C++ Structure

What is a Structure?

- A structure is a **collection of variables** under a **single name**. **Variables** can be of **any type**: int, float, char etc.
- The **main difference** between **structure** and array is that **arrays are collections of the same data type**structure is a collection of variables under a single name. and The **data items** in a **structure** are called the ***members of the structure***.

The program PARTS defines the structure part, defines a structure variable of that type called part1, assigns values to its members, and then displays these values.

```
// parts.cpp
// uses parts inventory to demonstrate structures
#include <iostream>
using namespace std;
```

```
struct part                    //declare a structure
   {
   int modelnumber;            //ID number of widget
   int partnumber;             //ID number of widget part
   float cost;                 //cost of part
   };
```

```
int main()
   {
   part part1;                  //define a structure variable

   part1.modelnumber = 6244;    //give values to structure members
   part1.partnumber = 373;
   part1.cost = 217.55 ;
                                //display structure members
   cout << "Model "    << part1.modelnumber;
   cout << ", part "   << part1.partnumber;
   cout << ", costs $" << part1.cost << endl;
   return 0;
   }
```

The program's output looks like this:

```
Model 6244, part 373, costs $217.55
```

Declaring a Structure

- The structure is declared by using the keyword **struct** followed by structure name, also called a **tag**. Then the structure **members** (**variables**) are defined with their type and variable names inside the open and close **braces** "**{**"and "**}**".
- Finally, the closed braces end with a **semicolon** denoted as "**;**" following the statement. The above **structure declaration** is also called a Structure **Specifier**.

A Simple Structure

- Let's start off with a structure that contains **three variables**: two **integers** and a **floating-point number**.
- This structure represents an **item** in company's parts inventory.
- The program **PARTS defines** the structure **part**, defines a structure **variable** of that type called **part1**, assigns values to its **members**, and then displays these values.

```
  // uses parts inventory to demonstrate structures
 #include <iostream>
  using namespace std;

struct   part          //declare a structure
  {
  int    modelnumber;
   //ID  number of widget(structure member)
  int    partnumber;
  //ID   number of part(structure member)
  float  cost;
  //cost of part(structure member)
};
```

```
int main()
  {
   part part1;
  //define a structure variable
   part1.modelnumber = 6244;
  //give values (assign) to structure members
   part1.partnumber = 373;
   part1.cost = 217.55;
//display structure members

cout << “Model “ << part1.modelnumber;
cout << “, part “ << part1.partnumber;
cout << “, costs $” << part1.cost << endl;
return 0;
}
```

The program’s output looks like this:

```
Model 6244, part 373, costs $217.55
```

The **PARTS** program has **three** main aspects

1-**defining** the structure,
2-**defining** a structure variable,
3-**accessing** the members of the structure.

Let’s look at each of these.

