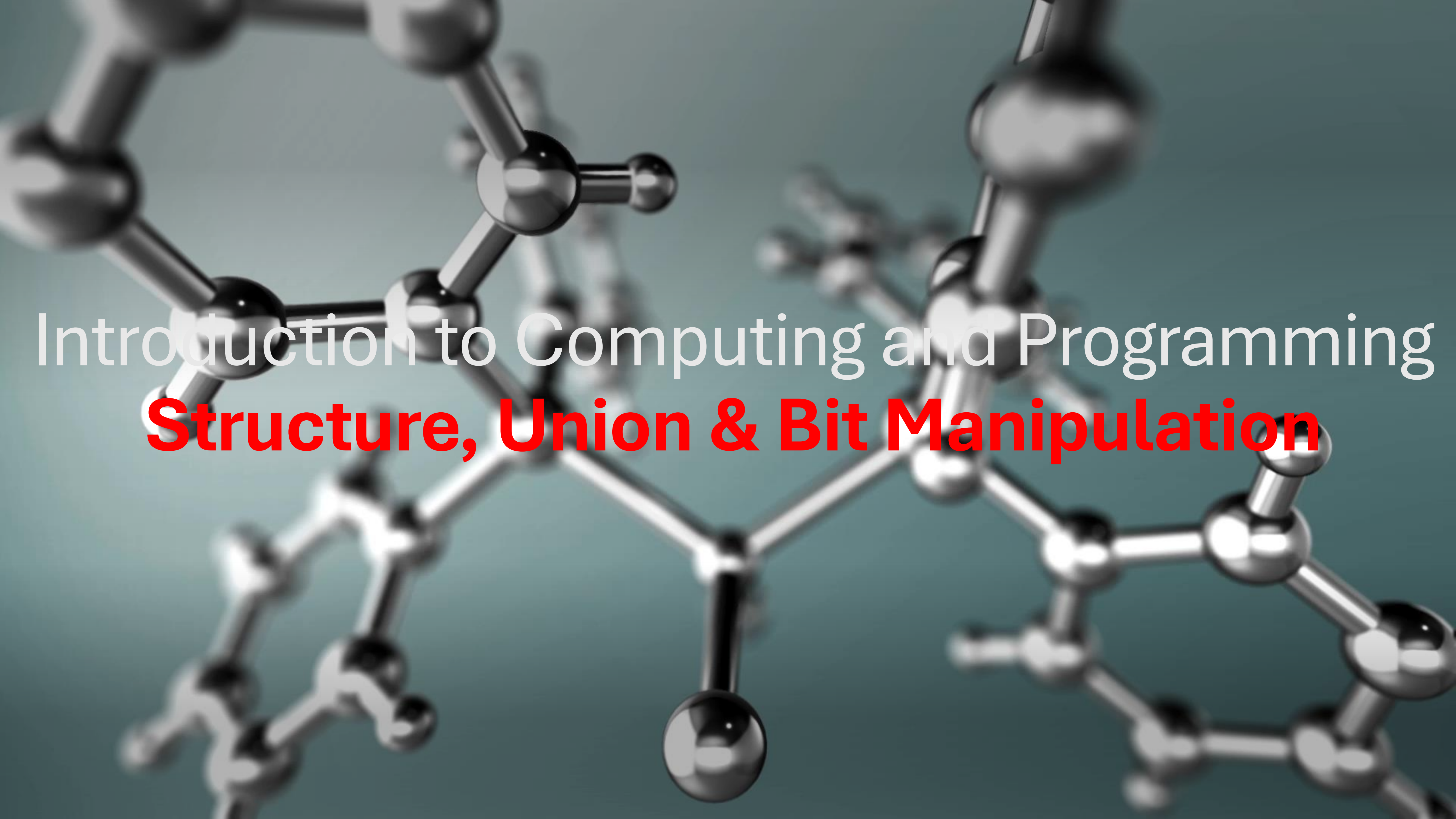


Introduction to Computing and Programming

Structure, Union & Bit Manipulation

Content

- Recap
- User-defined Vs Derived data Types
- Structure
- Union
- Bit Manipulation

