联合体的概念

联合体从表面上看与结构体非常类似，但是他们的本质完全不同。结构体每一个成员都有一个独立的内存空间， 但是联合体的每一个成员公用同一个内存空间因此联合体也被称为共同体。

语法：

```
union 联合体标签
{
        成员1 ;
        成员2 ;
        成员3 ;

};
```

联合体标签：用来区分不同类型的联合体

成员： 联合体内存的各个成员

```
// 声明联合体
union node
{
    int i ;
    char c ;
    short s ;
    double d ;
};


int main(int argc, char const *argv[])
{
    // 联合体变量定义
    union node data;

    printf("size:%ld\n" , sizeof(data));
    printf("i:%p\n" , &data.i );
    printf("c:%p\n" , &data.c );
    printf("s:%p\n" , &data.s );
    printf("d:%p\n" , &data.d );

    return 0;
}
```

注意：

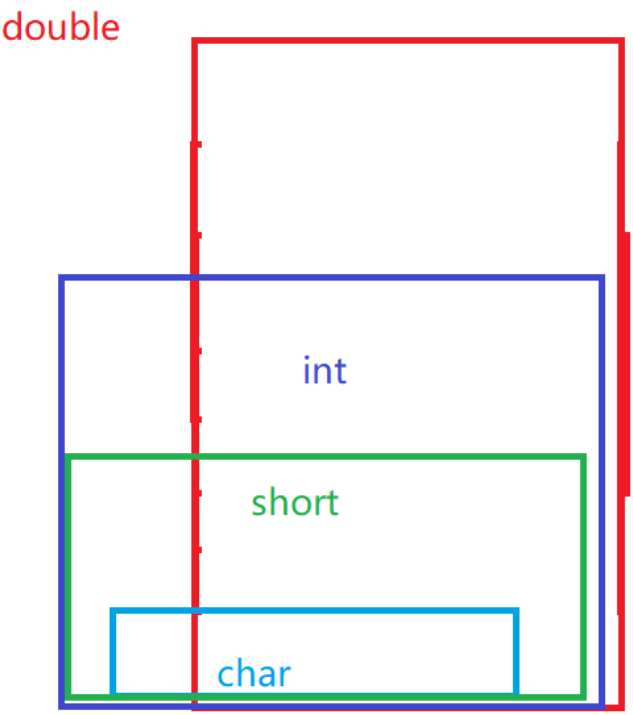
联合体的尺寸取决于，联合体中成员宽度最宽的。

联合体中所有成员的内存地址（首地址）是同一个

在同一时间内，有效的成员只有一个

当我们给联合体某一个成员赋值的时候其它成员则失效（数据无效）

```
// 声明联合体
typedef union node
{
    int i ;
    char c ;
    short s ;
    double d ;
}Node ;
```



