

# C语言——面向过程编程

## 单元五、结构体与链表

### 第一次课 结构体

#### 回忆枚举类型的定义

##### ① 创建枚举类型

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

##### ② 使用枚举类型

```
enum week a = MON;
```

##### ③ 用typedef简化类型名称

```
typedef enum week WEEK;
//简化后的用法
WEEK b = MON;
```

也可以在枚举类型创建时直接用typedef定义类型名称

```
typedef enum week//此处week名称可以省略
{
    MON, TUE, WED, THU, FRI, SAT, SUN
} WEEK;
```

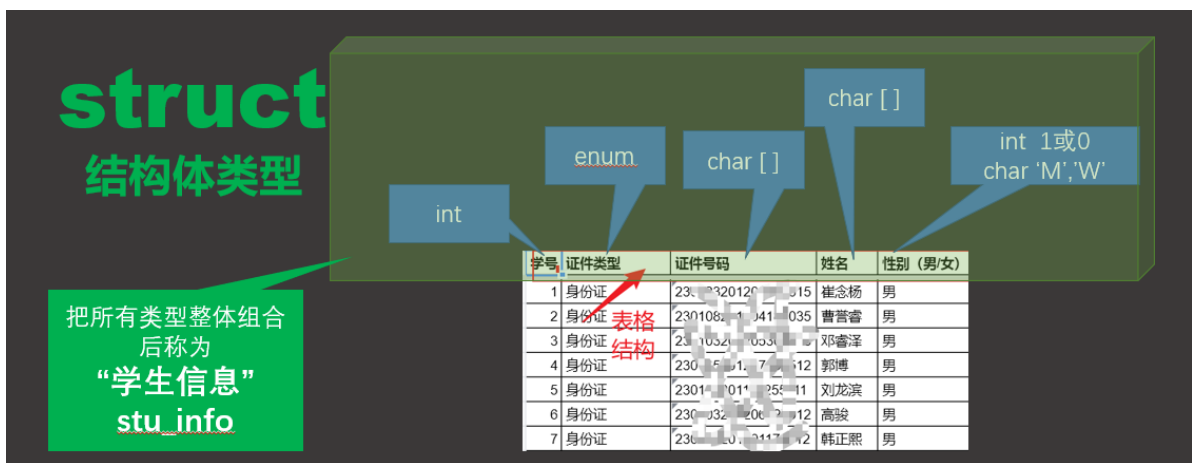
通过回忆可知：enum关键词引流程序员创建一种新的数据类型，只不过枚举类型的特点是：本质是一个整型，其次值是必须固定几个，由程序员自行行为每个值命一个常量名。和枚举enum类似的，可以让程序员自己定义类型的关键词还有：struct结构体类型 union 联合体或称共用体类型。

### 一、结构体定义

结构体想表达的是类似表格头的一种数据结构。

学号	证件类型	证件号码	姓名	性别（男/女）	年级	班级	联系电话	民族
1	身份证	230103201201010515	崔念杨	男	四年	二班	13100817858	汉族
2	身份证	23010820110411035	曹营睿	男	四年	二班	13903608310	汉族
3	身份证	23010320110531053	邓睿泽	男	四年	二班	13796039950	汉族
4	身份证	2301052011070612	郭博	男	四年	二班	15846638852	汉族
5	身份证	2301012011025111	刘龙滨	男	四年	二班	15145133377	汉族
6	身份证	2301032010062112	高骏	男	四年	二班	18608812623	汉族
7	身份证	2301012011070612	韩正熙	男	四年	二班	13263617277	汉族

它是一种把现有的数据类型，根据表格里面数据的类型组织成一个类型的组合体。



## ① 定义一个结构体

//结构体定义格式:

```
struct 结构体名
```

```
{
```

```
    类型 成员变量1;
```

```
    类型 成员变量2;
```

```
    ....
```

```
};
```

上面的表格学生就可以定义如下了:

//提前定义身份证件的类型为自定义枚举类型

```
enum type { 身份证, 学生证, 驾驶证 } ;
```

```
struct stu_info
```

```
{
```

```
    int stuNO; //学生编号
```

```
    enum type typeID; //证件类型
```

```
    char cardID[20]; //证件编号
```

```
    char name[20]; //姓名
```

```
    int sex; //性别
```

```
};
```