Recap

- String Function
- String Pointers
- Array of Pointers to String

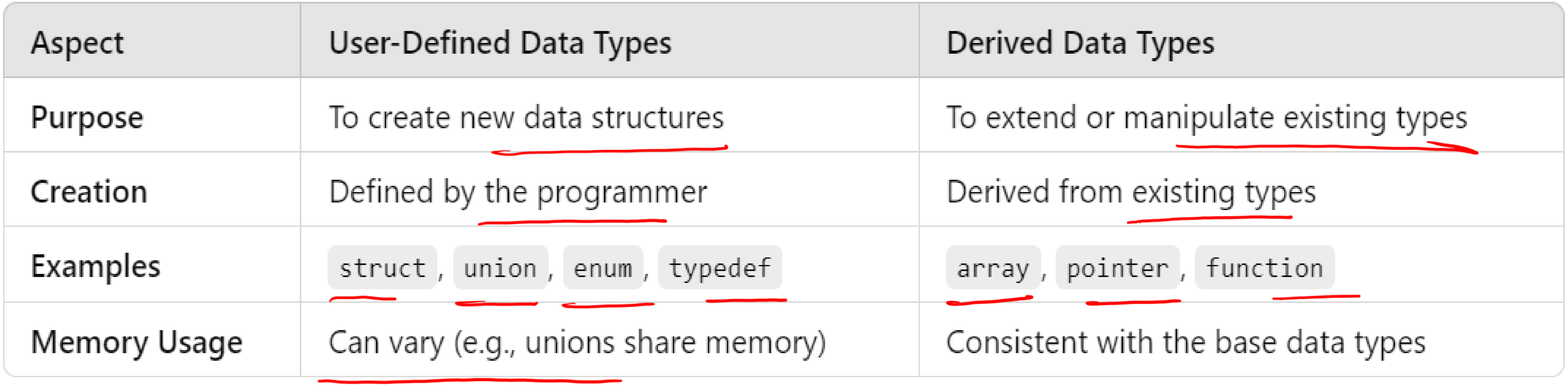
User-defined Vs Derived data Types

- **User-defined** and **Derived data types** both extend the basic data types, but they differ in how they are created and used.

- **User-defined data types** are created by the programmer using C's fundamental data types and allow custom structures tailored to specific needs. **Eg:** Structure, Union, etc.

- **Derived data types** are created from existing data types but don’t involve creating new types. They extend the behavior or structure of existing data types rather than defining entirely new types. **Eg:** arrays, pointers, functions, etc.

User-defined Vs Derived data Types Cont..



Aspect	User-Defined Data Types	Derived Data Types
Purpose	To create new data structures	To extend or manipulate existing types
Creation	Defined by the programmer	Derived from existing types
Examples	`struct`, `union`, `enum`, `typedef`	`array`, `pointer`, `function`
Memory Usage	Can vary (e.g., unions share memory)	Consistent with the base data types

Why Structures

- Arrays require that all elements be of the same data type. Many times it is necessary to group information of different data types **(Example: Student Details)**

- C and C++ support data structures that can **store combinations of character, integer floating point and enumerated type data** (explained later). They are called **structs.**

What is Structures?

- **Definition:** A structure is a **user-defined data type** that groups related variables of different data types under a single name.

- **Syntax:** struct structureName {

```
    dataType1 member1;
    dataType2 member2;
    ...
};
```

Student

float ICP_marks ;

Char Name[20] ;

Int Roll ;

- **Purpose:** Useful for **organizing complex data**, like representing a **student's information** with name, age, and grades.

Example of defining Structure in C

- **Example:** struct Student {

```
 char name[50];
 int age;
 float grade;
};
```

struct Student s1;

- **Explanation:** struct Student defines a structure named Student. s1 is a variable of this type with name, age, and grade fields.

Two ways to declare variables of a struct

1. We can declare variables of type struct **right at the place of declaration**:

```
struct Student {
   char name[50];
   int age;
   float grade;
}s1;  /* we declare a variable s1*/
```

2. We can declare variables of type struct in **another place** we need:

```
struct Student {
   char name[50];
   int age;
   float grade;
};
int main(void){
struct Student s1;
}
```

Accessing Structure Members

- **Accessing Members:** Use the dot (.) operator with structure variables.
- **Example:** s1.age = 20;

  printf("Age: %d", s1.age);

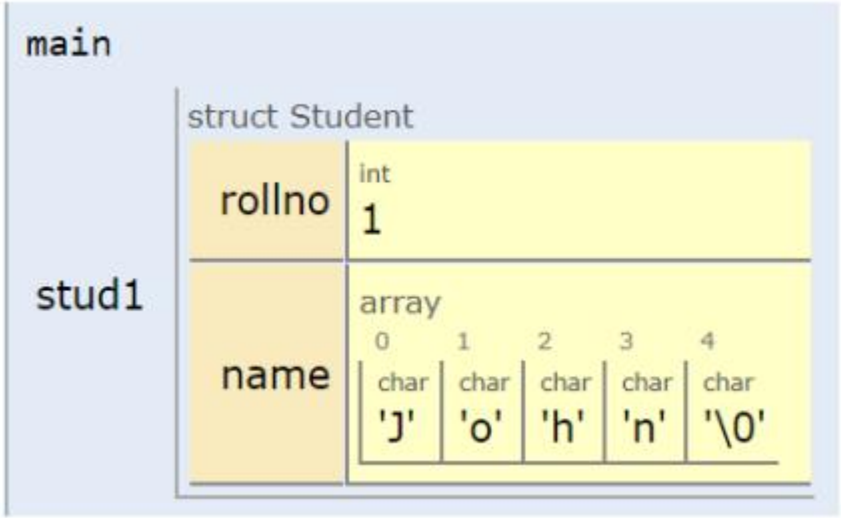
- **Explanation:** Assigns and prints the age of s1.

