Structures

Ho Dac Hung

Structures

- A structure is a collection of simple variables. The variables in a structure can be of different. The data items in a structure are called the members of the structure.

Structures

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

Structures

part part1; //define a structure variable

part1.modelnumber = 6244; //give values to structure members

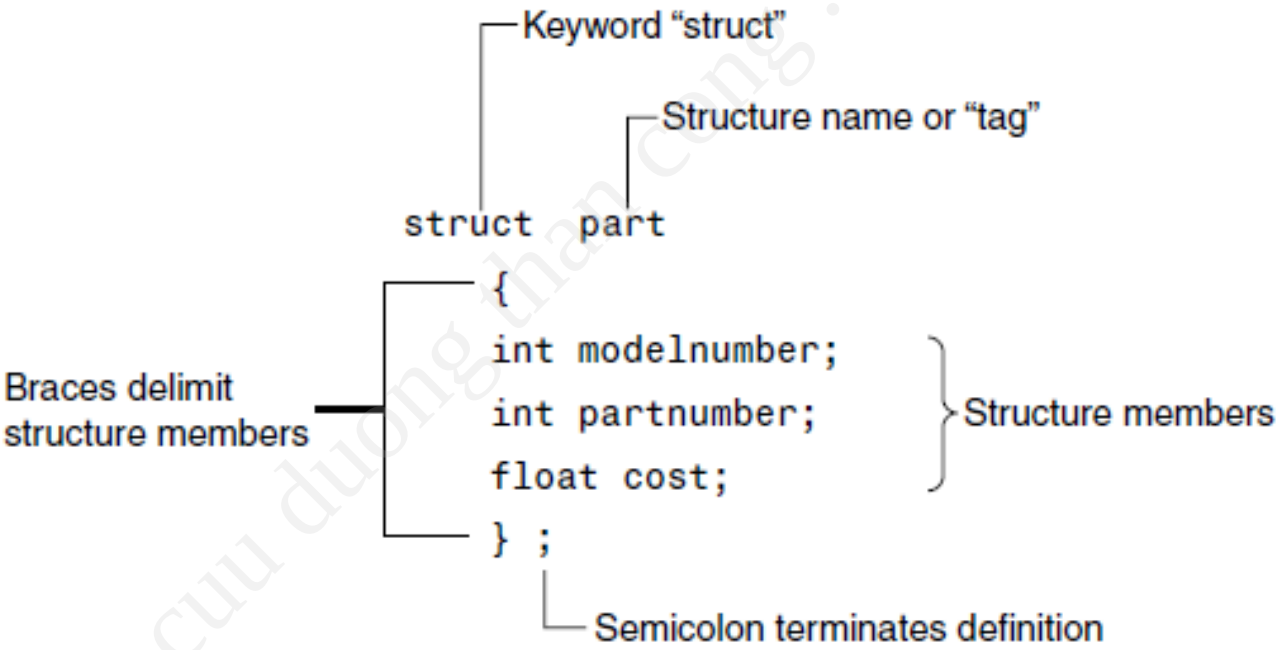
part1.partnumber = 373;

part1.cost = 217.55F;

