PROGRAMMING FUNDAMENTALS

WEEK 14: STRUCTURES AND UNIONS IN C

LECTURE GOALS

- Understand what structures and unions are.
- Learn to declare and use structures.
- Compare structures vs unions (memory usage).
- Practice with real-world examples.
- Understand memory layout differences.

WHAT IS A STRUCTURE?

Core Idea: A structure is a user-defined data type that groups different types of data together.

Why do we need structures?

- Store related information in one place
- Instead of separate variables: name, age, gpa
- Group them: student.name, student.age, student.gpa

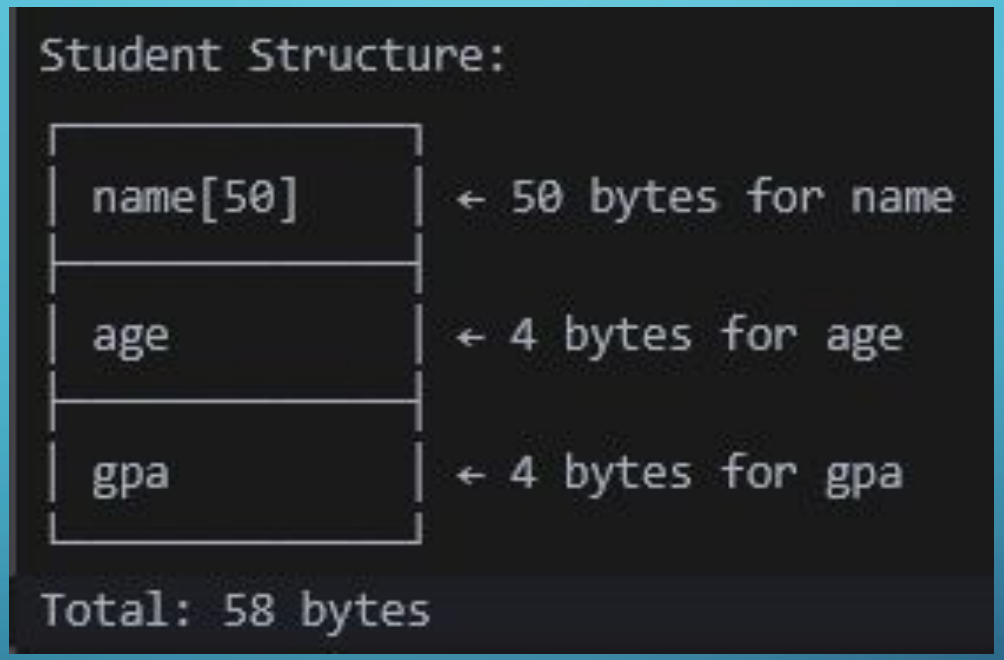
WHAT IS A STRUCTURE?

Example:

```
struct Student {
    char name[50];
    int age;
    float gpa;
```

WHAT IS A STRUCTURE?

Visual:



💡 Think: A student record card that holds all information in one place.

DECLARING AND USING STRUCTURES

Declaration:

Example 1 Code:

```
struct Student {
    char name[50];
    int age;
    float gpa;
};
```

DECLARING AND USING STRUCTURES

Example 1:

```
int main() {
    struct Student s1;  // Declare a variable of type
Student
    // Assign values
    strcpy(s1.name, "Ali");
    s1.age = 20;
    s1.gpa = 3.75;
```

DECLARING AND USING STRUCTURES

Example 1:

```
// Access values
    printf("Name: %s\n", s1.name);
    printf("Age: %d\n", s1.age);
    printf("GPA: %.2f\n", s1.gpa);
    return 0;

}
```

DECLARING AND USING STRUCTURES

Alternative: Using typedef for easier typing:

Example 1 part 2:

```
typedef struct {
    char name[50];
    int age;
    float gpa;
} Student;

Student s1;  // Now we can use just "Student"
```

ARRAY OF STRUCTURES

Purpose: Store multiple records of the same type (like multiple students).