Example of C program using Struct

```
#include <stdio.h>
#include <string.h>
typedef struct Students{
            int rollno;
            char name[5];
            }Student;
int main() {
            Student stud1;
            stud1.rollno = 1;
            strcpy(stud1.name, "John");
            printf("Student rollno: %d\n", stud1.rollno);
            printf("Student name: %s\n", stud1.name);
return 0;
}
```

Output:

```
Student rollno: 1
Student name: John
```



Typedef

- The C language provides a facility called typedef for creating **synonyms** for previously defined data type names.
- For example: **typedef int Length;**
- The above makes the name **‘Length’ a synonym (or alias) for the data type int.**
- Now ‘Length’ be used in declarations in exactly the same way that the data type int can be used:

Length a, b, len ;

Length numbers[10] ;

Typedef & Struct

- Often, typedef is used in **combination with struct to declare a synonym (or an alias) for a structure**:

```
        typedef struct Students
        {
                int roll_no; /* assume integer roll no */
                char name[20]; } Student;
/* We can use Student like any other type*/
int main(void)
{ Student stud1; }
```

Array of Structures

- Just like any other type you can declare an array of struct

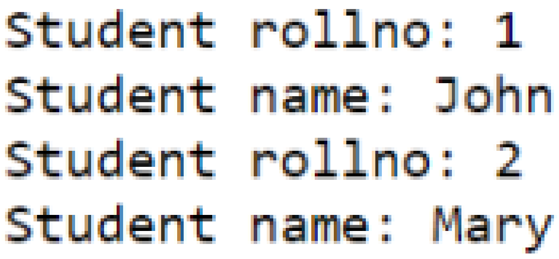
  Student stud[10];

- The above declares an array of size 10.
- As any other array, the storage in memory is contiguous

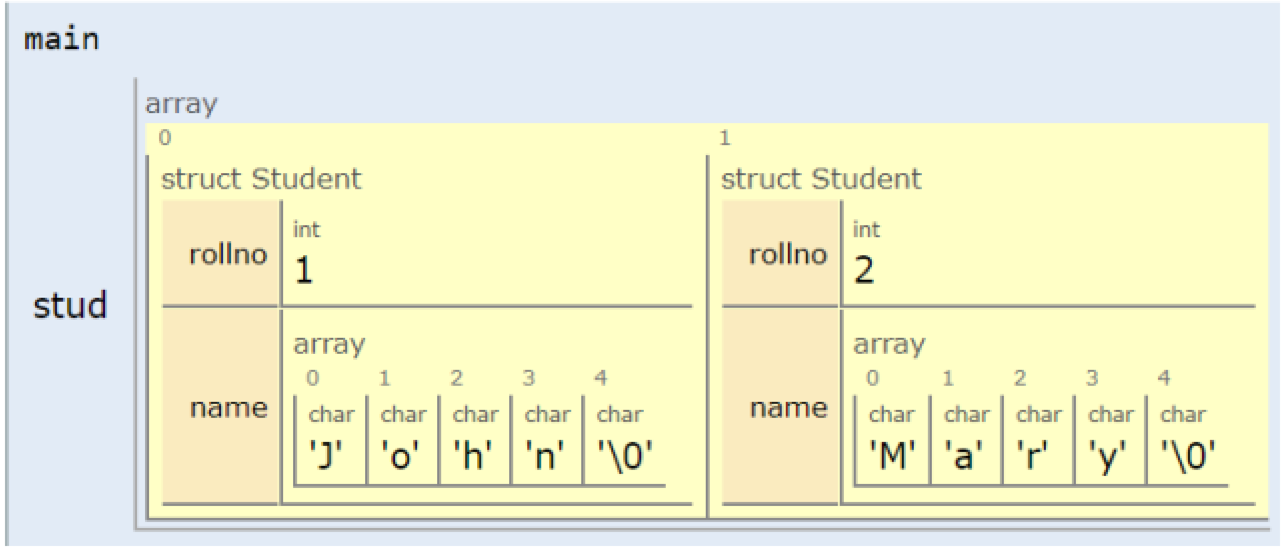
Example of Array of Structures

```
#include <stdio.h>
#include <string.h>
typedef struct Students{
        int rollno;
        char name[5];}Student;
int main() {
        Student stud[2];
        stud[0].rollno = 1;
        strcpy(stud[0].name, "John");
        stud[1].rollno = 2;
        strcpy(stud[1].name, "Mary");
for(int i=0; i <2; i++){
printf("Student rollno: %d\n", stud[i].rollno);
printf("Student name: %s\n", stud[i].name);  }
 return 0;}
```

Output:



```
Student rollno: 1
Student name: John
Student rollno: 2
Student name: Mary
```



main

stud

array

0 struct Student

rollno: int 1

name: array 0 char 'J' | 1 char 'o' | 2 char 'h' | 3 char 'n' | 4 char '\0'

1 struct Student

rollno: int 2

name: array 0 char 'M' | 1 char 'a' | 2 char 'r' | 3 char 'y' | 4 char '\0'

Pointers to structures

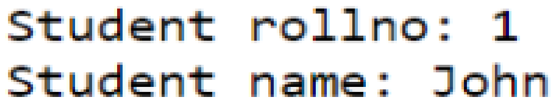
- Similar to declaring pointers to integers, double, etc. We can declare pointers to structs
- **Int* p1;**
- **float* p2;**
- **Student* p3;**
- The pointer **p3 stores the address to a struct** and works exactly like pointers to other data types
- We can access the values as follows

