based feature for his college campus app that helps students measure the **straight-line distance between any two places** (like the library and canteen). **He decides to write a C program that takes two points (coordinates) from the user and calculates the distance using the standard Euclidean distance formula.**
   Example #1:
   Input:
   x and y for first point: 6 3
   x and y for second point: 2 2
   Expected output:
   The distance between the points is: 4.12

10. At a school health camp, students had to submit their **name, age, height, and blood group** on different papers. It became messy and hard to manage. So, the teacher suggested combining all the details into **one ID card**. A student smiled and said, "That's like a struct in C — we can **group different types of data together in one place!**" Everyone realized how useful it was to organize information that way — both in real life and in programming. Help the student to arrive at programming solution to this.
    Sample Execution:
    Input:
    Enter name: Riya
    Enter age: 14
    Enter height (in cm): 150.5
    Enter blood group: B+
    Expected output:
    --- Student ID Card ---
    Name : Riya
    Age : 14
    Height : 150.50 cm
    Blood Group: B+

Level-2:Orange

11. In a digital student management system, each student's record not only includes personal details such **as name and roll number but also their address(street, city and zip code) details**. Instead of creating separate variables for address details, a nested structure is used. The **nested structure makes the code more organized and modular by grouping related information together**. Your task is to create a C program that uses nested structures to input and display a student's record in a nice manner.
    Sample Execution:
    Input:
    student's name: Alice
    student's roll number: 101
    address: 123 Maple Street
    city: Springfield
    zip code: 62704
    Expected output: Name: Alice, Roll Number: 101
    Address: 123 Maple Street, Springfield – 62704

12. John works at a candy store with 100 chocolates in stock. His daily target is to sell X chocolates. For each chocolate sold up to the target, he earns 1 rupee. If he sells more than X, he earns 2 rupees for every extra chocolate. Use structures and functions to calculate how much John earns in a day by taking all the details of his day like target and sold.
Example #1: John daily goal was 3. Since he sold only 1 chocolate, he'll get only 1 rupee.
Input:
Enter daily target and chocolates sold: 3 1
Expected output:
John earned: rupees 1
Example #2: John daily goal - 4. Since he sold 7 chocolates. He will get 4 rupees for the 4 chocolates as his daily goal and 2 rupees per chocolate for the extra 3 chocolates. The total amount he gets is $4+3\cdot2=10$.
Input:
Enter daily target and chocolates sold: 4 7
Expected output:
John earned: rupees 10

13. Arjun wanted to buy chocolates for his sister. He had **some 5-rupee and 10-rupee coins** in his pocket. **Each chocolate cost a fixed amount.** To know how many he could buy, he calculated the total money and divided it by the price of one chocolate. With a quick calculation — like using a struct in C — he **found the exact number of chocolates he could gift her,** making her birthday special
Sample Execution: Arjun has $3\cdot5+1\cdot10=25$rupees in total. Since each chocolate costs 8 rupees, Arjun can buy a maximum of 3 chocolates , leaving him with 1 rupee.
Input:
Enter number of 5-rupee coins, 10-rupee coins, and chocolate cost: 3 1 8
Expected output:
Maximum chocolates Arjun can buy: 3

14. In a puzzle class, the teacher gave students two numbers, a and c, and asked them to **guess a missing number b such that a, b, and c form an arithmetic sequence.** But there's a twist - the teacher said
    - Only if both a and c are either even or odd, then the missing number b will also be an integer, and they can calculate it using the formula: b = (a + c) / 2
    - If a and c are of different parity (one odd, one even), then it's impossible to have such an integer b, and result to be -1.

    A structure is used to group a and c, helping them pass it cleanly to a function — just like how organized thinking helps in real-life puzzle solving!
    Sample Execution #1:
    Input: 3 5
    Expected output: 4
    Explanation:
    Considering B=4, the three integers are in arithmetic progression as $B-A=C-B=1$
    Sample Execution #2:
    Input: 4 7
    Expected output: -1

15. In a smart building, elevators only work if the **difference between two button codes, A and B, meets a rule: the difference modulo 3 should be 0 or 1**. This helps ensure that commands from the floor panel and elevator panel are nearly synchronized. If the condition is met, the elevator says "YES" and opens; otherwise, it says "NO". Engineers use a Pair structure to handle each test case cleanly.

```
typedef struct {
   int A;
   int B;
} Pair;
```

Example #1:
Input: 3 6
Expected output: YES
Example #2:
Input: 4 9
Expected output: NO

Level-3: Jackfruit

16. In a remote village, **two farms rely on a shared irrigation system controlled by a smart water flow controller.** Due to varying land slopes and pipe setups, each farm has pipes of **different lengths and resistances.** When one farm's water tank starts to overflow while the other runs low, the smart system must quickly decide the fastest way to transfer water between the two. **Using real-time pipe data(resistance(float) and length(float)), the controller calculates how long it will take to balance the water level from either side**. By simulating this logic with structures and a simple formula, engineering students can help automate water conservation in agriculture — a crucial need in modern farming. **Flow rate Contant is 10.**

$$\text{Time} = \frac{\text{Resistance} \times \text{Length}}{\text{Flow Rate Constant}}$$

Sample Execution:
Input:
Enter resistance and length of pipe for Farm 1: 2.5 100
Enter resistance and length of pipe for Farm 2: 3.0 80
Expected Output:
Time to balance from Farm 1: 25.00 seconds
Time to balance from Farm 2: 24.00 seconds
Start flow from Farm 2 for quicker balance.