## ② 定义结构体类型的变量

```
//          类型名    变量名
```

```
struct stu_info    stu;
```

## ③ 创建结构体变量并赋初值

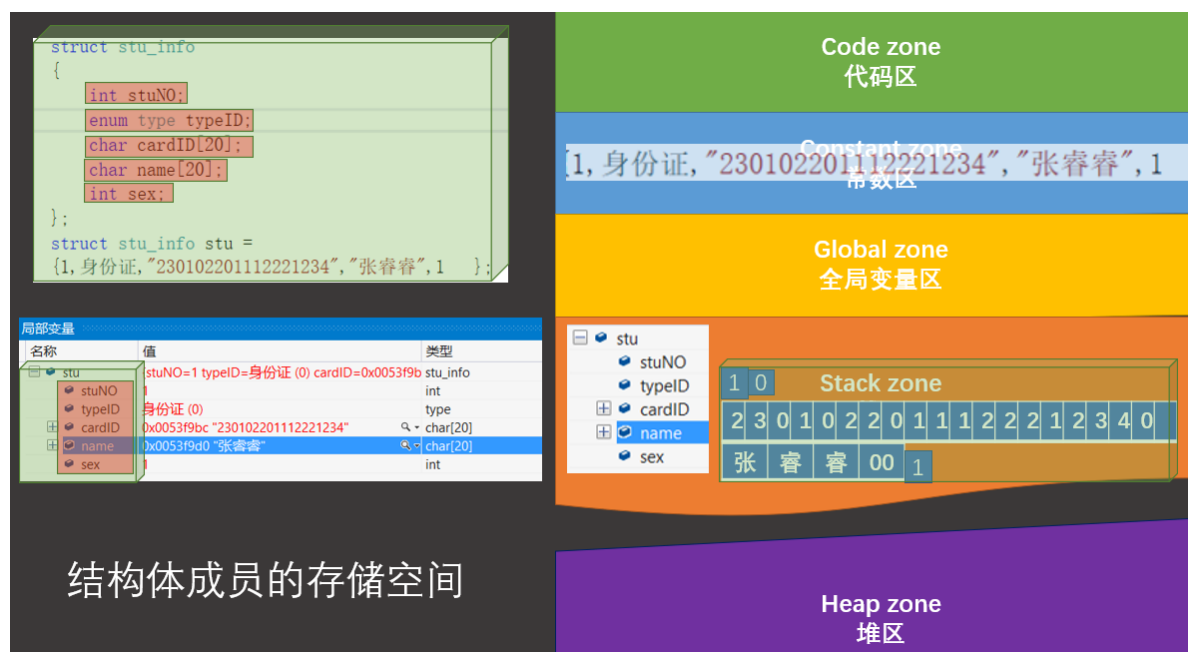
```
struct stu_info stu = {1, 身份证, "230102201112221234", "张睿睿", 1    };
```

#### ④ 存取结构体成员值 ——用 . 操作符

```
printf(" %d\n",stu.stuNO);
printf(" %d\n",stu.typeID);
printf(" %s\n",stu.cardID);
printf(" %s\n",stu.name);
printf(" %d\n",stu.sex);
```

```
C:\WINDOWS\system32\cmd.exe
1
0
230102201112221234
张睿睿
1
请按任意键继续. . .
```

#### ⑤ 结构体的内存原理



## 二、typedef命名结构体

使用 typedef 对结构体类型进行简化名称

```
enum type { 身份证, 学生证, 驾驶证, } ;
typedef struct stu_info //此处名称可以省略
{
    int stuNO;
    enum type typeID;
    char cardID[20];
    char name[20];
    int sex;
} STU_INFO;
```

【通常】定义完结构体类型后再定义一个结构体指针类型

```
typedef STU_INFO* P_STU_INFO;
```

【建议：】可以把以上的分别定义合体在一起。

```
enum type { 身份证, 学生证, 驾驶证, } ;
typedef struct stu_info //此处名称可以省略
{
    int stuNO;
    enum type typeID;
    char cardID[20];
    char name[20];
    int sex;
} STU_INFO, *P_STU_INFO; //为新的结构体类型定义类型别名, 为此类型指针定义类型别名
```