(* p3).roll_no

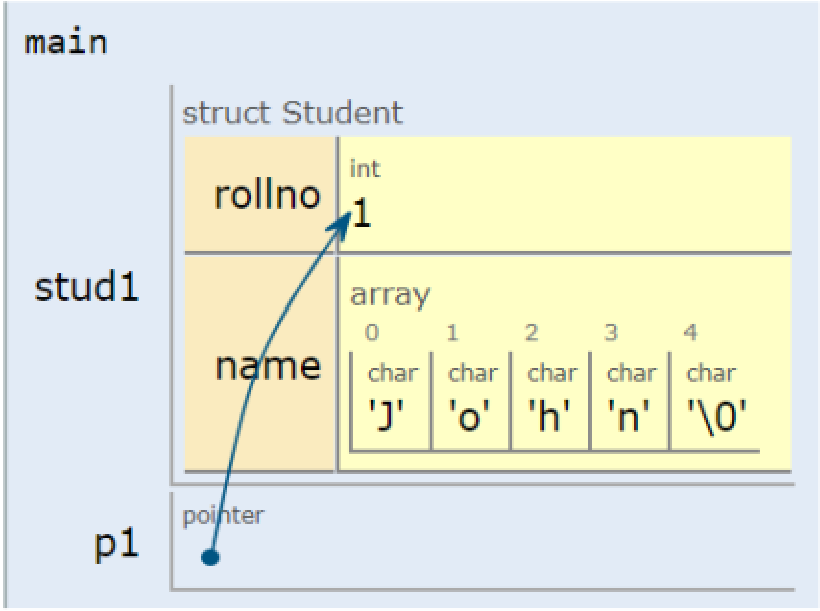
Example of Pointer to Structure

```
#include <stdio.h>
#include <string.h>
typedef struct Students{
        int rollno;
        char name[5];
}Student;
int main() {
        Student stud1;
        stud1.rollno = 1;
        strcpy(stud1.name, "John");
        Student* p1;
        p1 = &stud1;
        printf("Student rollno: %d\n", (*p1).rollno);
        printf("Student name: %s\n", (*p1).name);
        return 0;}
```

Output:



```
Student rollno: 1
Student name: John
```



The ‘->’ shorthand operator

- We can access the values as using the **shorthand -> operator**
- The below two statements have the **same meaning**

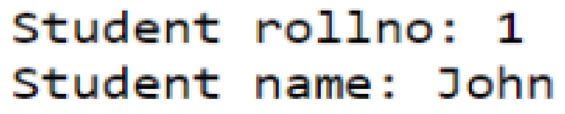
(* p3).roll_no;

p3->roll_no;

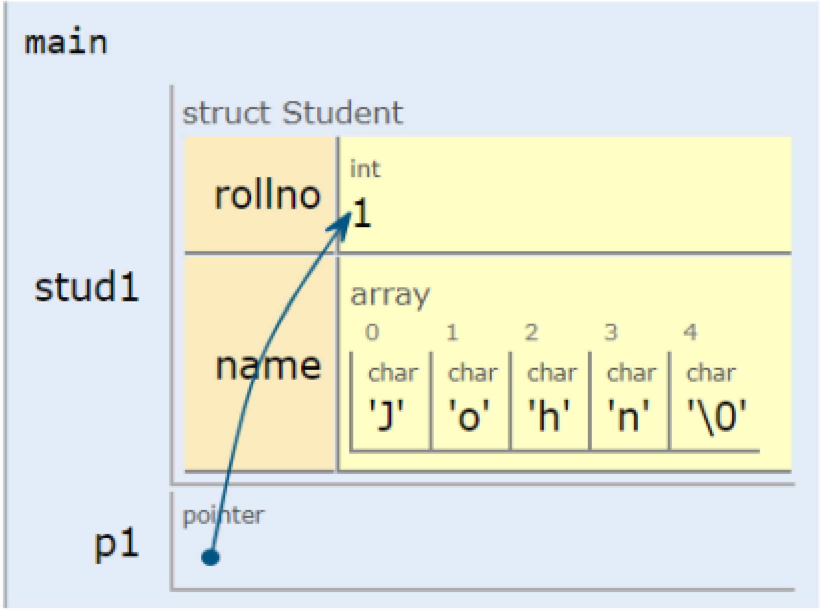
Example of Pointer to Structure

```
#include <stdio.h>
#include <string.h>
typedef struct Students{
        int rollno;
        char name[5];
}Student;
int main() {
        Student stud1;
        stud1.rollno = 1;
        strcpy(stud1.name, "John");
        Student* p1;
        p1 = &stud1;
        printf("Student rollno: %d\n", p1->rollno);
        printf("Student name: %s\n", p1->name);
        return 0;}
```

Output:



```
Student rollno: 1
Student name: John
```



Unions

- **Definition:** A union is similar to a structure but shares memory for all its members.

- **Syntax:**

```
union unionName {
   dataType1 member1;
   dataType2 member2;
   ...
};
```

Roll

Name[5]

- **Purpose:** Useful when variables don't need to exist simultaneously, conserving memory.

