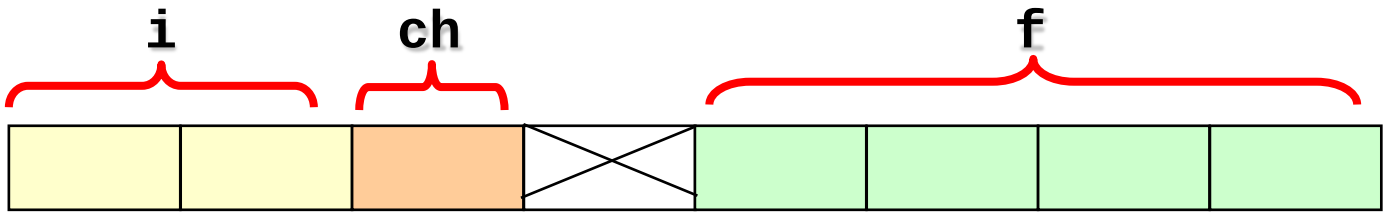
用户自定义的数据类型

- 结构体（struct）
  - 把关系紧密且逻辑相关的多种不同类型的的变量，组织到一个统一的名字之下
- 共用体，也称联合（union）
  - 把情形互斥但逻辑相关的多种不同类型的变量，组织到一个统一的名字之下

共用体与结构体的不同点

```
struct sample
{
    short  i;
    char   ch;
    float  f;
};
```

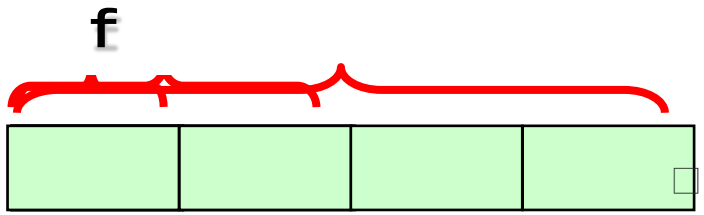
```
printf("%d\n", sizeof(struct sample));
```



8个字节

```
union sample
{
   short  i;
   char   ch;
   float  f;
};
```

```
printf("%d\n", sizeof(union sample));
```

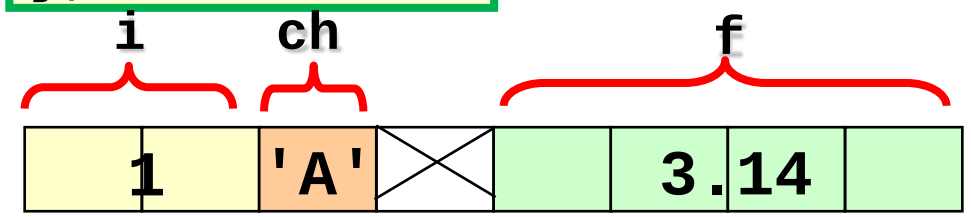


取决于占空间最多的那个成员变量

共用体与结构体的不同点

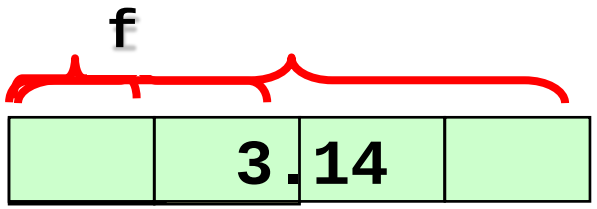
```
struct sample
{
    short  i;
    char   ch;
    float  f;
};
```

```
struct sample s;
s.i = 1;
s.ch = 'A';
s.f = 3.14;
```




```
union sample
{
   short  i;
   char   ch;
   float  f;
};
```

```
union sample u;
u.i = 1;
u.ch = 'A';
u.f = 3.14;
```



- 同一内存在每一瞬时只能保存一个成员
- 起作用的成员是**最后一次赋值**的成员

```
struct sample s = {1, 'A', 3.14};
```

```
union sample u = {1};
```

- 只能对共用体的第一个成员进行初始化

共用体的第一个应用——节省存储空间

姓名	性别	年龄	婚姻状况						婚姻状况标记
			未婚	已婚			离婚		
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量	

```
struct person
{
    char name[20];
    char sex;
    int  age;
    union maritalState marital;
    int marryFlag;
};
```

```
union maritalState
{
    int single;                          /*未婚*/
    struct marriedState married; /*已婚*/
    struct divorceState divorce; /*离婚*/
};
```

共用体的第一个应用——节省存储空间

姓名	性别	年龄	婚姻状况						婚姻状况标记
			未婚	已婚			离婚		
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量	

```
union maritalState
{
    int single;                    /*未婚*/
    struct marriedState married;   /*已婚*/
    struct divorceState divorce;   /*离婚*/
};
```

```
struct marriedState
{
  struct date marryDay;
  char spouseName[20];
  int  child;
};
```

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

```
struct divorceState
{
  struct date divorceDay;
  int  child;
};
```

为共用体添加标记字段

姓名	性别	年龄	婚姻状况						婚姻状况标记
			未婚	已婚			离婚		
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量	

```
struct person
{
    char name[20];
    char sex;
    int  age;
    union maritalState marital;
    int marryFlag; //婚姻状态标记字段
};
struct person p1;
```

每次对共用体的成员赋值时，程序负责改变标记字段的内容

共用体的一个主要问题：如何标记共用体中当前起作用的成员是哪一个？

```
if (p1.marryFlag == 1)
{
   //未婚
}
else if (p1.marryFlag == 2)
{
   //已婚
}
else
{
   //离婚
}
```

enum flag {SINGLE, MARRIED, DIVORCE};
typedef enum flag FLAG;

int marryFlag;可修改为：
FLAG marryFlag;

共用体的第二个应用——构造混合的数据结构

- 假设需要的数组元素是int型和float型数据的混合

```
typedef union
{
   int     i;
   float   f;
}NUMBER;
```

```
NUMBER array[100];

array[0].i = 10;

array[1].f = 3.14;
```

- 每个NUMBER类型的数组array的数组元素都有两个成员，既可以存储int型数据，也可以存储float型数据

小结

- 两种新的数据类型

结构体（struct）	共用体（union）
关系紧密且逻辑相关的多种不同类型的数据的集合	情形互斥但逻辑相关的多种不同类型的数据的集合
可以保存所有成员的值，用sizeof来计算占用内存的总字节数	内存重叠存储，每一瞬时只能保存一个成员，最后一次赋值的成员起作用
对所有成员初始化	~~只能对第一个成员初始化~~