用新的类型名创建结构体变量

```
struct stu_info stu1 = {1, 身份证, "230102201112221234", "张睿睿", 1 } ; //原有方式
也可继续使用
STU_INFO stu2 = {1, 身份证, "230102201112221234", "张睿睿", 1 } ;
```

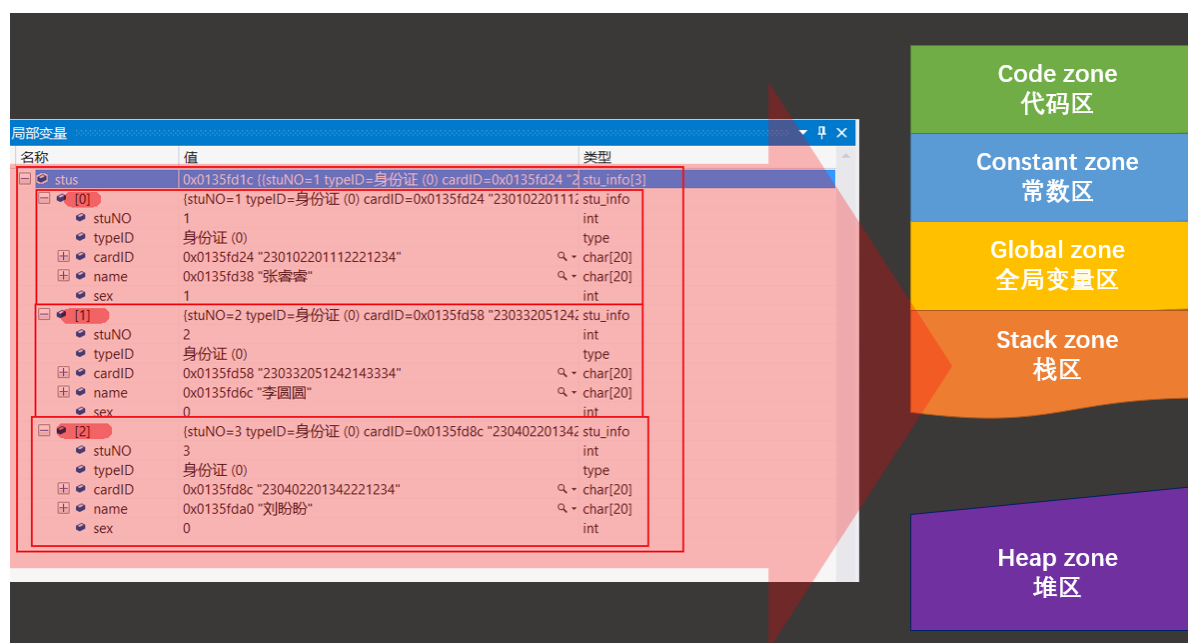
### 三、结构体数组

创建结构体数组，并赋初始值

```
STU_INFO stus[]={
    {1, 身份证, "230102201112221234", "张睿睿", 1 },
    {2, 身份证, "230332051242143334", "李圆圆", 0 },
    {3, 身份证, "230402201342221234", "刘盼盼", 0 },
};
```

调试观察存储形态

局部变量		
名称	值	类型
stus	0x0135fd1c {(stuNO=1 typeId=身份证 (0) cardID=0x0135fd24 "230102201112221234" stu_info[3]}	
[0]	{(stuNO=1 typeId=身份证 (0) cardID=0x0135fd24 "230102201112221234" stu_info[3]}	
stuNO	1	int
typeID	身份证 (0)	type
cardID	0x0135fd24 "230102201112221234"	char[20]
name	0x0135fd38 "张睿睿"	char[20]
sex	1	int
[1]	{(stuNO=2 typeId=身份证 (0) cardID=0x0135fd58 "230332051242143334" stu_info[3]}	
stuNO	2	int
typeID	身份证 (0)	type
cardID	0x0135fd58 "230332051242143334"	char[20]
name	0x0135fd6c "李圆圆"	char[20]
sex	0	int
[2]	{(stuNO=3 typeId=身份证 (0) cardID=0x0135fd8c "230402201342221234" stu_info[3]}	
stuNO	3	int
typeID	身份证 (0)	type
cardID	0x0135fd8c "230402201342221234"	char[20]
name	0x0135fda0 "刘盼盼"	char[20]
sex	0	int



