LAB 15: Structures

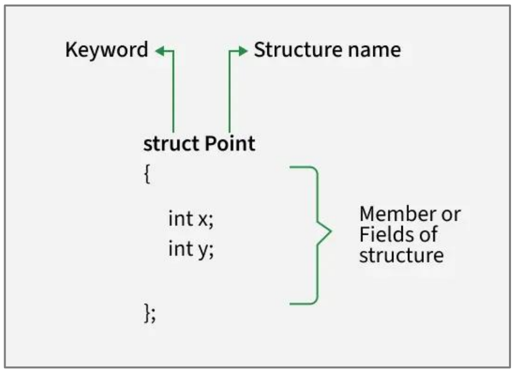
CLOs: CLO-4

Introduction

You learned how to group values of the same type by using arrays. You will learn how to group related values that are of different types. An array is a homogeneous data structure; a struct is typically a heterogeneous data structure.

struct: A collection of a fixed number of components in which the components are accessed by name. The components may be of different types. It is similar to a class in that both hold a collection of data of different data types.

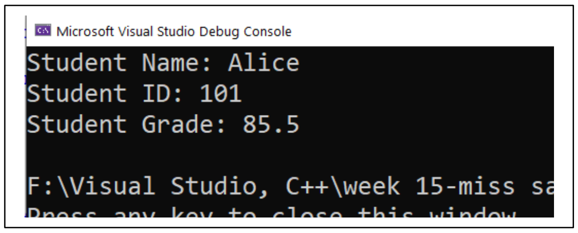
```
struct structName
{
      dataType1 identifier1;
      dataType2 identifier2;
          .
          .
          .
      dataTypen identifiern;
};
```



In C++, structures (structs) allow you to create custom data types by grouping related variables together. Unlike built-in types, structs can hold mixed data (e.g., int, float, char arrays). This lab focuses on reimplementing common operations on structs from scratch using basic logic. We'll use fixed-size arrays for any collections and manual logic for operations.
All examples will use simple structs with fixed-size char arrays for strings (e.g., char name[50]).
Remember: **Structs are passed by value** to functions (copies made), so be mindful of efficiency with large structs.
Key Rules for All Implementations:

- Use struct keyword to define.
- Access members with the dot (.) operator.
- Handle arrays inside structs manually (e.g., loops for copying).
- Test for edge cases: uninitialized structs, zero values, maximum array sizes.
- Functions should take structs by value or use arrays of structs

```
#include <iostream>
#include <string>
using namespace std;

struct Student {
    string name;
    int id;
    float grade;
};

int main() {
    Student s1;

    s1.name = "Alice";
    s1.id = 101;
    s1.grade = 85.5f;

    cout << "Student Name: " << s1.name << endl;
    cout << "Student ID: " << s1.id << endl;
    cout << "Student Grade: " << s1.grade << endl;

    return 0;
}
```



```
Student Name: Alice
Student ID: 101
Student Grade: 85.5

F:\Visual Studio, C++\week 15-miss sa
Press any key to close this window
```

Exercise 1

- Modify the code to create two students and print their details.
- Add a new member (e.g., int age) to the structure and update the code accordingly.

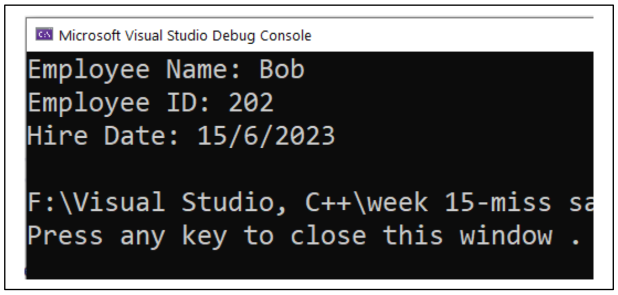
Nested Structures

Nested structures occur when one structure is a member of another structure. This allows hierarchical data representation (e.g., a date inside an employee record).