Example of Union in C

Example:

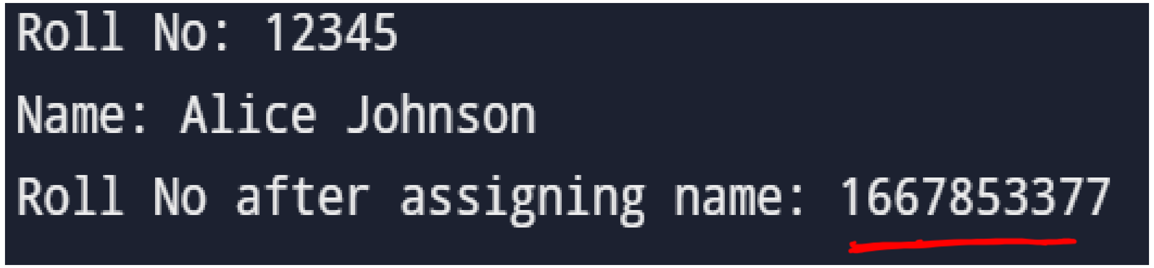
```
union Data {
   int intVal;
   float floatVal;
   char charVal;
};
union Data d1;
```

- **Explanation:** union Data defines a union with an integer, float, and char sharing the same memory location.

Example of C program using union

```c
#include <stdio.h>
#include <string.h>
union Student {
   int roll_no;
   char name[50];
};
int main() {
   union Student student;
   // Assign and display roll number
   student.roll_no = 12345;
   printf("Roll No: %d\n", student.roll_no);
   // Assign and display name
   strcpy(student.name, "Alice Johnson");
   printf("Name: %s\n", student.name);
   // Print roll_no after name assignment
   printf("Roll No after assigning name: %d\n", student.roll_no);  // Undefined value
   return 0;
}
```

Output:



```
Roll No: 12345
Name: Alice Johnson
Roll No after assigning name: 1667853377
```

Unions example (a real case)

- Assume that we are working with records of employees
- We would obviously declare a structure for employee
- Assume that we are reading the records from flat files
- The records have some issues
  - *The age of the employee is an integer*
  - *But, for some employees the age is written in English form "TWENTY"*
- For some reason, we want to preserve this: meaning don't modify/fix the error

Unions example (a real case)

```
struct employee {
        char *name;
        float salary;
        int utype;
        union {
                int int_age;
                char *str_age;
        } age;
} employees[100];
```

Age takes only 4 bytes of storage. Can you see why?

Assume we did this instead…

```
struct employee {
        char *name;
        float salary;
        int utype;
        int int_age;
        char *str_age;
} employees[100];
```

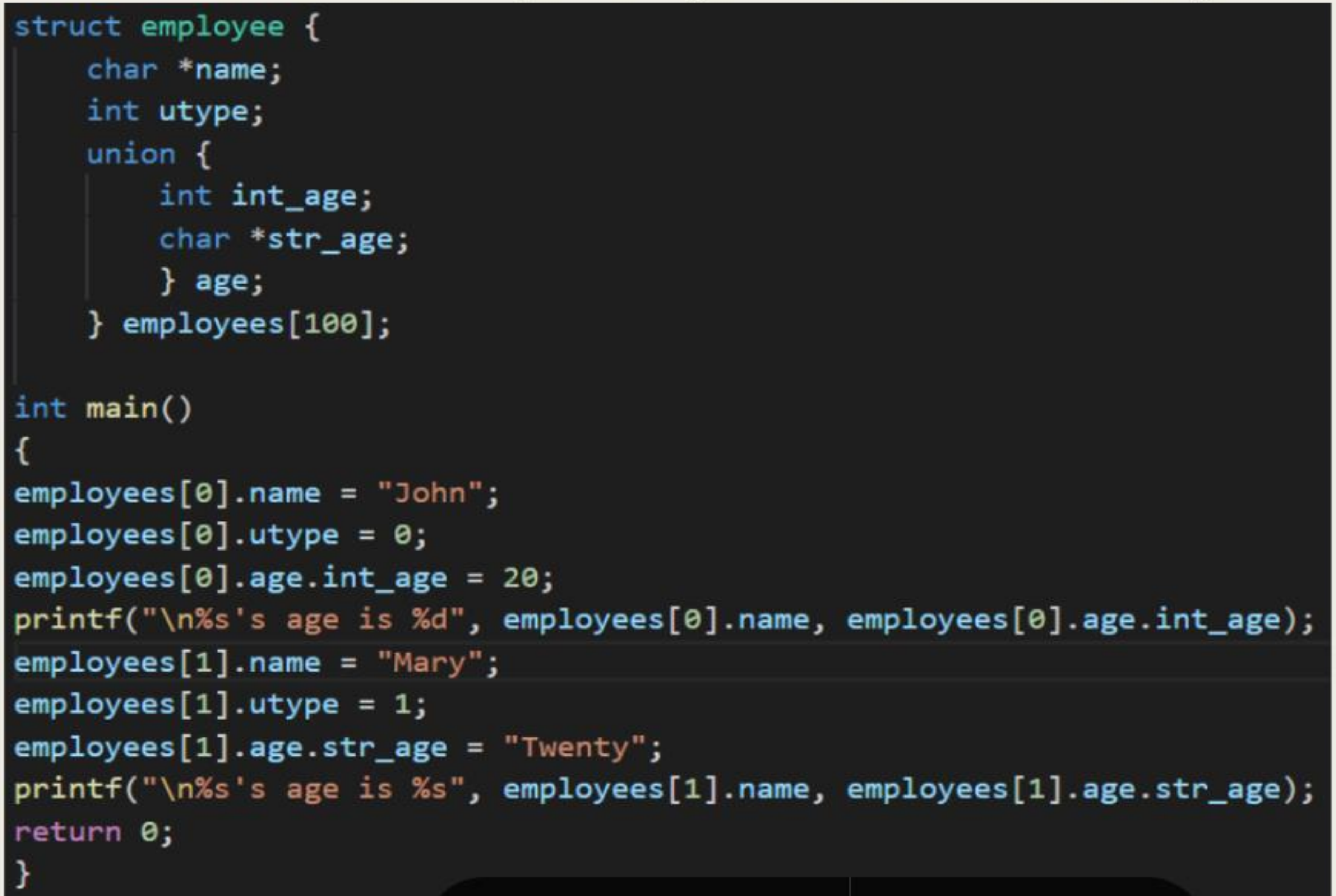
Each record will either have int_age or str_age, not both. But you are wasting memory.