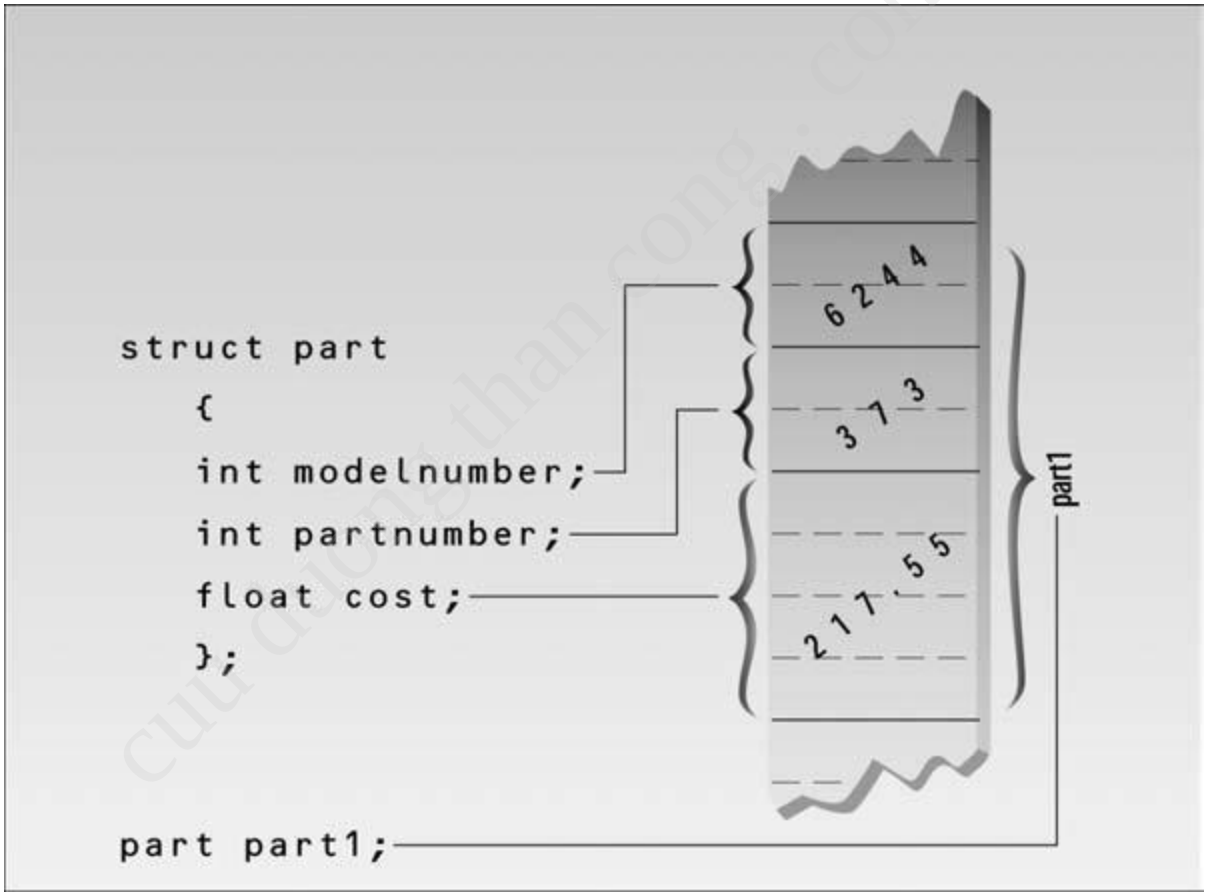
Structures

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

Structures



Structures



Structures

- Initializing Structure Members

```
part part1 = { 6244, 373, 217.55F };
```

- Structure Variables in Assignment Statements

```
part part2;
part2 = part1;
```

A Measurement Example

```
struct Distance
{
    int feet;
    float inches;
};
```

