



Structures

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.).

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 (.):

Example: Assign data to members of a structure and print it

```
// 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";
```

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

Example:

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

The example below shows how to use a structure in two different variables

Example: Use one structure to represent two cars

```
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:

Example:

```
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: Use one structure to represent two cars

```
// 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;
}
```