Key Concepts

- Declare the inner structure first or inside the outer one.
- Access nested members using multiple dot operators (e.g., outer.inner.member).

```
#include <iostream>
#include <string>
using namespace std;

struct Date {
    int day;
    int month;
    int year;
};

struct Employee {
    string name;
    int id;
    Date hireDate;  // Nested structure
};

int main() {
    Employee e1;

    e1.name = "Bob";
    e1.id = 202;
    e1.hireDate.day = 15;
    e1.hireDate.month = 6;
    e1.hireDate.year = 2023;

    cout << "Employee Name: " << e1.name << endl;
    cout << "Employee ID: " << e1.id << endl;
    cout << "Hire Date: " << e1.hireDate.day << "/"
        << e1.hireDate.month << "/" << e1.hireDate.year << endl;

    return 0;
}
```



```
Employee Name: Bob
Employee ID: 202
Hire Date: 15/6/2023

F:\Visual Studio, C++\week 15-miss sa
Press any key to close this window .
```

Exercise 2

- Add another nested structure (e.g., struct Address { string city; string country; };) to Employee.
- Create and print an employee with address details.

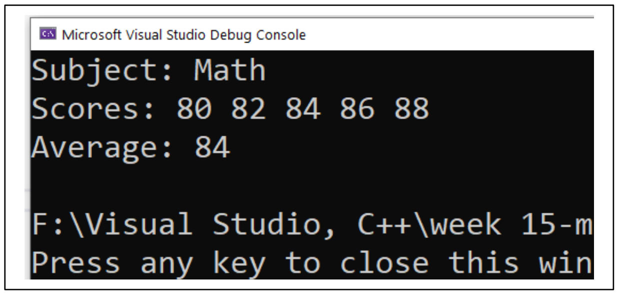
Structures with Arrays

Structures can contain arrays as members, allowing fixed-size collections within the structure (e.g., an array of scores in a student record).

Key Concepts

- Arrays in structures behave like regular arrays.
- Initialize and access array elements using indices.

```
#include <iostream>
#include <string>
using namespace std;

struct TestScores {
    string subject;
    int scores[5];  // Array of 5 scores
    float average;
};

int main() {
    TestScores ts;
    float sum = 0.0f;

    ts.subject = "Math";
    for (int i = 0; i < 5; ++i) {
        ts.scores[i] = 80 + i * 2;
        sum += ts.scores[i];
    }
    ts.average = sum / 5;

    cout << "Subject: " << ts.subject << endl;
    cout << "Scores: ";
    for (int i = 0; i < 5; ++i) {
        cout << ts.scores[i] << " ";
    }
    cout << endl << "Average: " << ts.average << endl;

    return 0;
}
```


Microsoft Visual Studio Debug Console

```
Subject: Math
Scores: 80 82 84 86 88
Average: 84

F:\Visual Studio, C++\week 15-m
Press any key to close this win
```

Exercise 3

- Write a function that takes a struct TestScores array by value and calculates the average of all the scores, and displays it using the function.

Structures and Functions

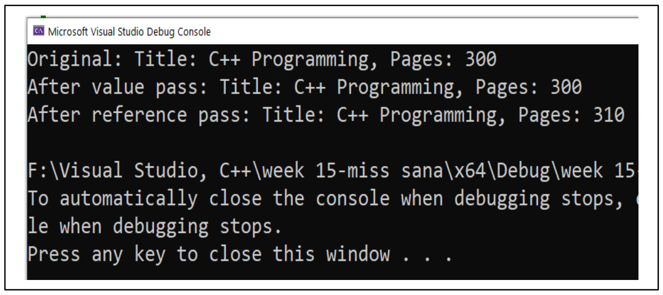
Structures can be passed to functions like other variables. We'll cover passing by value (copy) and by reference (pointer, for efficiency and modification).