17. In a bustling courier center of a smart city, a company has developed an intelligent parcel locker system to streamline deliveries. Couriers arriving at the hub are often assigned multiple packages for delivery. One day, **a delivery executive named Arjun scans two parcels into the system before heading out. Each parcel has specific details: a unique parcel ID, the weight of the package, the destination code, and the estimated delivery time in hours**. However, due to an unexpected power outage at the sorting station, the system couldn't optimize the delivery order automatically. Arjun, determined to stay on schedule, manually enters the parcel details into the system again.

The smart system, now rebooted, helps Arjun make key decisions. It checks which of the two parcels should be delivered first based on which one needs to reach its destination sooner. Not only that, the **system also identifies if both parcels are headed to the same destination. If so, it combines their weights and suggests handling them as a single delivery to save time and resources. If not, it displays both parcel details clearly so Arjun can plan two separate stops efficiently. This small but intelligent use of technology made Arjun's job easier and deliveries smoother**—even on a day when everything could have gone wrong.

Example#1:

Input:

Enter details for Parcel 1:
Parcel ID: P101
Weight: 2.5
Destination Code: 1001
Delivery Time: 5
Enter details for Parcel 2:
Parcel ID: P102
Weight: 3.0
Destination Code: 1001
Delivery Time: 3

Expected Output:

Deliver Parcel P102 first.
Merging parcels going to the same destination:
Combined Parcel to Destination Code 1001
Total Weight: 5.50 kg

18. In a high-tech city mall, a smart parking system is designed to optimize space usage in a limited underground parking area. Each vehicle entering the premises is scanned by sensors that detect its **length in meters and category—whether it's a compact car, SUV, or a bike.** The parking system receives the data and, based on the vehicle's dimensions, dynamically decides whether there's enough space left in a designated zone.
One evening, two friends, Anya and Ravi, arrive at the mall in different vehicles. Anya drives a bike, and Ravi brings his mid-size SUV. The parking terminal asks each of them to input their vehicle type and length. The system, using this data, checks if there is sufficient space left (say 5 meters). **If both vehicles can be parked within the available space, it displays "Parking Confirmed." If not, it shows how much space remains and whether any adjustments can be made** (e.g., allowing just one of them to park).
Hint: Structures are used to hold vehicle info, and decision logic is encapsulated in a function.

Example#1:

Input:

Enter vehicle 1 type (B: Bike, C: Car, S: SUV): C
Enter vehicle 1 length in meters: 2.5
Enter vehicle 2 type (B: Bike, C: Car, S: SUV): B
Enter vehicle 2 length in meters: 1.2
Expected Output: Parking Confirmed for both vehicles
Enter vehicle 1 type (B: Bike, C: Car, S: SUV): B
Enter vehicle 1 length in meters: 6
Enter vehicle 2 type (B: Bike, C: Car, S: SUV): C

Enter vehicle 2 length in meters: 8
Not enough space! Required: 14.00 meters, Available: 5.00 meters.

19. Ayaan and Bella ran two delivery trucks that carried sacks of goods—**Ayaan's truck carried sacks weighing a kg(whole numbers only), and Bella's truck carried b kg(whole numbers only).** They wanted to load both trucks in a way that the total weight was evenly divisible, so nothing was wasted. Their tech-savvy cousin suggested **calculating the GCD to find the largest sack size they could share, and the LCM to know when their deliveries would perfectly align**. With a simple program using structures and functions, they could now plan joint deliveries with ease—saving time, money, and space!
Sample Execution:
Input: 12 18
Expected Output:
GCD: 6
LCM: 36

20. It's the final night of the Hackathon at the University. Students are working all night on their innovative ideas. Everyone is tired, drained, and their laptops are begging for juice — the battery symbol is red, the screen's dimming, and their code... almost compiled!
Twist is: There's only ONE portable debugging booster (basically a laptop power bank nicknamed "Debugger Daddy") left in the lab! The booster can charge one laptop at a time. The lab mentor, known for his logical decisions, makes an announcement: **"Only the coder whose laptop has the lowest battery should get the booster first — we must prevent shutdown of critical projects!"**
You're assigned to write a program to help the mentor decide which student should get the Debugger Daddy first. Each student walks up and enters: **name, laptop battery percentage, and estimated time needed for debugging (in minutes)**
Your task is to take the data of n students (number of entries decided at runtime), and decide who is the most in need of the booster — the one with the lowest battery. You should not use arrays of structures. Just one structure at a time! Why? Because just like sharing a single debugging device, we reuse one structure at a time for each student.
Example#1:
Input:
Number of students: 3
Enter name of student: Sneha
Enter battery percentage of Sneha: 25
Enter debugging time (minutes) of Sneha: 40
Enter name of student: Manish
Enter battery percentage of Manish: 17
Enter debugging time (minutes) of Manish: 35
Enter name of student: Arjun
Enter battery percentage of Arjun: 38
Enter debugging time (minutes) of Arjun: 50
Expected Output:
Manish will get the Debugger Daddy first! (Battery: 17%, Needs: 35 minutes)

Happy Coding using Structures and Functions in C!!