Structures Within Structures

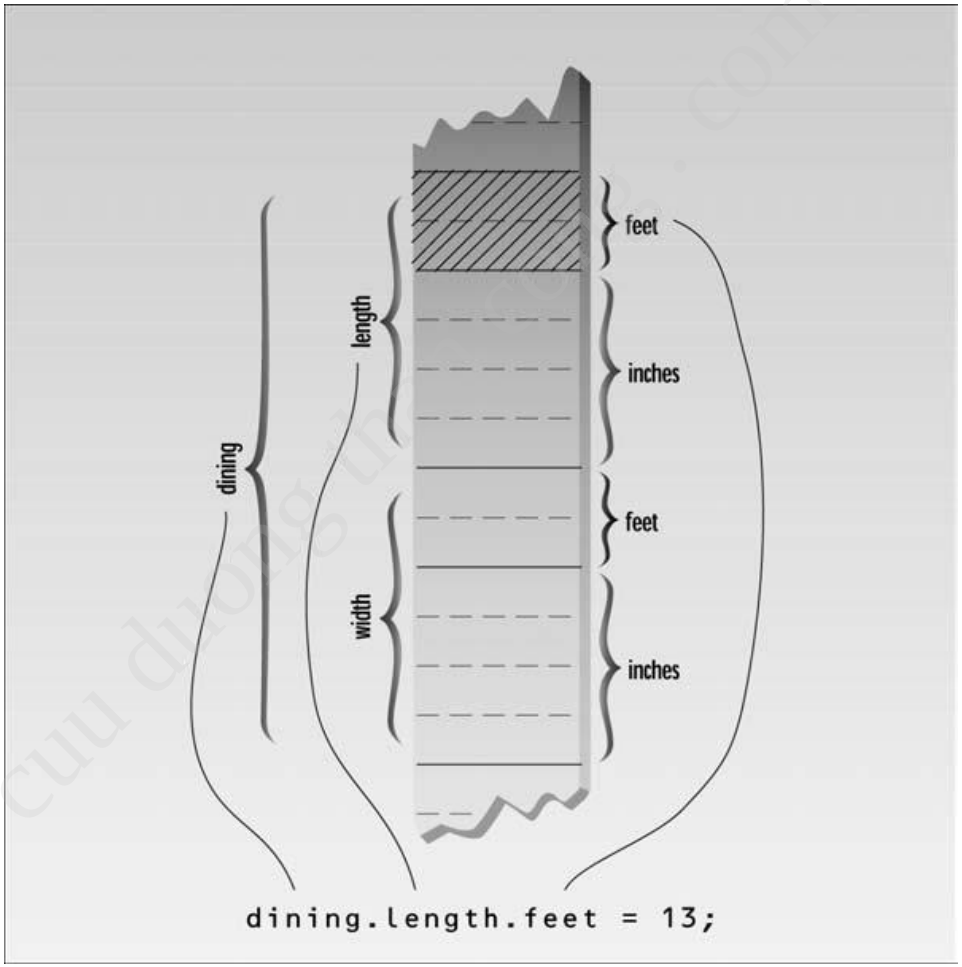
- You can nest structures within other structures.

```
struct Room
{
    Distance length;
    Distance width;
}
```

Structures Within Structures

```
Room dining; //define a room
dining.length.feet = 13; //assign values to room
dining.length.inches = 6.5;
dining.width.feet = 10;
dining.width.inches = 0.0;
float l = dining.length.feet + dining.length.inches/12;
float w = dining.width.feet + dining.width.inches/12;
```

Structures Within Structures



A Card Game Example

```
const int clubs = 0; //suits
const int diamonds = 1;
const int hearts = 2;
const int spades = 3;
const int jack = 11; //face cards
const int queen = 12;
const int king = 13;
const int ace = 14;
```

A Card Game Example

```
struct card
{
int number; //2 to 10, jack, queen, king, ace
int suit; //clubs, diamonds, hearts, spades
};
```

A Card Game Example

```
card card1 = { 7, clubs }; //initialize card1
cout << “Card 1 is the 7 of clubs\n”;
card card2 = { jack, hearts }; //initialize card2
cout << “Card 2 is the jack of hearts\n”;
card card3 = { ace, spades }; //initialize card3
cout << “Card 3 is the ace of spades\n”;
```