## 四、结构体指针

### ① 创建结构体类型指针变量

```
STU_INFO stu={1,身份证,"230102201112221234","张睿睿",1};
//如下都可以定义结构体指针变量
struct stu_info * p=&stu;//使用原始类型定义
STU_INFO * p2=&stu;//使用类型别名定义
P_STU_INFO p3=&stu;//使用结构体指针类型别名定义
```

很明显：第三种方式更精简。

### ② 使用结构体指针操作结构体成员变量

#### 【方式一：】 (\*). 的方式

即：首先：(\*指针)得到结构体整体空间。再.取得成员空间

```
printf(" %d\n",(*p).stuNO);
printf(" %d\n",(*p).typeID);
printf(" %s\n",(*p).cardID);
printf(" %s\n",(*p).name);
printf(" %d\n",(*p).sex);
```

【方式二：】-> 运算符，一个箭头的形象感。代表：用指针指向结构体某成员的空间。

```
printf(" %d\n",p3->stuNO);
printf(" %d\n",p3->typeID);
printf(" %s\n",p3->cardID);
printf(" %s\n",p3->name);
printf(" %d\n",p3->sex);
```

很明显：当使用结构体指针变量时，第二种箭头方式更方便直观。

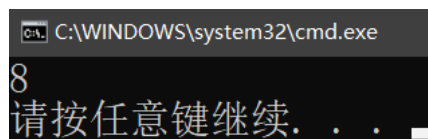
## 五、结构体的对齐

观察如下结构体它占用的空间字节数会是多少呢？

```
struct AA
{
    int a;
    short c;
    char b;
} ;
```

【错误的方式】：每个成员空间的累加 即  $4 + 2 + 1$  等于 7 。可用如下代码查看后

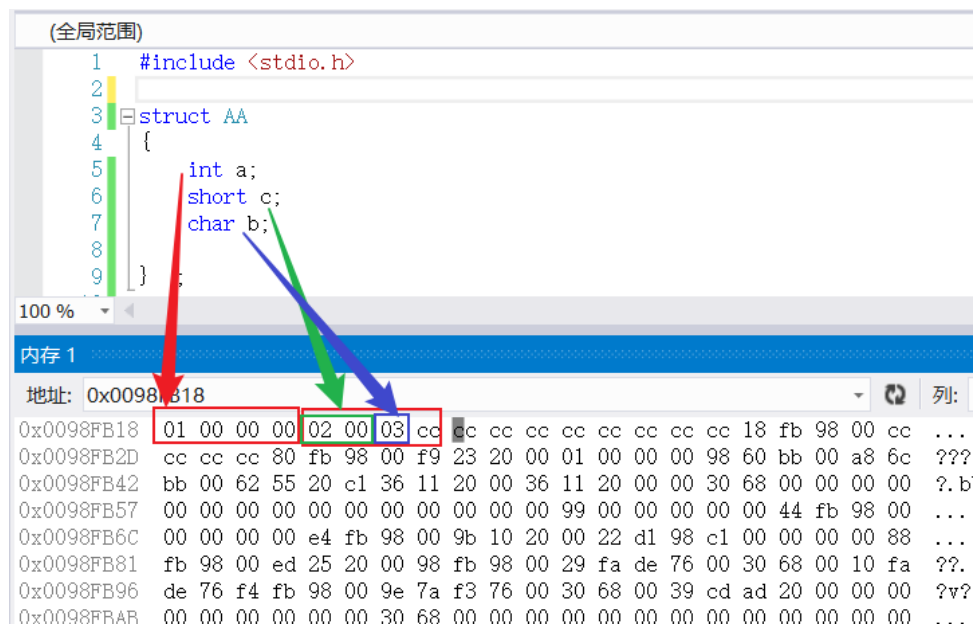
```
printf("%d\n",sizeof(struct AA));
```



为什么会是8呢。这个原因概括为“结构体的对齐”

