复杂构造数据类型

结构体的引出

提出问题：学生信息包括：学号、姓名、年龄、3门课的成绩和总成绩，编程实现以下功能：(1) 输入3个学生的信息；(2) 计算每个学生的总成绩并输出。

分析： 首先可以把学生信息用表格的形式表现出来

学号	姓名	年龄	英语	数学	C语言	总成绩
2009001	赵岩	18	86	90	82	258
2009002	王洋	19	92	88	89	269
2009003	李玲	18	80	85	79	244

根据已有知识，我们可能需要分别定义以下5个数组：

```
int num[3];          //学号用一维整型数组存放
char name[3][10];    //姓名用二维字符数组存放
int age[3];          //年龄用一维整型数组存放
int score[3][3];     //3门课成绩用二维整型数组存放
int total[3];        //总成绩用一维整型数组存放
```

将同一个学生信息存储在了不同变量中，割裂了信息之间的联系

希望用一个变量来存储一条学生信息，那么如何存储如下一条学生信息？

2009001	赵岩	18	86	90	82	258

可否使用数组？ 不可以

如何将不同数组类型的数据组织在一起呢？ 使用结构体

结构体的定义

- 结构体是由一些逻辑相关，但数据类型不同的分量组成的一组数据。
- 这些分量称为“成员”，每一个成员可以是一个基本数据类型或者又是一个构造类型。
- 结构体类型的定义如下：

```
struct   结构体标签名
{
    类型名1   成员名表1;
    类型名2   成员名表2;
       :          :
    类型名2   成员名表n;
};
```

关键字

用户定义的标识符

成员表列

注意不要忘了分号

```
struct  student
{ int   num;
   char  name[10];
   int  age;
   int  score[3];
   int  total;
};
```

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

```
struct stud
{   int num;
    char name[10];
    struct date birthday;
    float s1, s2, s3;
    float total;
};
```

- 说明：
  - 结构体成员的类型可以是基本类型、数组、指针或已说明过的结构体类型等。
  - 每个“成员列表”都可以包含多个相同类型的成员名，它们之间用逗号隔开。
  - 定义了结构体类型，仅仅是定义了数据的组织形式，创立了一种数据类型，但并不会为这种结构体类型分配内存空间，只有定义了结构体变量，才会为变量分配空间。

- 结构体类型也可以嵌套定义

例：
```
struct  date
{
      int  year ;
      int  month ;
      int   day ;
} ;
struct  stud
{
    char   name[10] ;
    struct  date   birthday ;
    int   s1 , s2 ;
} ;
```

或：
```
struct  stud
{
    char   name[10] ;
    struct  date
    {
        int  year ;
        int  month ;
        int   day ;
    } birthday ;
    int   s1 , s2 ;
} ;
```

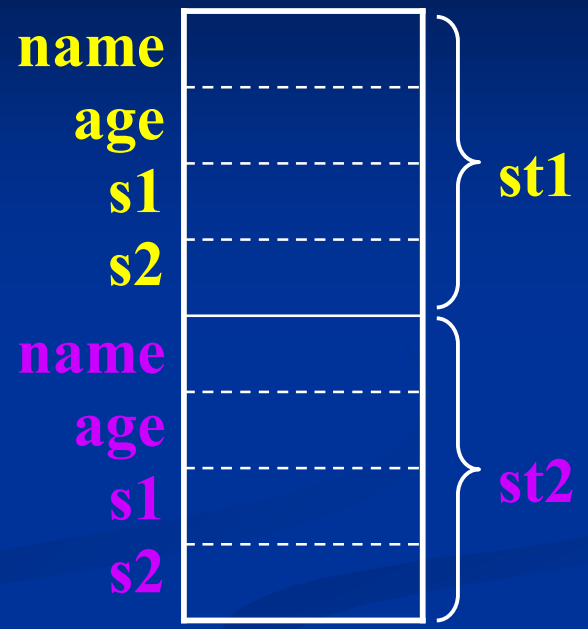
结构体变量的定义方法

- 先定义结构体类型, 再定义变量

```
struct student
{    char name[10];
     int age;
     int s1, s2;
};
struct student st1, st2;
```

结构体类型定义

结构体变量定义



- 内存中结构体变量占有一片连续的存储单元, 其占用的字节数可用sizeof 运算符算出:

```
printf("%d\n" , sizeof(struct   student) ) ;
printf("%d\n" , sizeof(st1) ) ;
```