Enumerations

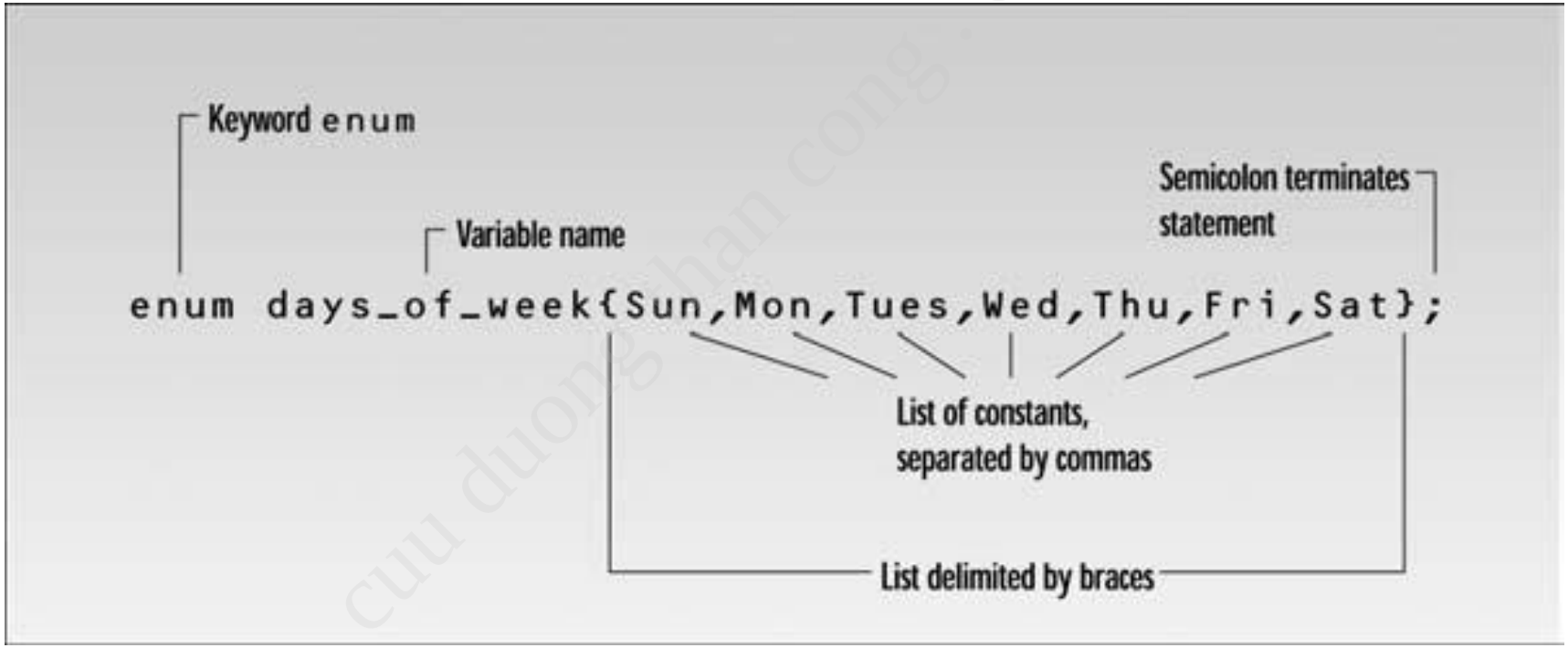
- As we've seen, structures can be looked at as a way to provide user-defined data types. A different approach to defining your own data type is the enumeration. This feature of C++ is less crucial than structures. You can write perfectly good object-oriented programs in C++ without knowing anything about enumerations.

```
enum days_of_week { Sun, Mon, Tue, Wed,
                    Thu, Fri, Sat };
```

