step further

联合与枚举 **专题**

起步：

认知与体验（硬件、软件、程序与C语言）

进阶：

判断与推理（流程控制方法、语句）

抽象与封装（模块设计方法、函数）

表达与转换（基本操作、数据类型）

提高：

构造与访问（数组、指针、结构）

归纳与推广（程序设计的本质）

联合（union）类型

```
union myType
{
  int i;
  char c;
  double d;
};
```

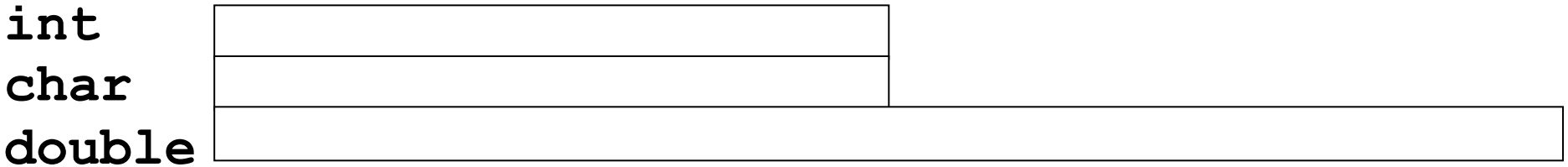
与结构类型类似，
联合类型由程序员构造而成，
构造时需要用到关键字union。

联合变量的初始化、成员的操作方式
也与结构变量类似。

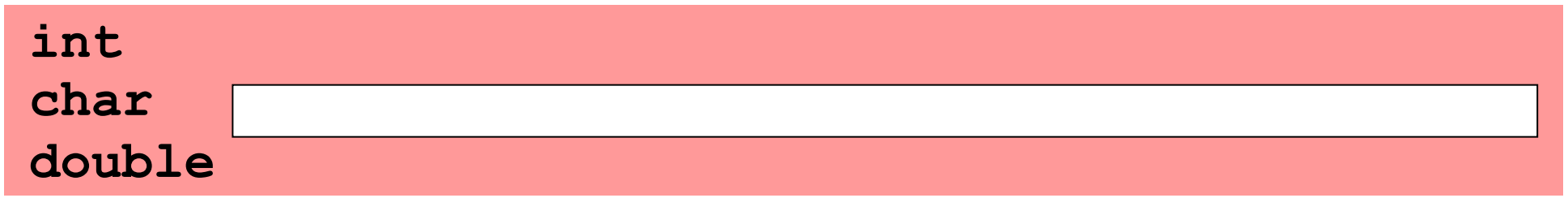
```
myType v;     //定义了一个myType类型的联合变量v
```

- 与结构变量不同的是，系统采用覆盖技术按需要占用内存单元最多的成员为联合变量分配内存

```
struct B
{   int i;
    char c;
    double d;
};
B b;
```




```
union A
{   int i;
    char c;
    double d;
};
A v;
```



- **对于上述联合变量v，在程序中可以分时操作其中不同数据类型的成员。比如，**

```
v.i = 12;              //以下只操作变量v的成员i
……
v.c = 'X';             //以下只操作变量v的成员c
……
v.d = 12.95;           //以下只操作变量v的成员d
……
```

- **当给一个联合变量的某成员赋值后，再访问该变量的另外一个成员，将得不到原来的值。比如，**

```
v.i = 12;
printf("%f", v.d);                    //不会输出12
```

- **即可以分时把v当作不同类型的变量来使用，但不可以同时把v当作不同类型的变量来使用。**

- 联合类型使程序呈现出某种程度的多态性。这种多态性的好处是：在提高程序灵活性的同时，可以实现多种数据共享内存空间。比如，

```
union Array
{
    int int_a[100];
    double dbl_a[100];
};
Array buffer;
… buffer.int_a …    //使用数组int_a，只有一半存储空间闲置
……
… buffer.dbl_a …    //使用数组dbl_a，没有存储空间闲置
```

如果不使用联合类型，例如，

```
int int_a[100];
double dbl_a[100];
… int_a …     //使用数组int_a（dbl_a所占的存储空间闲置）
……
… dbl_a …     //使用数组dbl_a（int_a所占的存储空间闲置）
……
```