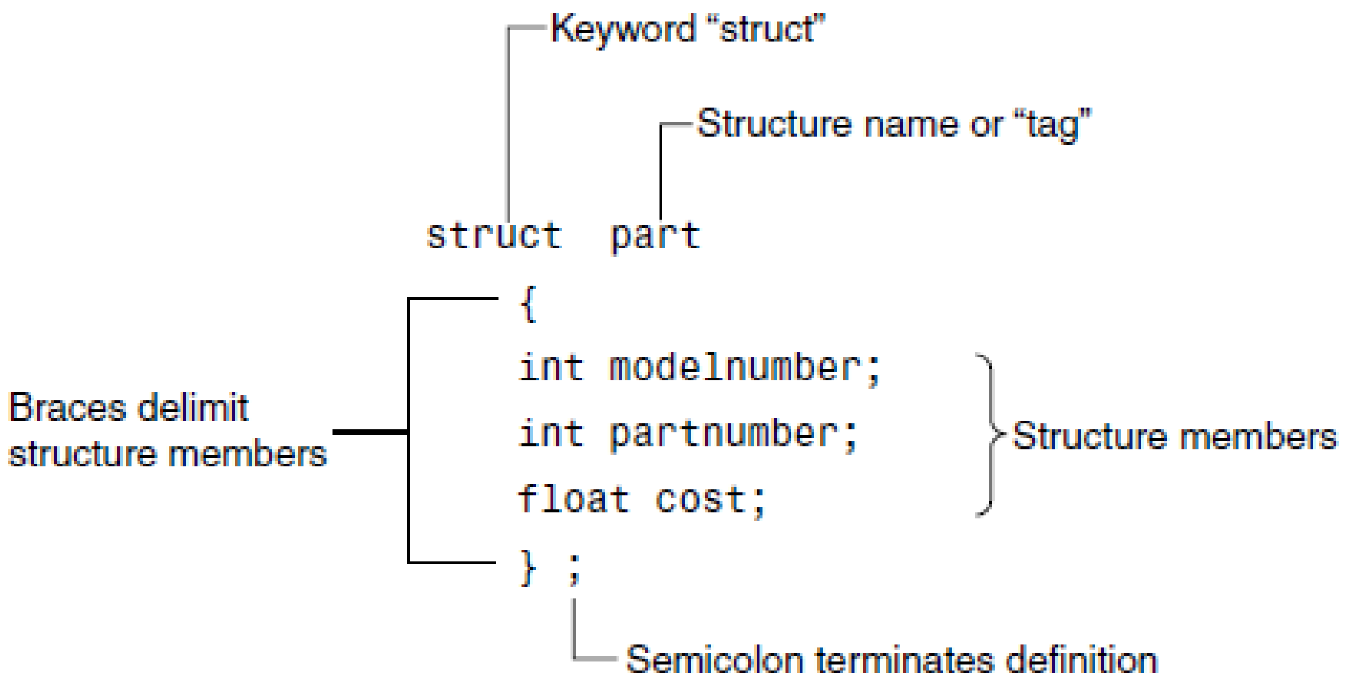
Defining the Structure

The structure **definition** tells **how** the structure is **organized**: It specifies what **members** the structure will have. Here it is:

```
struct part
{
 int   modelnumber;
 int  partnumber;
float   cost;
```

- };

Syntax of the structure definition.


Keyword "struct"
Structure name or "tag"
Braces delimit structure members
Structure members
Semicolon terminates definition

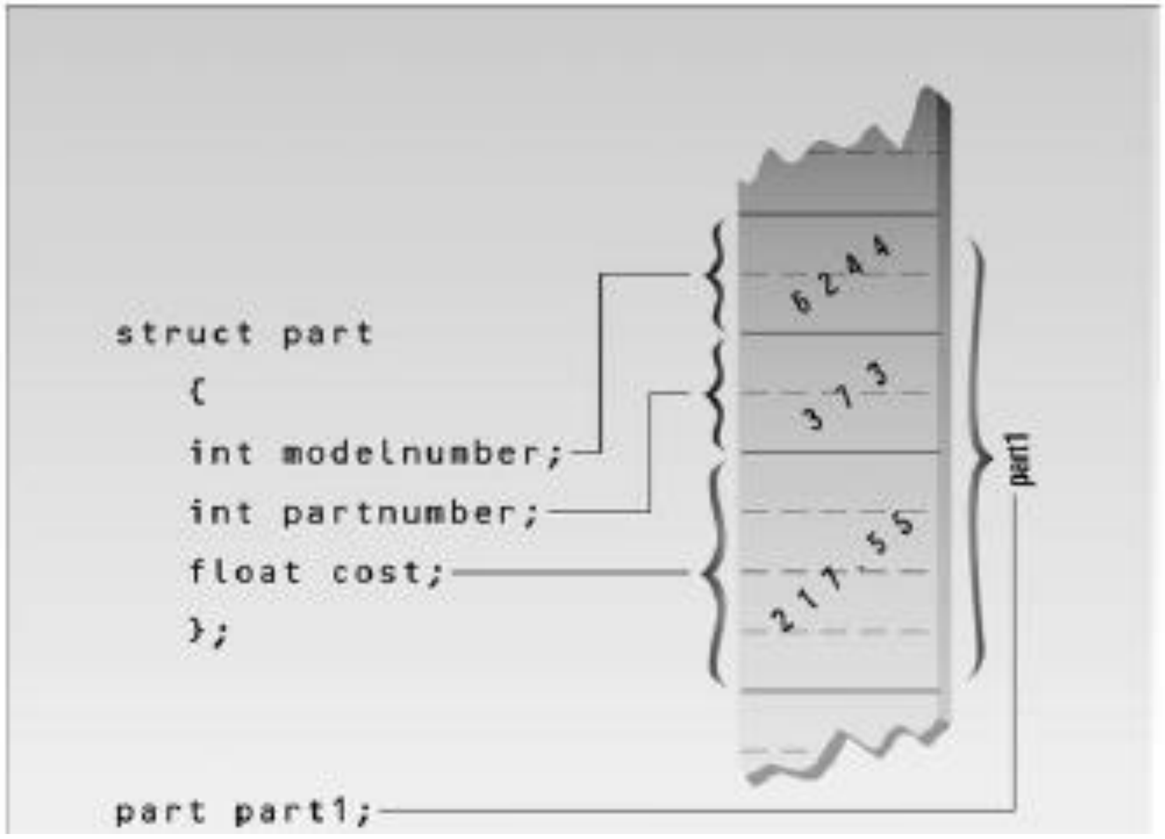
```
struct  part
    {
    int modelnumber;
    int partnumber;
    float cost;
    } ;
```