- 结构体变量st1和st2理论上都需要22个字节(VC)的存储空间。

- 定义结构体类型同时定义变量

```
struct  student
{  char  name[10] ;
   int  age ;
   int  s1 , s2 ;
} st1, st2 ;
```

定义结构体变量

- 直接定义结构体变量

```
struct
{  char  name[10] ;
   int  age ;
   int  s1 , s2 ;
} st1, st2;
```

注意:这里没有结构体类型名
这种方式有时使用并不方便
因此不建议大家采用

结构体变量的引用

- 一般对结构体变量的使用，如赋值、输入、输出、运算等都是对结构体变量的成员进行操作。
- 结构体变量成员的表示格式：结构体变量名. 成员名
- 说明：圆点“.”称为结构体成员运算符，它的运算优先级是最高的。可以把“结构体变量名.成员名”看成一个整体，像简单变量一样使用。

```
struct   student
 {  char   name[10];
    int   age;
    int   s1, s2;
 };
```

```
struct  student   st1;
strcpy(st1. name , “Mary”);
st1. age = 21;
st1. s1 = 78;
st1. s2 = 86;
```

- 说明
  - 结构体变量通常不能整体使用，不能整体输入、输出，只能对单个成员分别引用。但当结构体变量作为函数参数或赋初值时，可以整体输入；或者两个相同类型的结构体变量，如果一个已经赋值，可以对另外一个整体赋值。

    例：如果s1和s2类型相同且s1已被赋值，则可以：s2 = s1;
  - 如果成员本身又属于一个结构体类型，则这个成员也不能整体赋值，要用若干个成员运算符从外层到内层引用，如前面的birthday成员，其本身又是date类型的变量，则需：

    stu1.birthday.month=4;　stu1.birthday.day=5;
  - 可以引用结构体变量成员的地址，也可以引用结构体变量的地址，要区分清楚。如：

    printf("%x", &stu1.num);　　printf("%x", &stu1);

结构体变量的初始化

```
struct  student
{  char   name[10] ;
    int   age ;
    int   s1 , s2 ;
}st1={“Mary”, 21, 78, 86};
```

初始化, 正确

```
struct  stud
{   char   name[10] ;
   struct  date   birthday ;
   int   s1 , s2 ;
 } ;
struct  stud  st2={ “John” , {1980 , 11 , 23} , 89 , 95 } ;
```

初始化, 正确

```
struct  student
{   char   name[10] ;
    int   age ;
    int   s1 , s2 ;
} ;
struct  student   st1;
st1={“Mary”, 21, 78, 86} ;
```

这是赋值, 错误
C不允许这么做

结构体变量的输入、输出

- C语言不允许结构体变量整体进行输入和输出，只能对结构体变量的成员进行输入和输出。

例：

```
gets( st1. name ) ;
scanf( “%d%d%d”, &st1. birthday. year ,
        &st1. birthday. month , &st1. birthday. day ) ;
scanf ( “%d%d%d”, &st1. age , &st1. s1 , &st1. s2 ) ;
puts( s1. name ) ;
printf( “%4d”, st1. age );
printf( “%d .%d .%d”, st1. birthday. year ,
        st1. birthday. month , st1. birthday. day ) ;
printf(“%4d %4d\n”, st1. s1 , st1. s2 ) ;
```

结构体数组的引出

- 一个结构体变量只能存放一个学生的信息，对于多个学生的信息，可以使用一个结构体数组来存放，结构体数组的每个元素是一个结构体类型的变量。
- 定义结构体数组的方法与定义普通数组的方法类似:

  结构体类型　数组名[数组的长度];

结构体数组的定义及使用

(1) 先定义结构体类型，再定义结构体数组：

```
struct   student
{   char   name[10] ;
    int   age ;
    int   s1 , s2 ;
} ;
struct  student  st[6];
```

(2) 定义结构体类型的同时定义结构体数组：

```
struct   student
{    char   name[10] ;
     int   age ;
     int   s1 , s2 ;
 } st[6] ;
```

