

Creating Your Own Data Type

- Object
- Structure
- Define, declare and using structure object
- Structure members
- Union

Objects

- Object is an physical instance
- struct and union in C
- class in C++
- Examples
 - `int a = 4;`
 - Human Mary, John;
 - Furniture chair, table;

Members in struct

```
struct Book
{
    char title[80];
    char author[80];
    char publisher[80];
    int year;
};
```

Declaring Variables of a Structure Type

Book paperback; ← same
struct Book paperback; ←
Book novel;
Book *pTravelGuide; } ← same
Book languageGuide[10]; }
Book novel, *pTravelGuide, languageGuide[10];

```
graph LR; A[Book paperback;] -- same --> B[struct Book paperback;]; B -- same --> C[Book *pTravelGuide;]; B -- same --> D[Book languageGuide[10];]; C --- E[ ]; D --- E; E -- same --> F[Book novel, *pTravelGuide, languageGuide[10];];
```

Defining Structure Members and Variables

```
struct Book
{
    char title[80];
    char author[80];
    char publisher[80];
    int year;
} dictionary, thesaurus;
```

Creating Objects of a Structure Type

```
Book PDtextbook = { "C++ Beginning",  
    "Ivor Horton", "Wrox", 2000};
```

```
Book novels[] = {  
    {"笑傲江湖", "金庸", "港龍", 1980},  
    {"Our Game", "John Le Garre", "Hodder  
& Stoughton", 1995},  
    {"Illywhacker", "Peter Carey", "Faber &  
    Faber", 1985}    };
```

Accessing the Members of a Structure Object

```
novels[2].year = 1900;
```

```
PDtextbook.year += 2;
```

- Program 11.1

Member Functions of a Structure

```
struct Box
{
    double length;
    double breadth;
    double height;
    double volume( );
};

double Box::volume( )
{
    return length * breadth * height; }
```


Using Pointers with a Structure

```
Box * pBox = 0, abox; // NULL pointer
```

```
pBox = &abox;
```

```
Box theBox = {80.0, 50.0, 40.0};
```

```
Box *pBox = &theBox;
```

```
Book *pDictionary = new Book;
```

```
Delete pDictionary;
```

```
(*pBox).height += 10; // theBox.height = 50.0;
```

```
*(pBox.height) += 10; // compiler error
```

```
pBox->height += 10; // dereferencing operator
```

- Program 11.2

Unions

- A union is a data type that allows you to use the same block of memory to store values of different types at different times.
- Advantage
 - Memory saving
- Disadvantage
 - Access members without type casting can lead to an unexpected result

Using a Union

- Three ways of using a union
 - Enable the same block of memory to store different variables (possibly of different types)
 - Memory saving on a larger scale involves arrays
 - To interpret the same data in two or more different ways.

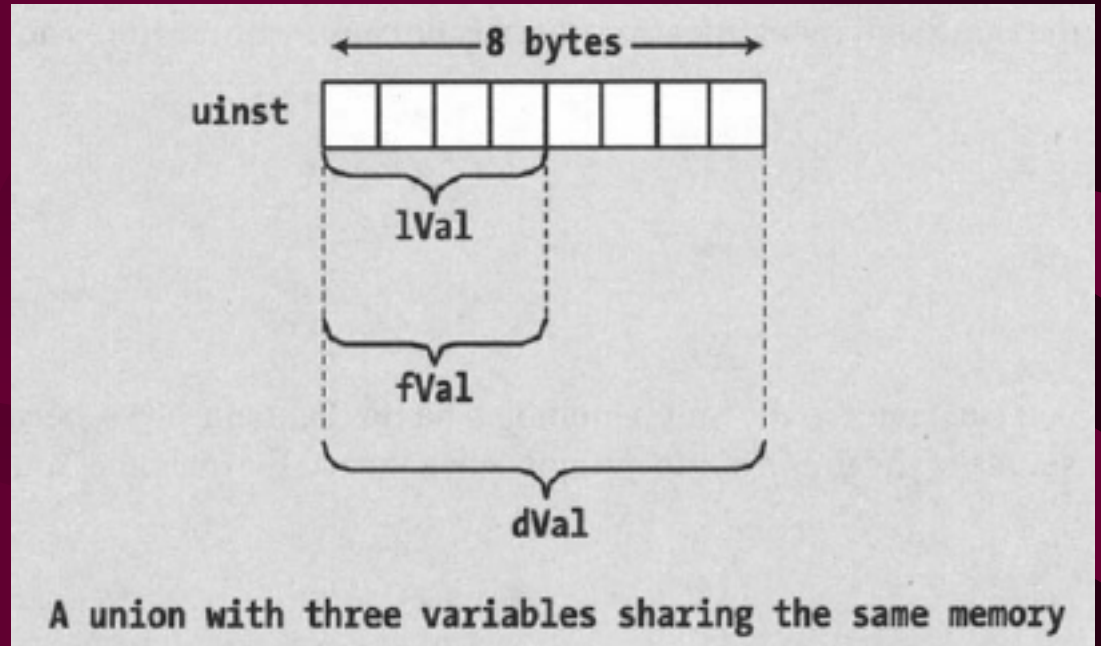
Declaring Unions

```
union shareDL  
{  
    double dVal;  
    long lval;  
} myUnion;    // 8 bytes
```

Initializing a Union

```
union ShareDLF
{
    double dVal;
    long lVal;
    float fVal;
};
```

```
ShareDLF unist = {1.5}; // initial the first member
ShareDLF unist;          unist.lVal=10;
```



Anonymous Union

```
union
{
    char* pVal;
    double dVal;
    long lVal;
} uvalue;
uvalue.dVal = 10.0;
```

Content in a Union

union item

```
{  
    double dData;  
    float fData;  
    long lData;  
    short iData;  
} value;
```

```
value.dData = 25.0;  
value.lData = 32768L;  
cout << value.iData;  
        // -32768  
  
cout << value.dData;  
        // ???????.?????
```

Complex Structure

```
enum Type {Double, Float, Long, Int};  
struct SharedData  
{ union  
  { double dData;  
    float fData;  
    long lData;  
    int iData;  
  };  
  Type type;  
};
```

```
SharedData value = {25.0, Float};  
SharedData value = {25.0};  
                                // {25.0, Double};  
value.lData = 10;  
value.type = Long;  
if (value.type == Long)  
    value.lData++
```


Structures with Structures As Members

```
struct Person
{
    Name name;
    Date birthdate;
    Phone number;
};
```

```
struct Name
{ char firstname[80];
  char surname[80];  };
```

```
struct Date
{ int day; int month; int year; };
```

```
struct Phone
{ int areacode; int number; };
```

Member Functions in a Structure

```
struct Name
{
    char firstname[80];
    char surname[80];
    void show()
        { cout << firstname << " " << surname; }
};
```

- Program 11.3