所谓结构体对齐：就是以结构体成员中占字节数最多的为基准，采用两边存放的策略定义成员的空间结构。

```
struct AA *p,aa={ 1,2,3 };//创建struct AA 类型的变量，调试器观察其存储形态
p=&aa;
```



【观察上图：】整个结构体最大类型是int，因此它的存储策略是以int为基准长度，然后看后面的成员能否在4个字节内存放，如果不足以放下，就再开辟下4个字节。

【对比：如下两个结构体类型定义的区别】

```
struct HH
{
    int a;
    short c;
    char b;
    short d;
    char e;
    short f;
};

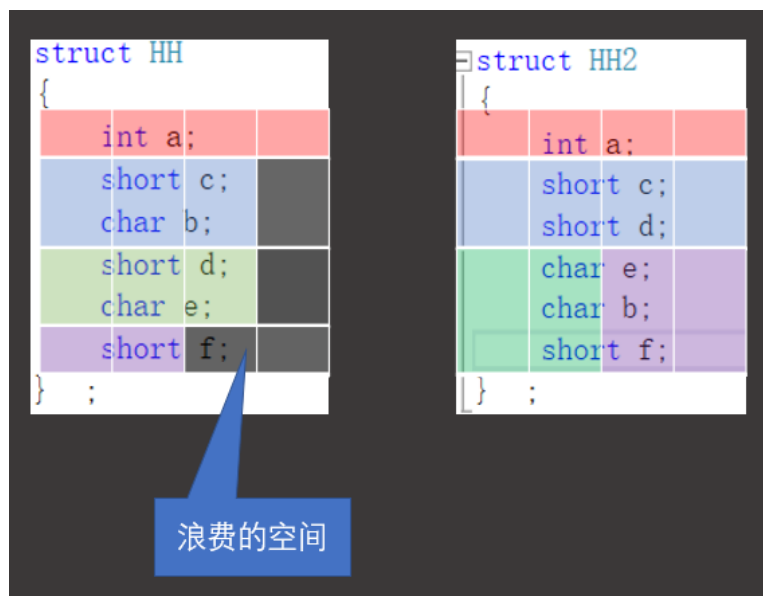
struct HH2
{
    int a;
    short c;
    short d;
    char e;
    char b;
    short f;
};
```

用如下代码测试：

```
printf("%d\n",sizeof(struct HH));
printf("%d\n",sizeof(struct HH2));
```

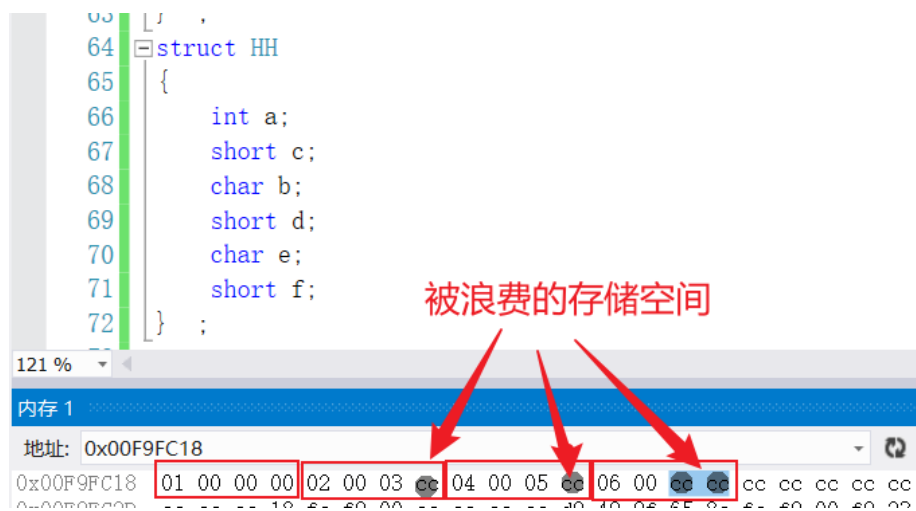
```
C:\WINDOWS\system32\cmd.exe
16
12
请按任意键继续. . .
```