Example 2:

```
#include <stdio.h>
#include <string.h>
typedef struct {
    char name[50];
    int age;
    float gpa;
} Student;
int main() {
    Student students[3];  // Array of 3 students
```

ARRAY OF STRUCTURES

Example 2:

```
// Input data
    strcpy(students[0].name, "Ali");
    students[0].age = 20;
    students[0].gpa = 3.75;
    strcpy(students[1].name, "Sara");
    students[1].age = 19;
    students[1].gpa = 3.90;
```

ARRAY OF STRUCTURES

Example 2:

```
 // Print all students
    for (int i = 0; i < 2; i++) {
        printf("Student %d: %s, Age: %d, GPA: %.2f\n",
               i+1, students[i].name, students[i].age,
students[i].gpa);
    }
    return 0;
}
```

ARRAY OF STRUCTURES

Example 2:

Memory Layout:

```
students[0]: [name][age][gpa] ← 58 bytes
students[1]: [name][age][gpa] ← 58 bytes
students[2]: [name][age][gpa] ← 58 bytes

Total: 174 bytes
```

NESTED STRUCTURES

Purpose: Structure inside another structure (like address inside student).

Example 3:

```
typedef struct {
    char street[50];
    char city[30];
    int zip;
} Address;

typedef struct {
    char name[50];
    int age;
    Address addr;  // Nested structure
} Person;
```

NESTED STRUCTURES

Purpose: Structure inside another structure (like address inside student).

Example 3:

```
int main() {
    Person p1;
    strcpy(p1.name, "Ali");
    p1.age = 25;
    strcpy(p1.addr.street, "Main St");
    strcpy(p1.addr.city, "Karachi");
    p1.addr.zip = 75600;
    printf("Name: %s\n", p1.name);
    printf("City: %s\n", p1.addr.city);  // Access nested member
    return 0;
}
```

WHAT IS A UNION?

Core Idea: A union is like a structure, but all members share the same memory space.

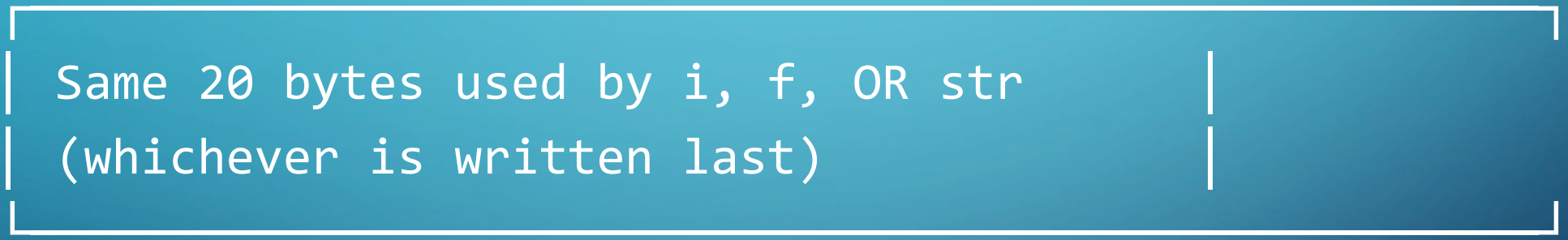
Syntax:

```
union Data {
    int i;
    float f;
    char str[20];
};
```

WHAT IS A UNION?

Memory Layout:

Union Data (size = 20 bytes – largest member):



WHAT IS A UNION?

Example 4:

```
union Data d;
d.i = 10;          // Uses first 4 bytes
printf("%d\n", d.i);  // Output: 10
d.f = 3.14;        // Overwrites same memory
printf("%.2f\n", d.f);  // Output: 3.14
strcpy(d.str, "Hello");  // Overwrites again
printf("%s\n", d.str);   // Output: Hello
printf("%d\n", d.i);     // Output: Garbage!
```

WHAT IS A UNION?

