University of Anbar
College of Computer Science and Information Technology
Department of Computer Networks

C++ Structures (struct)

Lecture 6

Dr. Ali Al-Kubaisi

Asst.Lect. Abdulrahman Abbas Mukhlif

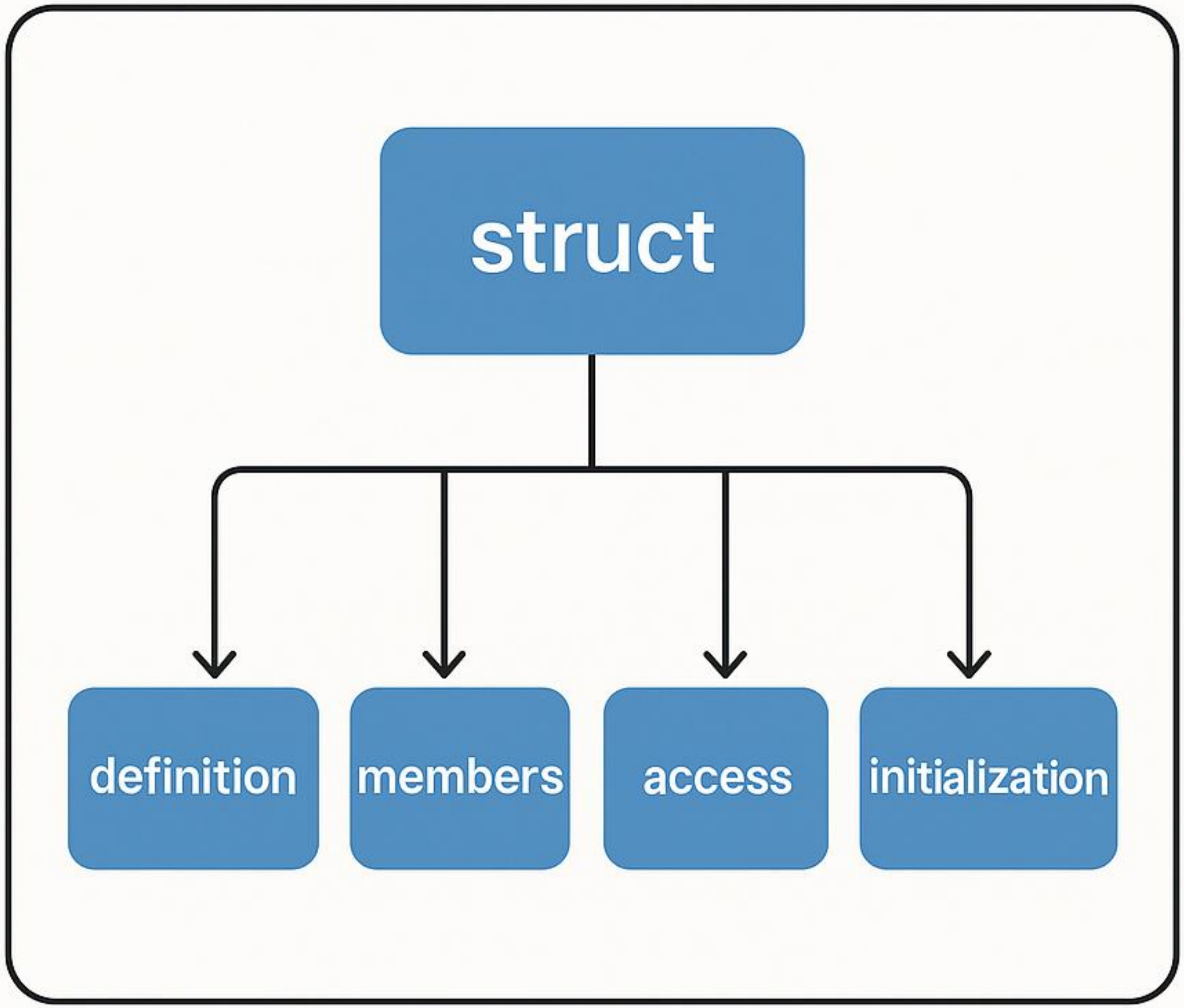
Outlines

- C++ Structures Concept
- Create a Structure
- Access Structure Members
- One Structure in Multiple Variables
- Named Structures
- Position Of The Name In A Struct Definition
- Creating an Array of Structs in C++

C++ Structures

- A structure in C++ is a user-defined data type that allows programmers to group together variables of **different data types** under a single name.
- Structures (also called structs) are a way to group several related variables into one place.
- Each variable in the structure is known as a **member** of the structure.
- Unlike an **array**, a structure can contain many different data types (int, string, bool, etc.).