Key Concepts

- **Pass by Value**: A copy is passed; changes don't affect the original.
- **Pass by Reference**: Use pointers (* and &); changes affect the original. (Note: C doesn't have true pass-by-reference like C++, but pointers simulate it.)
- Functions can take structures as parameters.

```
#include <iostream>
#include <string>
using namespace std;

struct Book {
    string title;
    int pages;
};
```

```cpp
// Function: Pass by Value
void printBookByValue(Book b) {
    cout << "Title: " << b.title << ", Pages: " << b.pages << endl;
    b.pages += 10;
}

// Function: Pass by Reference
void updateBookByReference(Book& b) {
    b.pages += 10;
}

int main() {
    Book bk;
    bk.title = "C++ Programming";
    bk.pages = 300;

    cout << "Original: ";
    printBookByValue(bk);  // Pass by value
    cout << "After value pass: Title: " << bk.title << ", Pages: " << bk.pages << endl;

    updateBookByReference(bk);  // Pass by reference
    cout << "After reference pass: Title: " << bk.title << ", Pages: " << bk.pages << endl;
    return 0;
}
```


Microsoft Visual Studio Debug Console

```
Original: Title: C++ Programming, Pages: 300
After value pass: Title: C++ Programming, Pages: 300
After reference pass: Title: C++ Programming, Pages: 310

F:\Visual Studio, C++\week 15-miss sana\x64\Debug\week 15
To automatically close the console when debugging stops,
le when debugging stops.
Press any key to close this window . . .
```

Exercise 4

- Create a structure for a book with an array of chapter titles (e.g., string chapters[10];).
- Initialize and print 3 chapter titles.

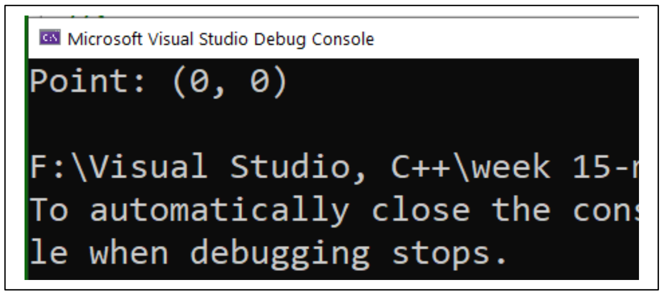
Returning Structures from Functions

Functions can return structures, allowing creation or modification in one place and use elsewhere.

Key Concepts

- Return type is the structure type.
- Useful for factory-like functions that create initialized structures.

```cpp
#include <iostream>
using namespace std;

struct Point {
    int x;
    int y;
};

// Function that returns a structure
Point createPoint(int x, int y) {
    Point p;
    p.x = x;
    p.y = y;
    return p;
}

int main() {
    Point origin = createPoint(0, 0);
    cout << "Point: (" << origin.x << ", " << origin.y << ")" << endl;
    return 0;
}
```


Microsoft Visual Studio Debug Console

```
Point: (0, 0)

F:\Visual Studio, C++\week 15-
To automatically close the cons
le when debugging stops.
```

Exercise 5

- Write a function that returns a struct Employee with initialized values.

- Combine with nested structures: Return an employee with a hire date.

Copying Structs (copyPerson)

Copy one Person struct to another, handling char array manually.

Prototype: void copyPerson(Person& dest, Person src); (Note: Use reference to avoid copy overhead, but no pointers.)

Logic:

- Assign scalar members directly: dest.age = src.age;
- Loop to copy name array char by char until '\0', then add '\0'.

Expected Behavior:

- src: "Bob", 30, 6.0 → dest becomes identical after copy.

```
void copyPerson(Person& dest, Person src) {
    dest.age = src.age;
    dest.height = src.height;
    int i = 0;
    while (src.name[i] != '\0') {
        dest.name[i] = src.name[i];
        i++;
    }
    dest.name[i] = '\0';
}
```

Tasks

Task 1: Student Record System: Make a structure named Student that has the members of name, rollNo, and cgpa. Your task is to take input for **N students**, store them in an array of structures, and then:

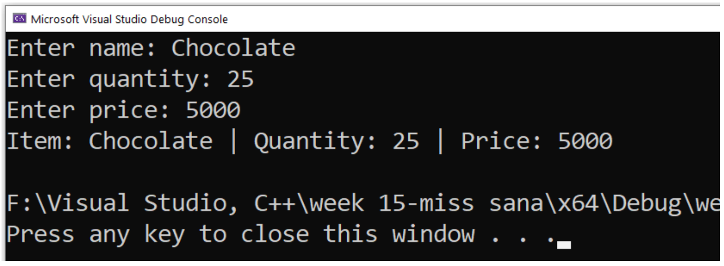
- Display student with **highest CGPA**
- Display student with **lowest CGPA**

Task 2: Library Book Management: Make a structure named Book that has the members of id, title, author, and price. Your task is to input details of 5 books and then:

- Search for a book by **ID**
- Print all books with **price > 1000**

Task 3: Simple Structure for Inventory Item

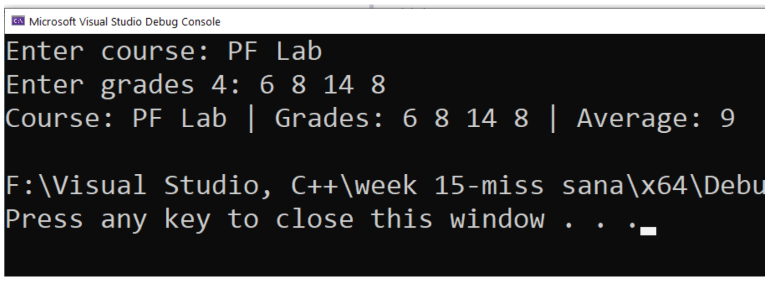
Create a struct Item with members: string name, int quantity, float price. Write a program to input details for one item from the user and display them in a formatted table. Add validation to ensure quantity and price are positive.



```
Enter name: Chocolate
Enter quantity: 25
Enter price: 5000
Item: Chocolate | Quantity: 25 | Price: 5000

F:\Visual Studio, C++\week 15-miss sana\x64\Debug\we
Press any key to close this window . . .
```

Task 4: Structure with Array for Exam Grades

Create struct Course with string name, int grades[4], and float average. Input the course name and 4 grades, compute the average, and display all details. Handle invalid grades (0-100 range).

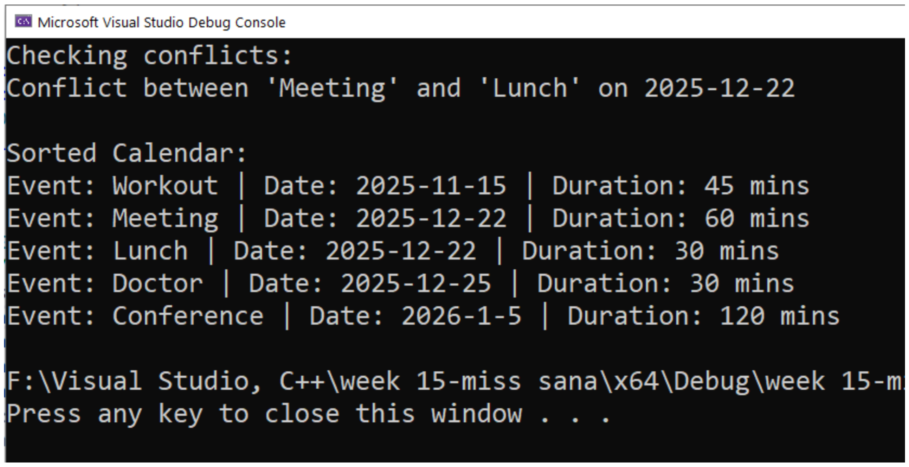


```
Enter course: PF Lab
Enter grades 4: 6 8 14 8
Course: PF Lab | Grades: 6 8 14 8 | Average: 9

F:\Visual Studio, C++\week 15-miss sana\x64\Debu
Press any key to close this window . . .
```