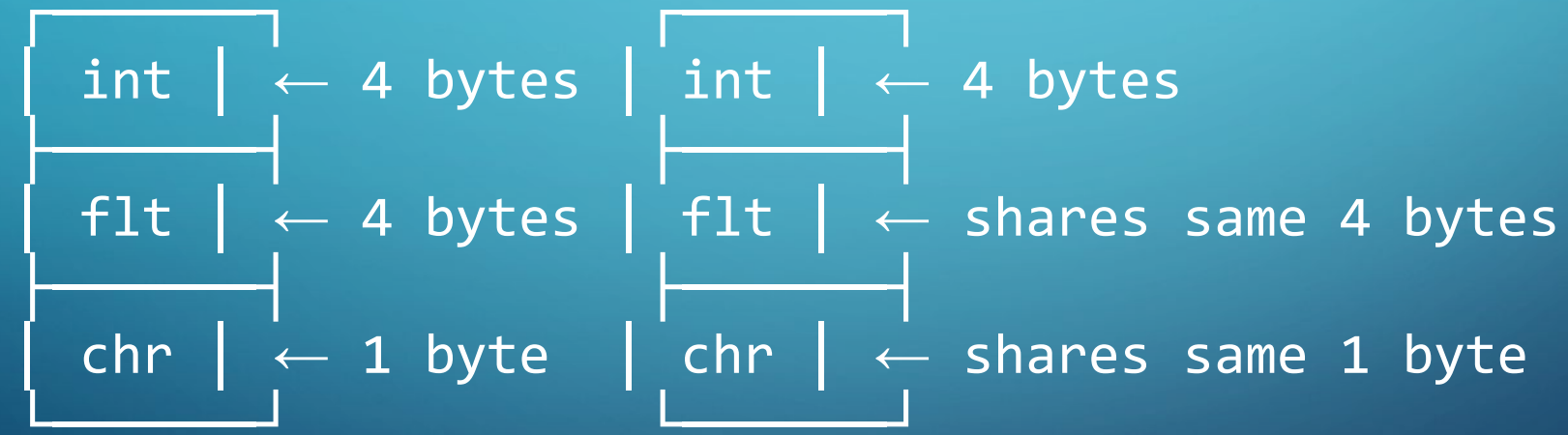
💡 Think: One room, multiple uses - like a classroom that's used by different classes.

STRUCTURE VS UNION COMPARISON

FEATURE	STRUCTURE	UNION
Memory	Each member gets separate memory	All members share same memory
Size	Sum of all member sizes	Size of largest number
Usage	Store different data simultaneously	Save memory, use one at a time
Access	All members can be used at once	Only last assigned member is valid

STRUCTURE VS UNION COMPARISON

Visual Representation:



PRACTICAL EXAMPLE - STUDENT DATABASE

Goal: Create a simple student management system using structures.

Example 5:

```
#include <stdio.h>
#include <string.h>
typedef struct {
    int id;
    char name[50];
    float gpa;
    char grade;
} Student;
```

PRACTICAL EXAMPLE - STUDENT DATABASE

Example 5:

```
int main() {
    Student students[100];// Can store up to 100 students
    int count = 0;
    int choice;
    while (1) {
        printf("\n1. Add Student\n2. Print All\n3. Exit\n");
        printf("Choice: ");
        scanf("%d", &choice);
```

PRACTICAL EXAMPLE - STUDENT DATABASE

Example 5:

```
 if (choice == 1) {
    printf("Enter ID: ");
    scanf("%d", &students[count].id);
    printf("Enter Name: ");
    scanf("%s", students[count].name);
    printf("Enter GPA: ");
    scanf("%f", &students[count].gpa);
    // Calculate grade
    if (students[count].gpa >= 3.5) students[count].grade = 'A’;
    else if (students[count].gpa >= 2.5) students[count].grade =
'B’;
    else students[count].grade = 'C’;
    count++;
}
```

PRACTICAL EXAMPLE - STUDENT DATABASE

Example 5:

```
  else if (choice == 2) {
   for (int i = 0; i < count; i++) {
     printf("ID: %d, Name: %s, GPA: %.2f, Grade: %c\n",
       students[i].id, students[i].name,
       students[i].gpa, students[i].grade);
   }
  } else if (choice == 3) break;
 }
 return 0;
}
```

ASSIGNMENT

1. Create a structure called Book with title, author, and price. Create one book variable, assign values, and print them.
2. Create an array of 3 Book structures, input data for each, and print all books.
3. Create a union with int, float, and char. Show that assigning to one member affects the others.
4. (Challenge) Create a structure called Employee with name, id, salary, and department. Create 5 employees and find the one with highest salary.

Instructions:
Submit as .c files