(3) 直接定义结构体数组

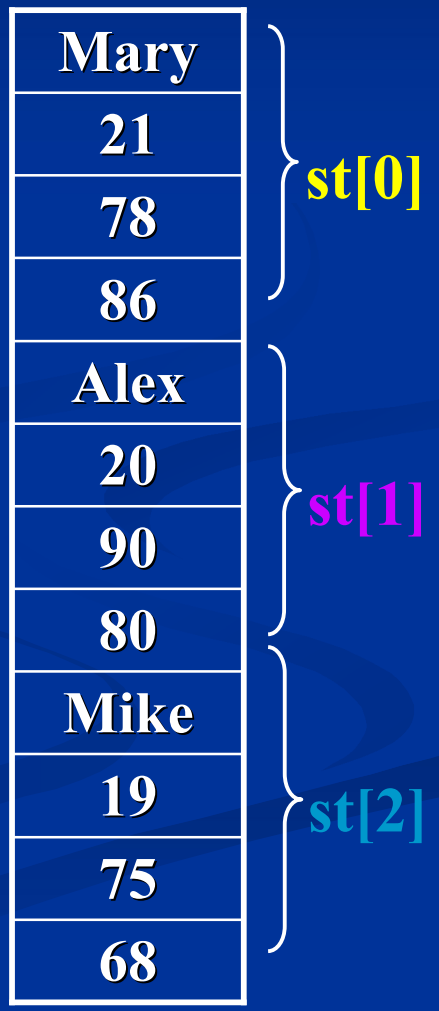
```
struct
{   char   name[10] ;
    int   age ;
    int   s1 , s2 ;
}  st[6];
```

不提倡使用该方法

结构体数组的初始化

- 和数组的初始化类似，将每个数组元素的数据用花括号{ }括起来。

```
struct   student
{    char   name[10] ;
     int   age ;
     int   s1 , s2 ;
} ;
struct  student  st[3]={ {“Mary”, 21, 78, 86},
                         {“Alex”,   20, 90, 80},
                         {“Mike”,  19, 75, 68}
                       };
```



结构体数组的引用

- 引用某个数组元素的成员

  例: puts(st[0].name); printf("%d,%d", st[1].age, st[1].s1);

- 数组元素之间可以整体赋值也可以将一个元素赋给一个相同类型的结构体变量

  例： struct student st[3]={{"Mary",21,78,86}, {"Alex", ...}} ;

  ```
  struct student x ;
  st[2] = st[0] ;
  x = st[1] ;
  ```

- 输入和输出操作只能对数组元素的成员进行

- 例：按成绩对学生信息进行从高到底的排序

```
#include <stdio.h>
#define N 30
struct stud
{    int  n;
     char  name[10];
     int  s;
};
void  input(struct stud  a[ ])
{  int  i ;
   for ( i=0 ; i<N ; i++)
      scanf("%d%s%d", &a[i].n, a[i].name, &a[i].s) ;
}
void  output(struct stud  a[ ])
{   int  i ;
   for ( i=0 ; i<N ; i++)
      printf("%4d %10s %4d",  a[i].n, a[i].name, a[i].s) ;
}
```

注意a[i].name前不加&,因name是数组名，因用%s，输入时名字不能加空格

```
void   sort(struct  stud   a[ ] )
{   int  i , j ;
    struct stud   temp;
    for ( i=0 ; i<N−1 ; i++)
        for ( j=i+1 ; j<N ; j++)
            if ( a[i].s<a[j].s )
            {  temp=a[i] ; a[i]=a[j] ; a[j]=temp ;}
}
int main(void)
{
   struct  stud  st[N];
   input(st);
   sort(st);
   output(st);
   return 0;
}
```

注意进行比较的是元素a[i]和a[j]的成绩成员s, 但进行交换的是元素a[i]和a[j]

指向结构体变量的指针

- 结构体指针变量的定义形式

  struct 结构体名 *结构体指针变量名

例：

```
struct stu
{    int num;
     char name[12];
     int score;
};
struct stu  a,*p;
p=&a;
a.num = 201001;
```

结构体变量的成员是通过点操作符直接访问的，也可以通过指向结构体的指针变量间接访问，方法如下：

① (*结构体指针变量名) . 成员名

```
(*p).num=2009003;
```

② 结构体指针变量名 -> 成员名

```
p->score=89;
```

->运算符（也称箭头运算符），其左侧的操作数必须是一个指向结构体的指针

指向结构体数组的指针

```
struct stu
{
    int num;
    char name[12];
    int score;
};
struct stu  a[10],*p;
```

- 结构体指针变量可以指向一个结构体数组，这时结构体指针变量的值是结构体数组的首地址。

  例如：p=a;