【发现：】同样的成员，由于定义的顺序不同导致所占内存空间的大小不同。

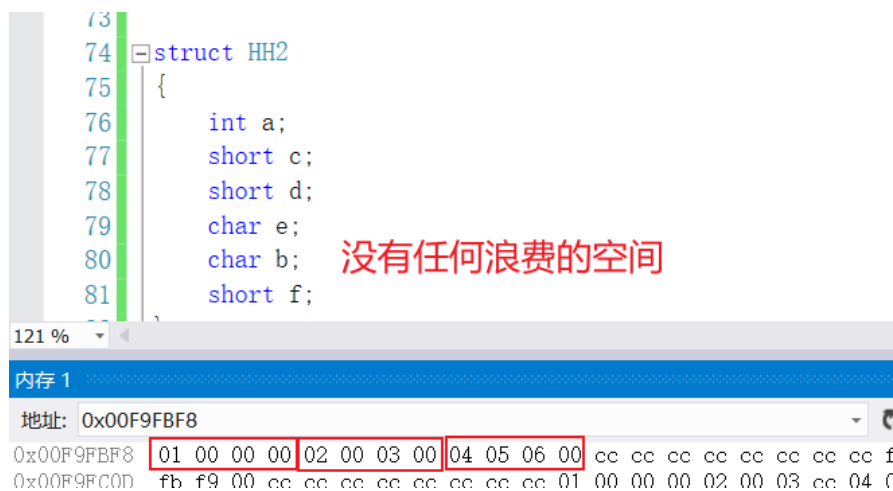


用如下代码调试查看内存状态：

```
struct HH *p,aa={ 1,2,3,4,5,6 };
struct HH2 *p2,bb={ 1,2,3,4,5,6 };
p=&aa;
p2=&bb;
printf("%d\n",sizeof(struct HH));
printf("%d\n",sizeof(struct HH2));
```



【HH的结构体空间——有浪费】



【HH2的结构体空间——合理无浪费】

**【对齐的启示：】**在定义结构体成员顺序时，需要考虑结构体对齐的机制，尽量减少不必要的空间浪费。

另外几个类型——数组与指针的对齐：

```
struct FF//空间长度为6
{
    char a[5];//数组整体不会作为基准长度，它就是4个同类型的变量，基准长度还是char的字节数
    char e;
};
```

```
struct FF2//空间长度为 12
{
    char a[5]; //数组长度为5，基准长度为4， 最后一个成员需要单独的存储单元。
    char *e;//此处为指针类型，32位系统占4个字节，因此基准长度为4
};
```

## 六、结构体与堆内存空间

观察如下代码：

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



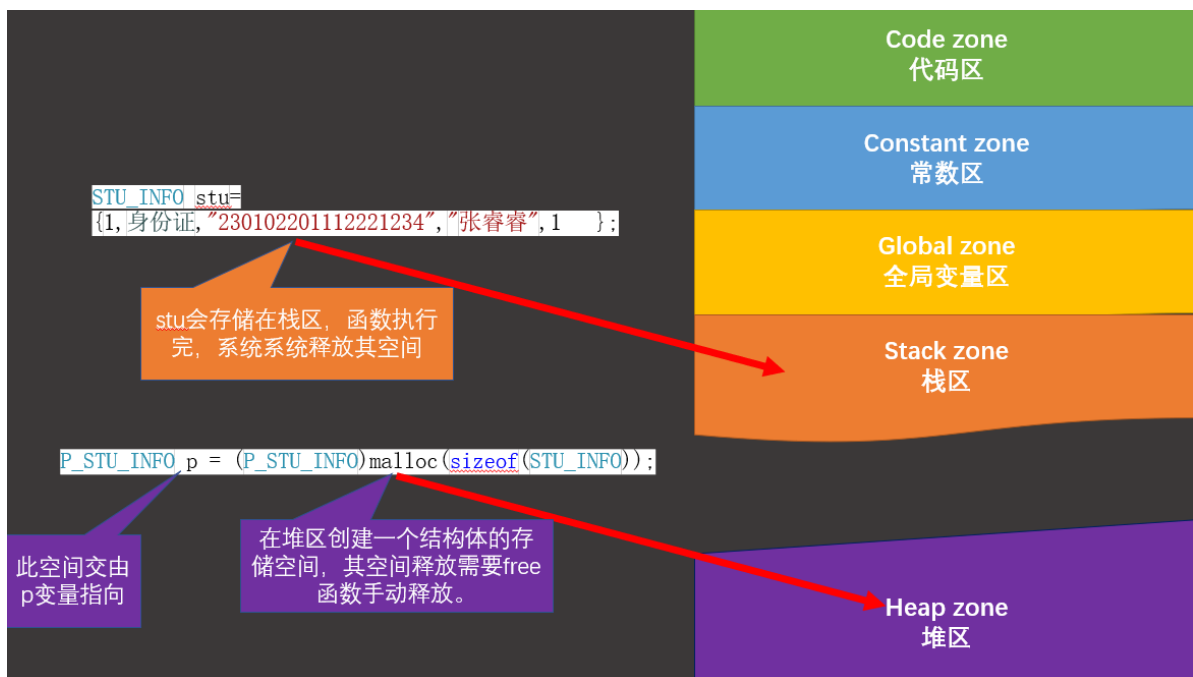
```

#include <string.h>
/*定义一个结构体*/
enum type { 身份证,学生证,驾驶证,} ;
typedef struct stu_info //此处名称可以省略
{
    int stuNO;
    enum type typeID;
    char cardID[20];
    char name[20];
    int sex;
} STU_INFO, * P_STU_INFO;
int main ()
{
    STU_INFO stu={1,身份证,"230102201112221234","张睿睿",1 };

    P_STU_INFO p = (P_STU_INFO)malloc(sizeof(STU_INFO));
    p->stuNO=1;
    p->typeID = 驾驶证;
    strcpy(p->cardID,"230102201112221234");//务必用此种方式,因为结构体成员cardID是数组
    strcpy(p->name,"李盼盼");//务必用此种方式,因为结构体成员name是数组
    p->sex = 0;

    free(p);//使用完结构体,记得释放堆空间
    return 0;
}