C++ STRUCT CONCEPTS



Create a Structure

- To create a structure, use the struct keyword and declare each of its members inside curly braces.
- After the declaration, specify the name of the structure variable (**myStructure** in the example below):

```
struct {                // Structure declaration
  int myNum;            // Member (int variable)
  string myString;      // Member (string variable)
} myStructure;          // Structure variable
```

Access Structure Members

- To access members of a structure, use the dot syntax (.):

```
// Create a structure variable called myStructure
struct {
  int myNum;
  string myString;
} myStructure;

// Assign values to members of myStructure
myStructure.myNum = 1;
myStructure.myString = "Hello World!";

// Print members of myStructure
cout << myStructure.myNum << "\n";
cout << myStructure.myString << "\n";
```

One Structure in Multiple Variables

- You can use a comma (,) to use one structure in many variables:

```
struct {
  int myNum;
  string myString;
} myStruct1, myStruct2, myStruct3; // Multiple structure variables separated with commas
```

Example

```
struct {
  string brand;
  string model;
  int year;
} myCar1, myCar2; // We can add variables by separating them with a comma here
// Put data into the first structure
myCar1.brand = "BMW";
myCar1.model = "X5";
myCar1.year = 1999;

// Put data into the second structure
myCar2.brand = "Ford";
myCar2.model = "Mustang";
myCar2.year = 1969;

// Print the structure members
cout << myCar1.brand << " " << myCar1.model << " " << myCar1.year << "\n";
cout << myCar2.brand << " " << myCar2.model << " " << myCar2.year << "\n";
```

Named Structures

- By giving a name to the structure, you can treat it as a data type. This means that you can create variables with this structure anywhere in the program at any time.
- To create a named structure, put the name of the structure right after the **struct** keyword:

```
struct myDataType { // This structure is named "myDataType"
  int myNum;
  string myString;
};
```

- To declare a variable that uses the structure, use the name of the structure as the data type of the variable:

```
myDataType myVar;
```

Example

```
// Declare a structure named "car"
struct car {
  string brand;
  string model;
  int year;
};
int main() {
  // Create a car structure and store it in myCar1;
  car myCar1;
  myCar1.brand = "BMW";
  myCar1.model = "X5";
  myCar1.year = 1999;
  // Create another car structure and store it in myCar2;
  car myCar2;
  myCar2.brand = "Ford";
  myCar2.model = "Mustang";
  myCar2.year = 1969;
  // Print the structure members
  cout << myCar1.brand << " " << myCar1.model << " " << myCar1.year << "\n";
  cout << myCar2.brand << " " << myCar2.model << " " << myCar2.year << "\n";
  return 0;
}
```

Position Of The Name In A Struct Definition

In C++, the position of the **name** (identifier) in a struct definition affects what exactly you're naming.

1. **Name After struct Keyword**
2. **Name After the Structure Body (Anonymous Struct with Variable Name)**
3. **Both a Type Name and a Variable Name**

Syntax	Type Name	Variable Created
struct MyStruct { ... };	**Yes**	**No**
struct { ... } myVar;	**No**	**Yes**
struct MyStruct { ... } myVar;	**Yes**	**Yes**

Creating an Array of Structs in C++

- To create an array of structs, we first need to define the **struct** type and then declare an **array** of that struct using the below syntax.

```
// Define the struct
struct StructName {
    dataType1 member1;
    dataType2 member2;
  // more members...
};
// Declare an array of structs
StructName arrayName[arraySize];
```

```
//Example
// Defining the struct
struct Student {
    int id;
    string name;
};

int main()
{
    // Declaring the size of the array
    int size = 3;

    // Declaring an array of structs
    Student myArray[size];

    // Initializing data to structs present in the array
    for (int i = 0; i < size; i++) {
        myArray[i].id = i + 1;
        myArray[i].name = "Student" + to_string(i + 1);
    }

    // Printing the data of structs present in the array
    cout << "Array Elements:" << endl;
    for (int i = 0; i < size; i++) {
        cout << "Element " << i + 1
             << ": ID = " << myArray[i].id
             << ", Name = " << myArray[i].name << endl;
    }
}
```