Defining a Structure Variable

- The first statement in main()

  part part1;
- defines a variable, called **part1**, of type structure part.
- This definition **reserves** space in memory for part1. In some ways we can think of the part structure as the **specification** for **a new data type**.
- **part** part1;
- **int** var1;

Structure members in memory.


```
struct part
    {
    int modelnumber;
    int partnumber;
    float cost;
    };

part part1;
```

Accessing Structure Members

members can be accessed using some thing called **the *dot operator*.** *Here's how the first member is given a value:*

- ***part1**.modelnumber = 6244*;

The structure member is written in **three** parts: the name of the structure variable (**part1**); **the dot operator**, which consists of a period (**.**); and the **member name** (**modelnumber)**.

Initializing Structure Members

- The next example shows how structure members can be **initialized** when the structure variableis defined.
- It also demonstrates that you can have **more** than **one variable** of a given structure type

Initializing Structure Members

- // shows initialization of structure variables
- #include <iostream>
- using namespace std;
- Struct part //specify a structure
- {
- int modelnumber; //ID number of widget
- int partnumber; //ID number of widget part
- float cost; //cost of part
- };

- int main()
- { //**initialize variable**
- part **part1** = { 6244, 373, 217.55F };
- part **part2**; //define variable
- //display first variable
- cout << “Model “ << part1.modelnumber;
- cout << “, part “ << part1.partnumber;
- cout << “, costs $” << part1.cost << endl;
- **part2** = **part1**; //assign first variable to second

- //display second variable
- cout << “Model “ << **part2**.modelnumber;
- cout << “, part “ << **part2**.partnumber;
- cout << “, costs $” << **part2**.cost << endl;
- return 0;
- }

- The part1 structure variable’s members are initialized when the variable is defined:
- part part1 = { 6244, 373, 217.55 };
- Here’s the output:

```
Model 6244, part 373, costs $217.55
```

```
Model 6244, part 373, costs $217.55
```

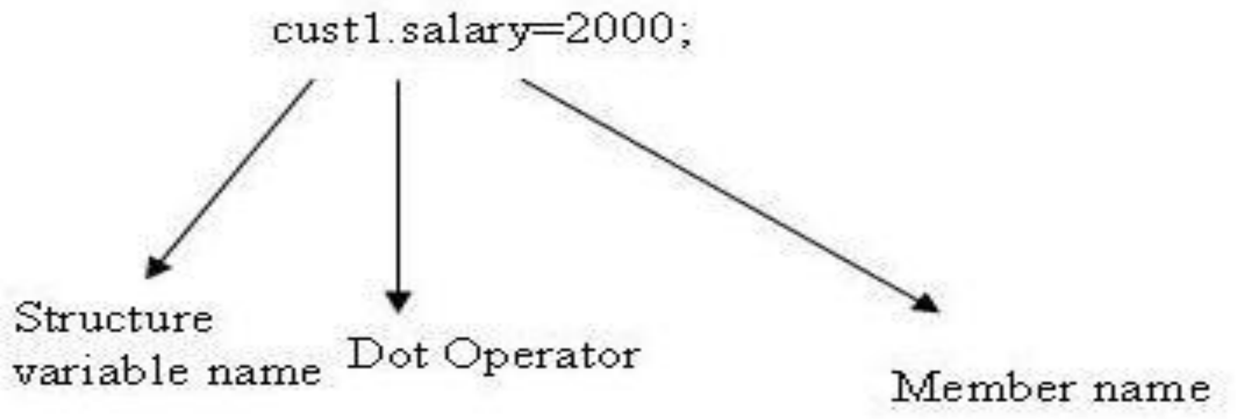
Example:

- **Three variables**: **custnum** of type **int**, **salary** of type **int**, **commission** of type **float** are structure members and the structure **name** is **Customer**.

- This structure is declared as follows:

For example:

- A programmer wants to **assign** 2000 for the structure member **salary** in the above example of structure **Customer** with structure variable **cust1** this is written as:



For Example

```
•   #include <iostream>
•   using namespace std;
•   struct Customer
•   {
•           int custnum;
•           int salary;
•           float commission;
•   };
•
```

Example continued

```
•     void main( )
•     {
       //initialize variable
•           Customer cust1={100,2000,35.5};
•           Customer cust2;
•           cust2=cust1;
•           cout << "n Customer Number: "<< cust1.custnum << "; Salary:
      Rs."<< cust1.salary << "; Commission: Rs." << cust1.commission;
•           cout << "n Customer Number: "<< cust2.custnum << "; Salary:
      Rs."<< cust2.salary << "; Commission: Rs." << cust2.commission;
•     }
```