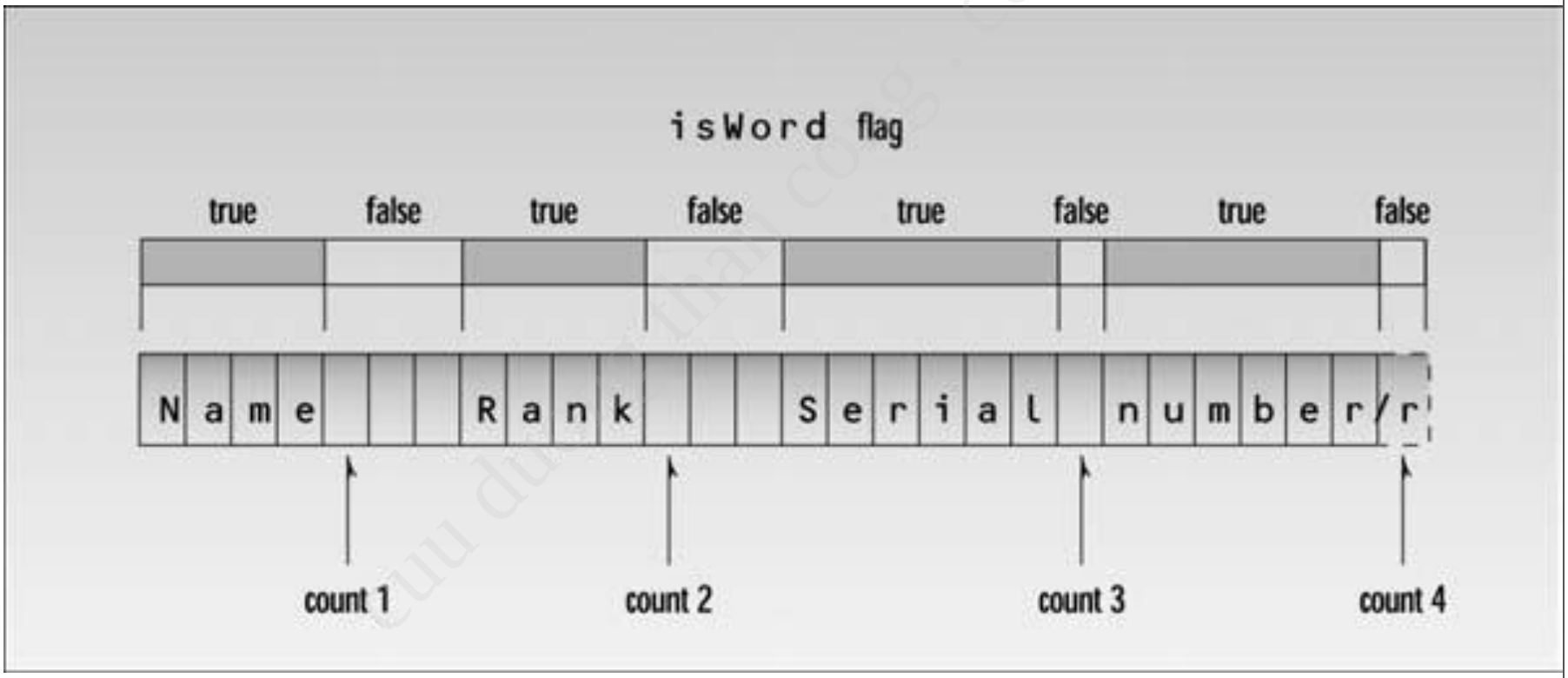
Enumerations

```
days_of_week day1, day2;
day1 = Mon;
day2 = Thu;
int diff = day2 - day1;
cout << “Days between = “ << diff << endl;
if(day1 < day2)
    cout << “day1 comes before day2\n”;
```

Enumerations



Enumerations



Enumerations

```
enum itsaWord { NO, YES };

itsaWord isWord = NO;
char ch = ‘a’;
int wordcount = 0;
cout << “Enter a phrase:\n”;
```

```
do {
    ch = getche();
    if(ch==’ ‘ || ch==’\r’)
    {
        if( isWord == YES )
        {
            wordcount++;
            isWord = NO;
        }
    }
    else
         if( isWord == NO )
            isWord = YES;
} while( ch != ‘\r’ );
```

```
const int jack = 11;
const int queen = 12;
const int king = 13;
const int ace = 14;
enum Suit { clubs, diamonds, hearts, spades };

struct card
{
    int number; //2 to 10, jack, queen, king, ace
    Suit suit; //clubs, diamonds, hearts, spades
};
```