Task 5: **Event Scheduling Calendar:** Schedule events with title, date (nested: year, month, day), duration (int mins). Add events, check conflicts, sort by date.
Requirements:

- Structs: struct Date { int year; int month; int day; }; struct Event { char title[50]; Date date; int duration; };
- Function 1: void addEvent(Event calendar[], int& num_events, char title[], int y, int m, int d, int dur); – Init and add.
- Function 2: bool hasConflict(Event e1, Event e2); – Compare dates (assume same day conflict if overlap, but simple: same date).
- Function 3: void sortByDate(Event calendar[], int num_events); – Bubble sort by year>month>day.
- Main: Add 5 events, check conflicts between pairs, sort, print calendar.

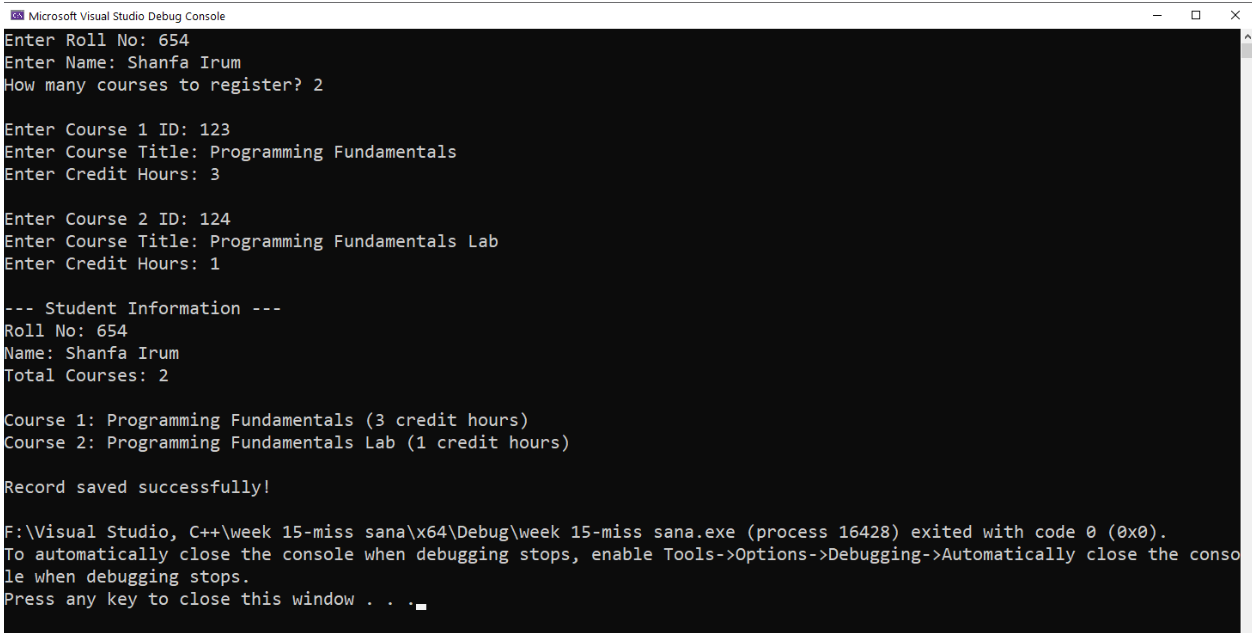
addEvent(calendar, num_events, "Meeting", 2025, 12, 22, 60);
addEvent(calendar, num_events, "Lunch", 2025, 12, 22, 30);
addEvent(calendar, num_events, "Conference", 2026, 1, 5, 120);
addEvent(calendar, num_events, "Workout", 2025, 11, 15, 45);
addEvent(calendar, num_events, "Doctor", 2025, 12, 25, 30);



```
Checking conflicts:
Conflict between 'Meeting' and 'Lunch' on 2025-12-22

Sorted Calendar:
Event: Workout | Date: 2025-11-15 | Duration: 45 mins
Event: Meeting | Date: 2025-12-22 | Duration: 60 mins
Event: Lunch | Date: 2025-12-22 | Duration: 30 mins
Event: Doctor | Date: 2025-12-25 | Duration: 30 mins
Event: Conference | Date: 2026-1-5 | Duration: 120 mins

F:\Visual Studio, C++\week 15-miss sana\x64\Debug\week 15-m
Press any key to close this window . . .
```

