西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C++程序设计
Programming in C++

1011018

主讲：魏英，计算机学院

自定义数据类型

- 3、共用体类型
- 4、枚举类型

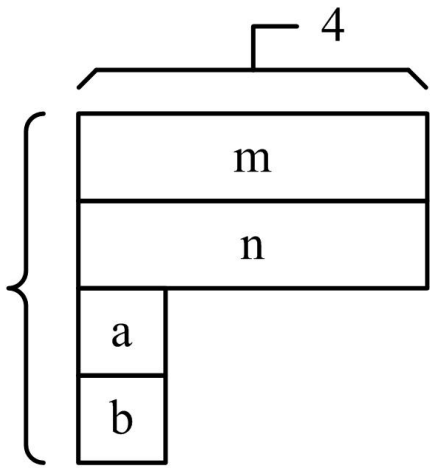
- 共用体（union）是一种成员共享存储空间的结构体类型。共用体类型是抽象的数据类型，因此程序中需要事先声明具体的共用体类型，一般形式为：

```
union 共用体类型名 {
        成员列表
};
```

- 共用体类型名与union一起作为类型名称，成员列表是该类型数据元素的集合。一对大括号｛｝是成员列表边界符，后面必须用分号（；）结束。

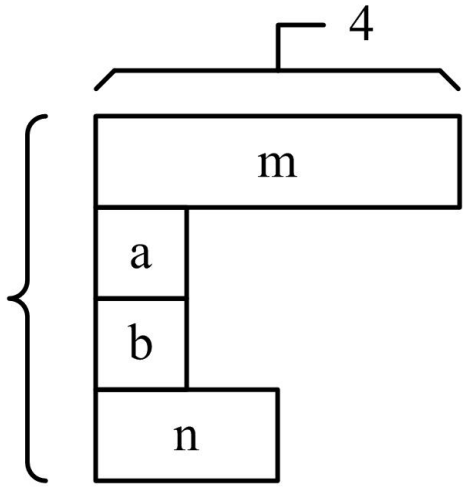
```
union A {
    int m,n; //整型成员
    char a,b; //字符成员
};
```

- 共用体中每个成员与其他成员之间共享内存。
- 对于union A，m、n、a、b共享内存单元，其内存结构如图所示。



```
union B {
    int m; //整型成员
    char a,b; //字符成员
    short n; //短整型成员
};
```

- 对于union B，m、a、b、n共享内存单元，其内存结构如图所示。



- 结构体与共用体的内存形式是截然不同的。共用体内存长度是所有成员内存长度的最大值，结构体内存长度是所有成员内存长度之和。可以用sizeof取它们的内存长度进行比较。

- 与结构体对象相似，定义共用体对象也有三种形式：

①先声明共用体类型再定义共用体对象

```
union 共用体类型名 共用体对象名列表;
```

②同时声明共用体类型和定义共用体对象

```
union 共用体类型名 { 成员列表 } 共用体对象名列表;
```

③直接定义共用体对象

```
union { 成员列表 } 共用体对象名列表;
```

- 定义共用体对象时可以进行初始化，但只能按一个成员给予初值，例如：

```
union A x={ 5678 }; //正确，只能给出1个初值
union A y={5,6,7,8}; //错误，试图给出4个初值（结构体做法）
```

- 共用体对象的使用主要是引用它的成员，方法是对象成员引用运算（. ），例如：

```
1   x.m=5678; //给共用体成员赋值
2   cout<<x.m<<","<<x.n<<","<<x.a<<","<<x.b<<endl;
    //输出 5678,5678,46,46
3   cin>>x.m>>x.n>>x.a>>x.b;
4   x.n++; //共用体成员运算
```

- 第1句给成员m赋值5678，由于所有成员内存是共享的，因此每个成员都是这个值。
- 第2句输出m和n为5678，输出a和b为46，因为a和b类型为char，仅使用共享内存中的一部分（4个字节的低字节），即5678（0x162E）的0x2E（46）。

- 第三句每个成员的起始地址是相同的，当运行第3句时输入1 2 3 4↙，x.m得到1，但紧接着x.n得到2时，x.m也改变为2了（因为共享），依次类推，最终x.b得到4时，所有成员都是这个值。

- 第4句当x.n自增运算后，所有成员的值都改变了。

- 由于成员是共享存储空间的，使用共用体对象成员时有如下特点：
- ①修改一个成员会使其他成员发生改变，所有成员存储的总是最后一次修改的结果；
- ②所有成员的值是相同的，区别是不同的类型决定了使用这个值的全部或是部分；
- ③所有成员的起始地址值是相同的，因此通常只按一个成员输入、初始化；

- 枚举类型是由用户自定义的由多个命名枚举常量构成的类型，其声明形式为：

```
enum 枚举类型名 {命名枚举常量列表};
```

- 例如：

```
enum DAYS {MON,TUE,WED,THU,FRI,SAT,SUN};
```

- DAYS是枚举类型，MON等是命名枚举常量。默认时枚举常量总是从0开始，后续的枚举常量总是前一个的枚举常量加一。如MON为0，TUE为1，……，SUN为6。

- （1）可以在（仅仅在）声明枚举类型时，为命名枚举常量指定值。例如：

```
enum COLORS {RED=10,GREEN=8,BLUE,BLACK,WHITE};
```

- 则RED为10、GREEN为8、BLUE为9、BLACK为10、WHITE为11。

- （2）命名枚举常量是一个整型常量值，也称为枚举器，在枚举类型范围内必须是唯一的。命名枚举常量是右值不是左值，例如：

```
RED=10; //错误，RED不是左值，不能被赋值
GREEN++; //错误，GREEN不是左值，不能自增自减
```

- 义枚举类型对象有三种形式：

```
①enum 枚举类型名 {命名枚举量列表} 枚举对象名列表;
②enum 枚举类型名 枚举对象名列表;
//在已有枚举类型下，最常用的定义形式
③enum {命名枚举量列表} 枚举对象名列表;
//使用较少的定义形式
```

- 可以在定义对象时进行初始化，其形式为：

枚举对象名1=初值1，枚举对象名2=初值2，……；

```
enum DIRECTION {LEFT,UP,RIGHT,DOWN,BEFORE,BACK} dir=LEFT;
```

- 当给枚举类型对象赋值时，若是除枚举值之外的其他值，编译器会给出错误信息，这样就能在编译阶段帮助程序员发现潜在的取值超出规定范围的错误。例如：

```
enum COLORS color;
color=101; //错误，不能类型转换
color=(COLORS)101; //正确，但结果没有定义
```

- 可以用typedef声明一个新类型名来代替已有类型名，其形式为：

```
typedef 已有类型名 新类型名;
```

- 其中已有类型名必须是已存在的数据类型的名称，新类型名是标识符序列，习惯上用大写标识；如果是多个新类型名，用逗号（，）作为间隔。最后以分号（；）结束。例如：

```
typedef unsigned char BYTE; //按计算机汇编指令习惯规定的字节型
typedef unsigned short WORD; //按计算机汇编指令习惯规定的字类型
typedef unsigned long DWORD; //按计算机汇编指令习惯规定的双字类型
```

C程序设计