```



【建议：】结构体的存储用堆空间的方式，它不会因为所在函数执行完自动释放，只要能够保存其指针地址，就可以访问里面的数据。通常结构体的数据都是多函数访问的，因此用结构体变量(尤其是局部变量)存储数据会产生被释放的bug。导致其它函数无法访问结构体变量。

## 七、结构体位域定义

结构体位域：就是可以定义结构体某成员所占空间的字节数。比如：int默认占4个字节，可以用如下写法定义int只占几个01位。此种方式主要用于单片机嵌入式方面的开发，主要是单片机硬件资源有限。

```
typedef struct
{
    int a:2;//a变量占两个位的存储空间 但int会分配4个字节。
    int b:8;//b占8位,
    int c:2;//c占8位
    //到这里: a b c一共占12位 约等于2个字节,但是由于int占4个字节,其余空间也不会给别人的。
    int :0;//断开: 即跳到下一个4个字节
    int d:2;
} TT;
```

```
printf("%d\n",sizeof(TT));//打印的字节数为8
```

```
int main()
{
    TT t = {3,1,1} ;
    printf("%d\n",t.a);
    printf("%d\n",t.b);
    printf("%d\n",t.c);
    return 0;
}
```



## 八、结构体嵌套

即：在结构体的成员变量中，不仅仅是基础数据类型，也可以有其它的结构体。

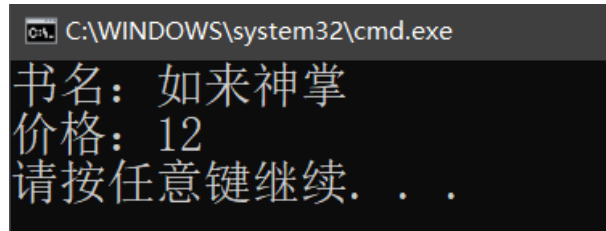
```
#include <stdio.h>

typedef struct book
{
    int price;
    char* name;
} Book;

typedef struct
{
    int age;
    char *name;
    Book book;//结构体嵌套
} Student;

int main ()
{
    Student s={ 20,"张三",{12,"如来神掌"} };
```

```
printf("书名: %s\n", s.book.name); //得到嵌套结构体内的成员, 继续 . 运算  
printf("价格: %d\n", s.book.price);  
return 0;  
}
```



C:\WINDOWS\system32\cmd.exe  
书名: 如来神掌  
价格: 12  
请按任意键继续. . .