Unions example (a real case)

```
struct employee {
        char *name;
        float salary;
        int utype;
        union {
                int int_age;
                char *str_age;
        } age;
} employees[100];
```

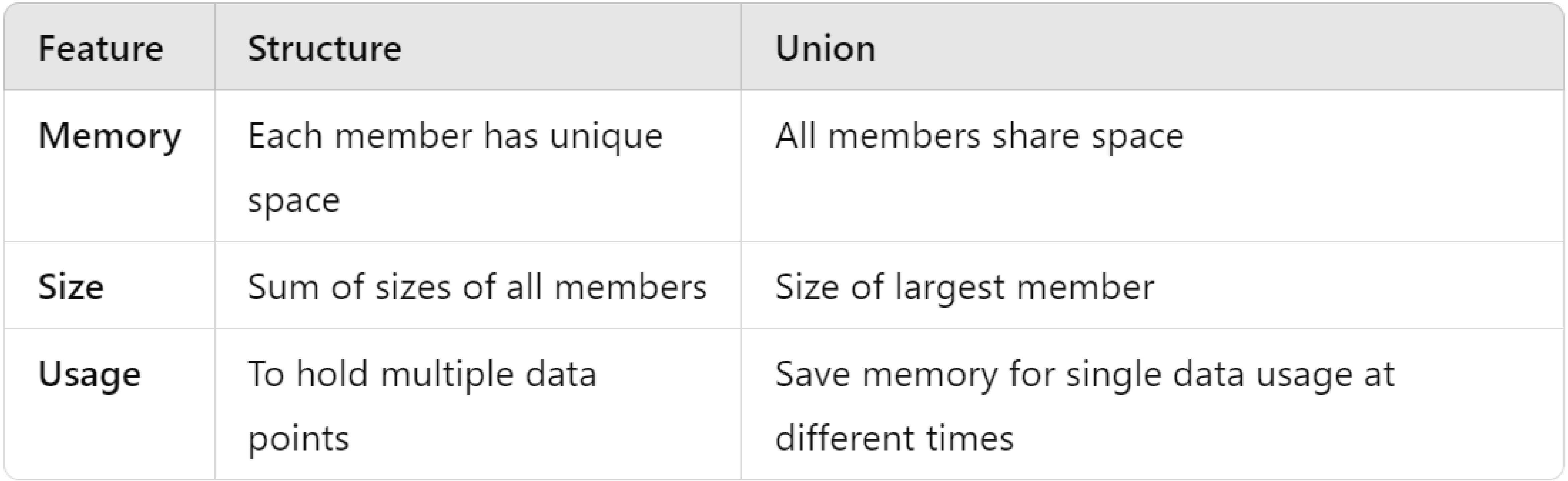
We will always use **utype** to figure out how to treat age

Unions example (a real case)