- 结构体指针变量也可指向结构体数组中的一个元素，这时结构体指针变量的值是该数组元素的地址。

  例如：p=&a[5];

例：用指向结构体数组的指针变量输出数组中数据。

```
#include<stdio.h>
struct stu
{    int num;
     char name[12];
     int score;
} st[3] = { {2009001, "赵岩", 86},
            {2009002, "王洋", 92},
            {2009003, "李玲", 78}
          };
int main(void)
{    struct stu *ps;
     for( ps=st; ps<st+3; ps++ )
        printf("%d, %s, %d\n", ps->num, ps->name, ps->score);
     return 0;
}
```

结构体变量作为函数参数

- 函数实参和形参都用结构体变量，参数之间为值传递，实参结构体变量各成员的值依次传给形参结构体变量。

返回结构体类型值的函数

- 定义格式:

  结构体类型名　函数名(形参表)

  {

  　　函数体 ;

  }

- 例 :　struct　student　 func(int　x, float　y)

  　　{　函数体 ; }

共用体

- 所谓“共用体”类型, 是指使几个不同类型的变量共同占用同一段内存单元。即：共用体的所有成员引用的是内存中相同的位置。
- 当你想在不同的时刻把不同的东西存储于同一个位置时，可以使用共用体。
- 共用体类型的定义如下：

```
union 共用体名
{
        数据类型    成员名1;
        数据类型    成员名2;
          :
        数据类型    成员名n;
};
```

例：

```
union  data
{   int     i;
    char   ch;
    float   f;
};
```

共用体变量的定义方法

- 定义共用体类型的变量：形式上也有3种，同结构体类型

1、先定义类型, 再定义变量

```
union  student
{  char   name[10] ;
   int   age ;
   float   s;
};
union student   st1, st2 ;
```

2、定义类型同时定义变量

```
union  student
{  char   name[10] ;
   int   age ;
   float   s;
} st1, st2 ;
```

3、直接定义共用体变量

```
union
{   char   name[10] ;
    int   age ;
    float   s ;
} st1, st2 ;
```

共用体变量的引用

- 对共用体变量的赋值和使用都只能是对变量的成员进行。
- 共用体变量的引用：共用体变量.成员名

例：

```
union  student
{
    char   name[10] ;
    int   age ;
    float   s;
};
```

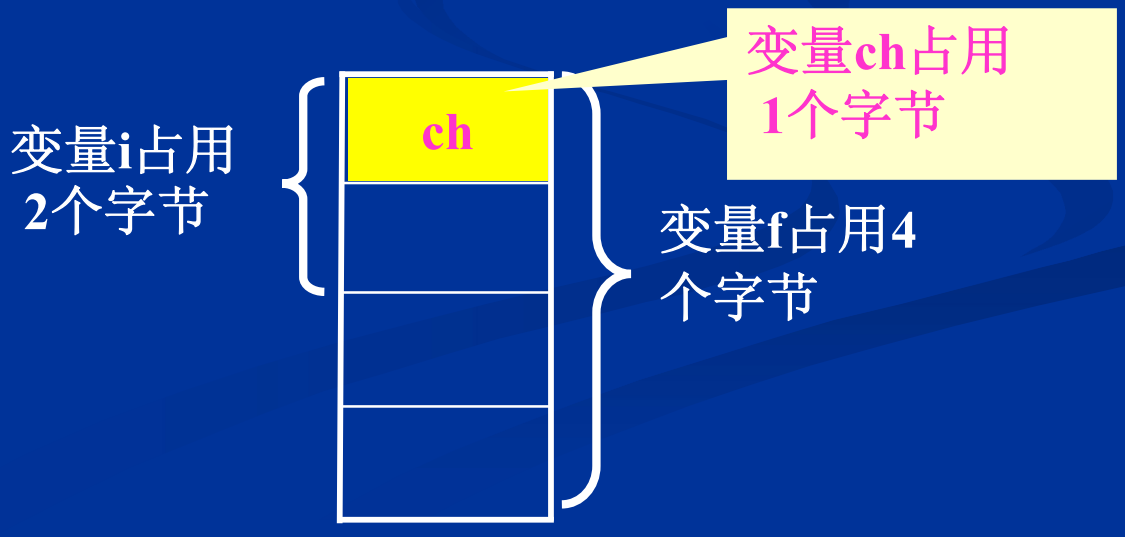
```
union student   st1;
strcpy(st1.name, “zhang”);
st1.age = 20;
st1.s = 78.5;
```