联合体的使用

联合体一般来说比较少使用， 如果要用很可能会出现在结构体内部，用来描述一种互斥的数据。

示例：

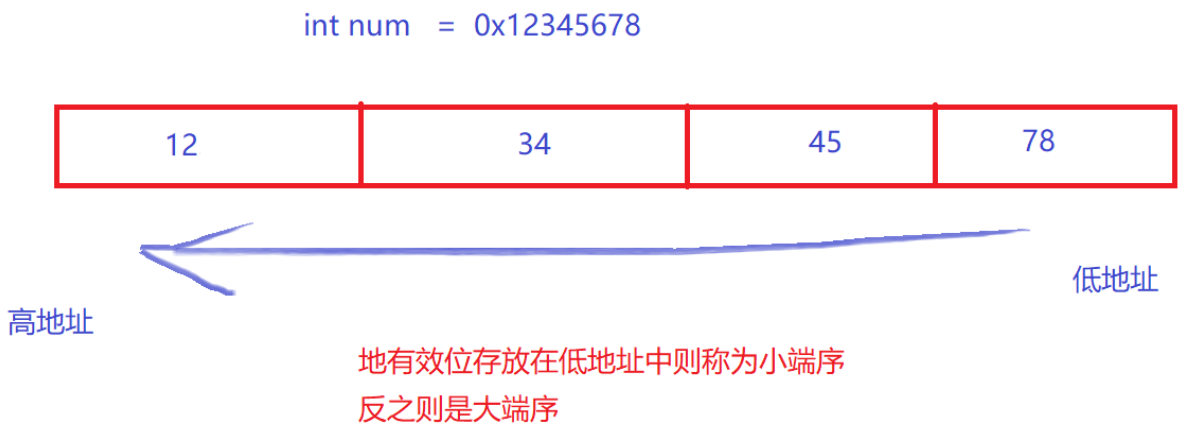
```
#include <stdio.h>
#include <string.h>

// 声明结构体
typedef struct demo
{
    char Nmae[32];
    char Type ;  // S 学生  T  老师  F  打饭阿姨  Q  清洁工
    union
    {
        float num ; // 成绩
        int i ; // 手抖值
```

```
13          char c ; // 清洁能力
14      }stat;
15  }Node ;
16
17
18
19
20  int main(int argc, char const *argv[])
21  {
22      Node data [3] ;
23
24
25      // 把"Even" 拷贝到 结构体数据则第0 位， 希望拷贝32 字节
26      strncpy(data[0].Nmae , "Even" , 32 );
27      data[0].Type = 'Q' ;
28      data[0].stat.c = 'A';
29
30      // 学生
31      strncpy((data+1)->Nmae , "Jacy" , 32 );
32      (data+1)->Type = 'S' ;
33      (data+1)->stat.num = 13.86 ;
34
35      // 阿姨
36      strncpy((data+2)->Nmae , "CuiHua" , 32 );
37      (data+2)->Type = 'F' ;
38      (data+2)->stat.i = 13 ;
39
40      for (size_t i = 0; i < 3 ; i++)
41      {
42          switch((data+i)->Type)
43          {
44              case 'Q':
45                  printf("当前输出为清洁工: \n 名字: %s\t人猿类型:%c\t清洁能力: %c\n",
46                      (data+i)->Nmae ,(data+i)->Type , (data+i)->stat.c );
47                  break ;
48              case 'S':
49                  printf("当前输出为学生: \n 名字: %s\t人猿类型:%c\t学习能力: %f\n",
50                      (data+i)->Nmae ,(data+i)->Type , (data+i)->stat.num );
51                  break ;
```

```
52              case 'F':
53                  printf("当前输出为打饭阿姨：\n 名字：%s\t人猿类型:%c\t抖肉能力：%d\n",
54                      (data+i)->Nmae ,(data+i)->Type , (data+i)->stat.i );
55                  break ;
56          }
57      }
58
59      return 0;
60  }
61
```

字节序



使用联合体检查当前系统是大端/小端?

枚举

概念： 枚举本质上一种范围受到限制的整型，比我们可以使用 0 - 3 表示4种状态（比如某个硬件的状态:打开、运行、暂停、关闭）

枚举常量列表:

语法:

```
1  // 声明枚举常量列表
2  enum  枚举标签 {  常量1 , 常量2 ,常量3 ,常量4 ,} ;
```

```
3
4 enum  MIC {  open , run , stop  , close } ;
```

使用：

```
1  #include <stdio.h>
2
3  // 声明枚举常量列表
4  enum  MIC {  open , run , stop  , close } ;
5
6  int main(int argc, char const *argv[])
7  {
8      enum  MIC stat = run ;
9
10     switch(stat)
11     {
12         case open:
13             printf("麦克风状态为打开：%d\n" , open);
14             break;
15         case run:
16             printf("麦克风状态为运行中：%d\n" , run);
17             break;
18         case stop:
19             printf("麦克风状态为暂停：%d\n" , stop);
20             break;
21         case close:
22             printf("麦克风状态为关闭：%d\n" , close);
23             break;
24     }
25
26     return 0;
27 }
28
```

枚举常量的值可以手动修改：

```
1  // 声明枚举常量列表
```

```
2 enum  MIC {  open , run = 5 , stop  , close } ;
```

总结：

默认情况下枚举常量实际它的值为整型值， 并首个元素默认为0

可以在声明枚 举常量列表的时候对给他进行赋值， 如果没有赋值则等于前一个常量值+1

在C语言中 枚举等价于整型，支持所有整型的操作

```
1 printf("%d\n" , run+stop);
```

枚举的作用提供了一个确定范围， 使用一个有意义的单词来表示一个整型数字提高代码的可读性

作业：

1. 使用枚举来完善周五的小作业
2. 继续完成周五小作业（优化-函数封装与调用-返回）
3. 使用不同的方法来求得计算机的字节序

预习：

头文件

宏定义

多文件编译