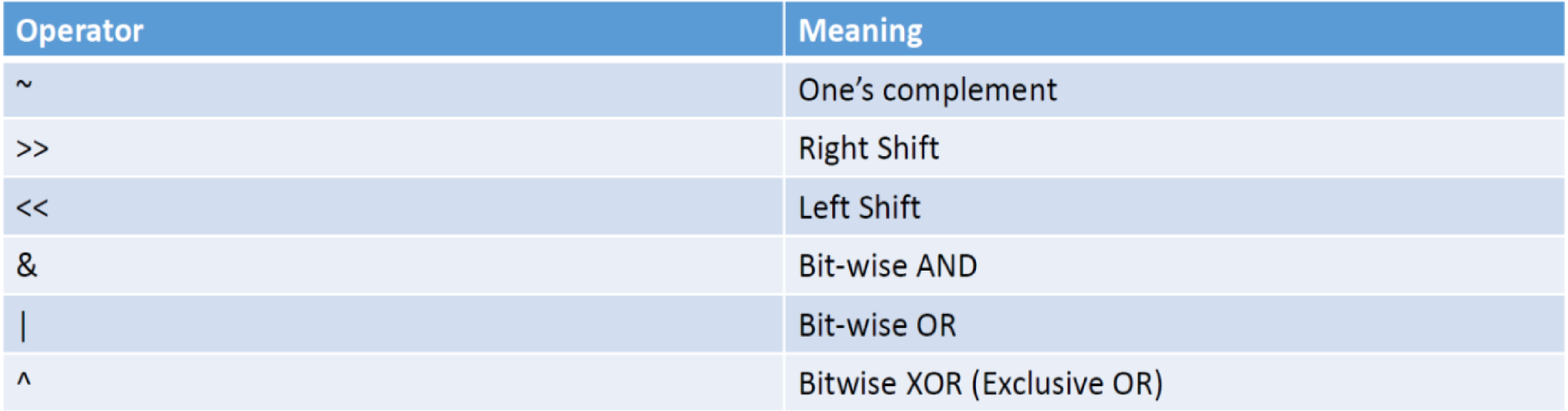
```c
struct employee {
    char *name;
    int utype;
    union {
        int int_age;
        char *str_age;
        } age;
    } employees[100];

int main()
{
employees[0].name = "John";
employees[0].utype = 0;
employees[0].age.int_age = 20;
printf("\n%s's age is %d", employees[0].name, employees[0].age.int_age);
employees[1].name = "Mary";
employees[1].utype = 1;
employees[1].age.str_age = "Twenty";
printf("\n%s's age is %s", employees[1].name, employees[1].age.str_age);
return 0;
}
```

Structures vs. Unions



Feature	Structure	Union
Memory	Each member has unique space	All members share space
Size	Sum of sizes of all members	Size of largest member
Usage	To hold multiple data points	Save memory for single data usage at different times

Bit Manipulation: Operators



Operator	Meaning
~	One’s complement
>>	Right Shift
<<	Left Shift
&	Bit-wise AND
|	Bit-wise OR
^	Bitwise XOR (Exclusive OR)

Note: Only defined for int and char values, not for float or double

Announcement

- Not considering SQ4 taken on 12th Nov.
- Uploaded Questions bank for Searching & Sorting
- Uploaded Question bank for Strings
- Uploaded Question bank for Structure, Union, & bit manipulation

ICP Project Discussion

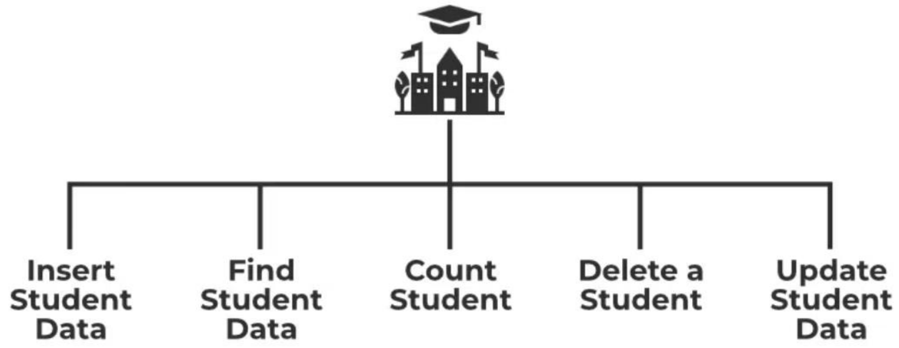
- Compulsory
- We will float a Google form to fill the details of group & project title
- Team of 2 students max
- Plag check would be there
- Mostly deadline would be Sunday midnight; (24th Nov 2024, 11:59pm)
- Submit the 1 to 2 page word doc report with your observations as well along with code
- If you have any query in the project then meet us on Monday lab(1 to 3) and Monday office hours (3 to 4pm)
- **Project titles are**: Student Management System, Library Management System, Health Management System, Hospital Management System, Flight reservation system, Others (You can choose by own but discuss with me on Monday)
- **Note**: You can use structure data type, array, loops, pointers, functions, recursion, etc to solve above problem.

Project Statement 1: *Student Management System*

- **Description:** Student Management maintained by the university is the way they are able to find data about every single student. Using a basic C application we can manage the data of the university.

- The functionality of the Student Management System Application is mentioned below:

1. Add Student Details (make a structure with Student name, roll No. & ICP marks)
2. Find the student by the given roll number
3. Find the ICP marks of students
4. Count Student
5. Delete a student
6. Update Student



Project Statement 2: *Library Management System*

- **Description:** *The library is the place where we find a collection of books organized in a particular order. In the library, we can collect book read them, and then return it. But, Managing a particular library is not an easy task. So, we can create a C language-based application using if-else statements, arrays, strings, switch cases, etc. Using this application we can easily manage the books in the library, we can get information about books, etc.*