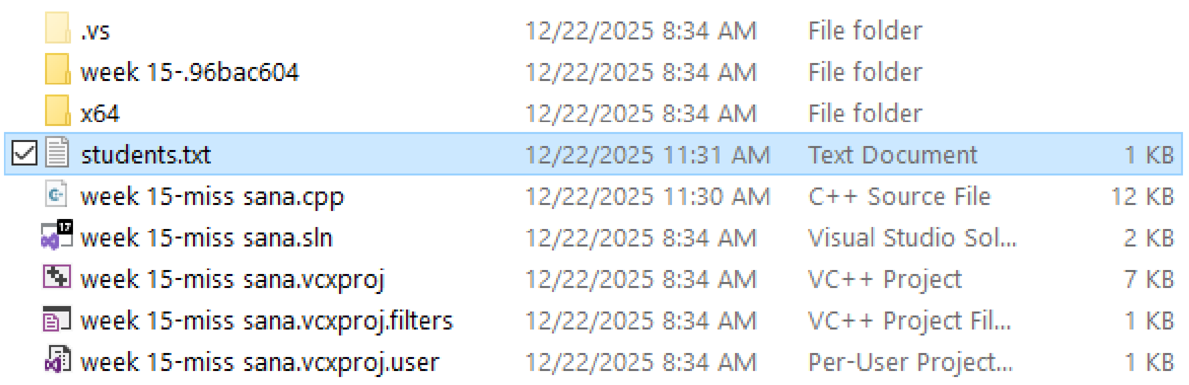
Task 6: The university wants a simplified system where each **Student** has roll number, Name, and up to 5 registered courses. Each **Course** has Course ID, Course Title, and Credit Hours. The program should:

1. Let the user create a **Course structure**.
2. Add a **Student structure** that contains an **array of Course structures** (nested structure).
3. Use:
    - Student addStudent() → returns structure
    - void registerCourse(Student&, Course) → pass-by-reference
    - void displayStudent(Student) → pass-by-value
4. Write student record to file students.txt
5. Read back all students and display them.



Microsoft Visual Studio Debug Console

```
Enter Roll No: 654
Enter Name: Shanfa Irum
How many courses to register? 2

Enter Course 1 ID: 123
Enter Course Title: Programming Fundamentals
Enter Credit Hours: 3

Enter Course 2 ID: 124
Enter Course Title: Programming Fundamentals Lab
Enter Credit Hours: 1

--- Student Information ---
Roll No: 654
Name: Shanfa Irum
Total Courses: 2

Course 1: Programming Fundamentals (3 credit hours)
Course 2: Programming Fundamentals Lab (1 credit hours)

Record saved successfully!

F:\Visual Studio, C++\week 15-miss sana\x64\Debug\week 15-miss sana.exe (process 16428) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```



.vs	12/22/2025 8:34 AM	File folder	
week 15-.96bac604	12/22/2025 8:34 AM	File folder	
x64	12/22/2025 8:34 AM	File folder	
students.txt	12/22/2025 11:31 AM	Text Document	1 KB
week 15-miss sana.cpp	12/22/2025 11:30 AM	C++ Source File	12 KB
week 15-miss sana.sln	12/22/2025 8:34 AM	Visual Studio Sol...	2 KB
week 15-miss sana.vcxproj	12/22/2025 8:34 AM	VC++ Project	7 KB
week 15-miss sana.vcxproj.filters	12/22/2025 8:34 AM	VC++ Project Fil...	1 KB
week 15-miss sana.vcxproj.user	12/22/2025 8:34 AM	Per-User Project...	1 KB