共用体变量的特点

- 共用体变量所占内存的长度等于最长成员的长度。而不是各成员的长度之和，这一点不同于结构体。
  - 例：test变量占4个字节。
- 每一瞬时只能存放其中的一个成员，而不是同时存放几种，且只有最后一个存放的成员值有效，其他成员将失去原值。
  - 例：依次执行 test.ch = 'A'; test.f = 3.5; test.i = 2; 则最后只有test.i有效。

例：

```
union  data
{  short   i;
   char    ch;
   float   f;
}test;
```



- 共用体变量可以初始化，但必须给第1个成员初始值，而且该值必须位于一对花括号中。（C99可以给任意成员初始值）
- 不能对共用体变量名整体赋值，不能引用变量名来输出一个值，只能引用它的某个成员。

例：

```
union data
{
        int a;
        float b;
        char c[8];
} test = {5};
```

相当于：test.a = 5;

例：

```
test = 3.2;   错
printf("%f", test);   错
test.b = 3.2 ;   对
printf("%f", test.b);   对
```

- 共用体变量的地址和它的成员地址都是同一地址。即&st1和st1.name、&st1.age、&st1.s的起始地址都是一样的。
- 共用体变量不能作为函数的参数，也不能作为函数返回值。但可以使用指向共用体变量的指针。
- 可以定义共用体数组。
- 共用体类型可以出现在结构体中，共用体成员也可以是结构体类型。

例：
```
struct member
{   int       num;
    char      name[20];
    union  {
           int    class;
           char  position[10];
    }catagory;
}mem[20];
```

整体是一个结构体类型；其中第3项成员是个共用体类型的变量。

枚举类型

- 引出：在实际问题中，有些变量的取值被限定在一个有限的范围内。例如，一个星期内只有七天，一年只有12个月等等。为此，C语言提供了一种称为“枚举”的类型。
- 概念：枚举是指将变量的所有取值一一列举出来，变量的取值只限于列举出来的值的范围，且这些值为符号常量。该变量称之为枚举类型变量，所列举的值叫做枚举元素（枚举常量）。
- 定义形式如下：

  enum 枚举名{枚举元素1，枚举元素2，．．．，枚举元素n };

  例： **enum weekday{sun, mon, tue, wed, thu, fri, sat};**

枚举类型变量的定义方法

- 定义枚举类型的变量：形式上也有3种，同结构体类型
- 先定义类型, 再定义变量

```
enum   weekday {sun, mon, tue, wed, thu, fri, sat };
enum  weekday  day;
```

- 定义类型同时定义变量

```
enum  weekday {sun, mon, tue, wed, thu, fri, sat } day;
```

- 直接定义变量

```
enum  {sun, mon, tue, wed, thu, fri, sat } day;
```

使用枚举变量需注意的问题

- 声明为枚举类型的变量实际上是整数类型。枚举类型本质上是一种基本数据类型，而不是构造类型。
- 枚举元素是常量，不是变量，不能在定义以外的任何位置对它们赋值，如sun=5是错误的。
- C语言中枚举元素按常量处理，它们是有值的。它们的值是系统按其定义顺序自动赋予的 0、1、2、3、……。
- 当然，枚举元素的值也可以改变，但必须在定义时指定。
  例： enum weekday {sun=7,mon=1,tue,wd,thu,fri,dat };
  如果枚举元素定义时未明确指定值，则它的值就比前面一个枚举元素的值大1。

- 枚举类型数据可以进行关系运算。枚举元素的比较规则是：比较其定义时的顺序号。例如：

```
enum weekday {Sun=7,Mon=1,Tue,Wed,Thu, Fri, Sat};
enum weekday  day=Tue;
if ( day==Mon) …
if ( day>mon  && day<fri ) ……
```

- 枚举变量取值只能是所列举的枚举元素，不能直接赋予一个整数值，例如：

```
enum weekday {Sun, Mon, Tue, Wed, Thu, Fri, Sat};
enum weekday a, b, c;
a=sun;   b=mon;   //正确
c=2;     //错误
```

小结

- 掌握结构体的定义和引用
- 掌握结构体变量所占字节大小
- 掌握指向结构体变量的指针变量的用法
- 掌握共用体的特点及所占字体大小
- 掌握枚举类型的特点


Class is over