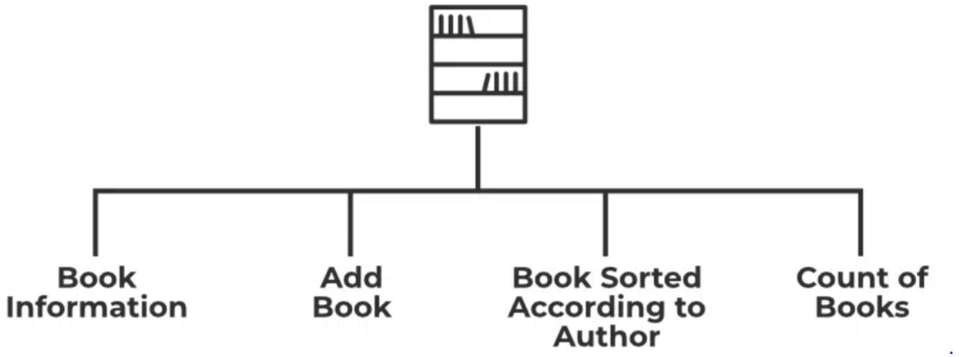
- **The functionality of the Library Management System is mentioned below:**

1. *Add book information. (Consider 5 types of ICP books, Physics, Mechanical books)*

2. *Display book information.*

3. *To list all books of a given author.*

4. *To list the count of books in the library*



Book Information

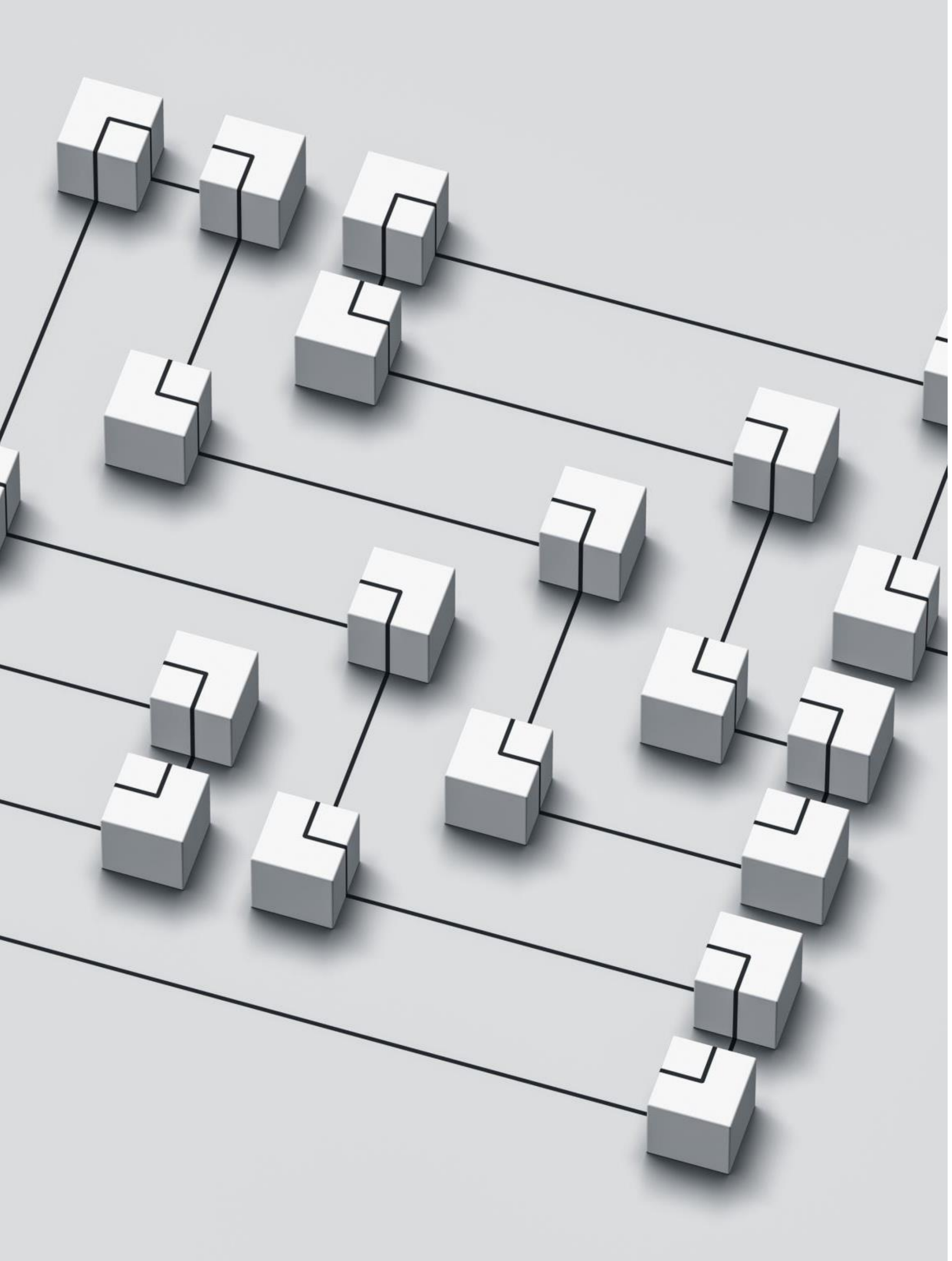
Add Book

Book Sorted According to Author

Count of Books

Announcement: Quiz 3 Discussion

- Quiz 3 is on **21st Nov (Thursday): from 12:30pm to 1pm**.
- Syllabus will be everything that we have covered **till today**, including **Structure, Union & Bit manipulation.**
- **5 to 6 questions**: MCQs, Short answer questions & coding question.
- **15 marks**



Upcoming Slides

- File Streaming and Processing
- Introduction to Data Structures