枚举（enum）类型

```
enum Color {RED, YELLOW, BLUE};
enum Color c; //定义了一个变量 c
```

枚举类型由程序员构造而成，
构造时需要用到关键字enum

```
typedef enum Color Color;
Color c;
```

```
enum Weekday {SUN, MON, TUE, WED, THU, FRI, SAT};
typedef enum Weekday Weekday;
Weekday d1 = SUN, d2 = d1;    //定义了两个变量 d1 和 d2
```

```
d1 = 1; ×
d1 = RED; ×
```

```
int i = d1; √
d2 = (Weekday)(d1 + 1); √
```

- 当把一个枚举值赋值给一个整型变量时，枚举值会隐式转换成整型，而当把一个整数赋给枚举类型的变量时，系统不会将整数转换成枚举类型数据，这时候可以用显式类型转换。
  - 例如，

```
Weekday d;
d = (Weekday)(d + 1) ;
```

//如果写“d = d+1;”编译器会报错，因为d+1的结果为int类型

- **如果是联合类型的数组或 含联合类型成员的结构数组，则其每一个元素的类型可以不同。比如，**

```
enum Grade{UNDERGRAD, MASTER, PHD, FACULTY}; //职级
union Performanc
{  int nPaper;      //已发表论文篇数
   float gpa;       //GPA
}; //业绩类型
struct Person
{  char id[20];
   char name[20];
   enum Grade grd;
   union Performanc pfmc;    //职级不同  业绩类型可以不同
};
```

```
const int N = 800;                    //人员的个数
void input(Person prsn[ ], int num);
int main( )
{  Person prsn[N] = {{0,0, UNDERGRAD,0}};
   input(prsn, N);
   float maxgpa = 0.1;
   int maxMaster = 0;
   int maxPhd = 0;
   int maxFaculty = 0;
```

```
for(int i = 0; i < N; ++i)
{       if(prsn[i].grd == UNDERGRAD
                && prsn[i].pfmc.gpa > maxgpa)
            maxgpa = prsn[i].pfmc.gpa;
        if( prsn[i].grd == MASTER
                && prsn[i].pfmc.nPaper > maxMaster)
            maxMaster = prsn[i].pfmc.nPaper;
        if( prsn[i].grd == PHD
                && prsn[i].pfmc.nPaper > maxPhd)
            maxPhd = prsn[i].pfmc.nPaper;
        if( prsn[i].grd == FACULTY
                && prsn[i].pfmc.nPaper > maxFaculty)
            maxFaculty = prsn[i].pfmc.nPaper;
}
```

```
    for(int i = 0; i < N; ++i)
        if(prsn[i].grd == UNDERGRAD
                && prsn[i].pfmc.gpa == maxgpa)
                printf( "本科生获奖者: %s \n",  prsn[i].name);
    //...
    return 0;
}
```

```
void input(Person prsn[ ], int num)
{  int g = 0;
   for(int i = 0; i < num; ++i)
   {    printf( "输入人员的编号与姓名: \n");
        scanf("%s%s", prsn[i].id, prsn[i].name);
        printf( "输入0~3分别代表本科、硕士、博士生和教师: ");
        scanf("%d", &g);
        switch(g)
        {    case 0: prsn[i].grd = UNDERGRAD;
                     printf( "输入学分绩: ");
                     scanf("%f", &prsn[i].pfmc.gpa);
                     break;
             case 1: prsn[i].grd = MASTER;
                     printf( "输入已发表论文篇数: ");
                     scanf("%d", &prsn[i].pfmc.nPaper);
                     break;          …… ……
```

小结

- 联合是一种与结构类似的派生数据类型，与结构类型的不同点，仅在于存储方式（系统对联合类型的成员采用了覆盖存储技术），可以在实现多态性程序的同时节约内存空间，程序的多态性可以提高程序的可读性。
- 枚举也是一种派生数据类型，用来描述 值集是可以枚举的有限个整数 的数据

- 要求：
  - 了解以上概念和应用
  - 掌握
    - 联合和枚举的特点
    - 一个程序代码量≈200行
  - 继续保持良好的编程习惯