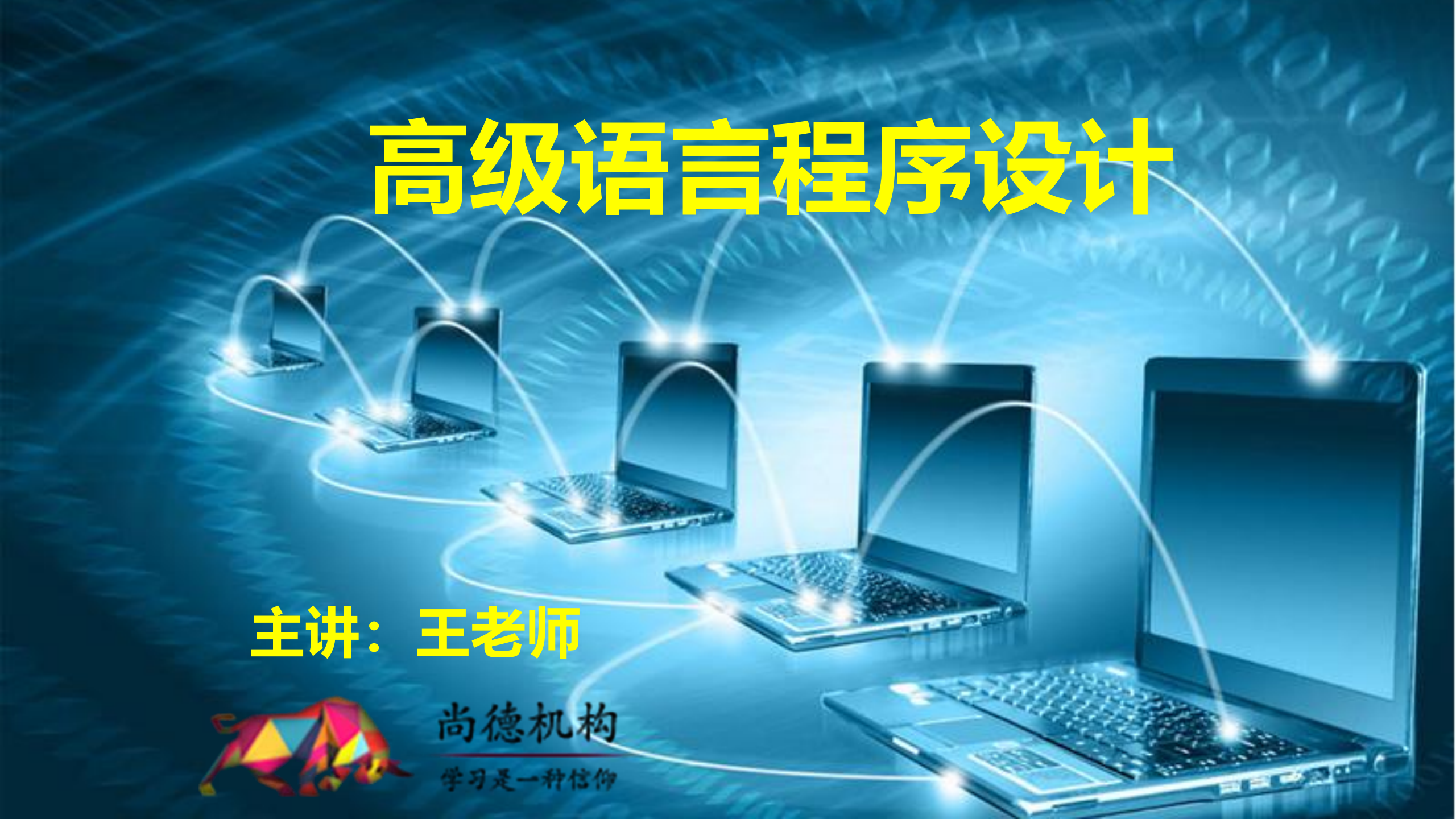
高级语言程序设计



主讲：王老师

尚德机构

学习是一种信仰

第八章　结构体类型和自定义类型

本章内容

- 结构体类型定义
- 结构体类型变量
- 结构体类型数组
- 结构体类型指针
- 结构体类型的程序设计实例
- 自定义类型

结构体类型和自定义类型

C语言提供了一种构造类型，即允许用户根据需要利用已有的数据类型自定义的一种数据类型，它由若干个数据类型相同或不同的数据项组合而成。为了存放类型不同的相关数据项，C语言提供了另一种构造类型，即结构体类型。

第一节 结构体类型定义

结构体类型定义的一般形式：

```
struct   结构体类型名                    /*struct是结构体类型的关键字*/
{
    数据类型符  成员名1;
    数据类型符  成员名2;
    …
    数据类型符  成员名n;
};                                  /*此行分号不能缺少*/
```

“结构体类型名”是用户命名的标识符；“数据类型符”可以是基本数据类型说明符，也可以是已定义的某种结构体类型；“成员名”可以是标识符、标识符[长度]、*标识符。

结构体类型定义

说明：

1、struct是结构体类型的关键字，“结构体类型名”是由用户自行命名，应遵循标识符的命名规则。

2、一对花括号括起来的部分是结构体类型的成员说明表，一个结构体类型可以包含任意个成员，这些成员可以是任何类型，包括各种基本数据类型和构造类型。

3、分号是结构体类型定义语句的结束标志，不能省略。

4、结构体类型定义的位置一般在程序开头，所有函数之前，此时所有函数都可以利用它来定义变量；若结构体类型定义在函数内部，则只能在本函数内部使用。

5、结构体类型仅是一种数据类型，系统不会为其成员分配内存。

课堂练习

1、结构型变量占用内存的字节数是（　　）

A. 各成员占用内存字节数之和

B. 第一个成员占用的内存字节数

C. 占用内存最大成员所需的字节数

D. 最后一个成员占用的内存字节数

课堂练习

1、结构型变量占用内存的字节数是（　　）

A. 各成员占用内存字节数之和

B. 第一个成员占用的内存字节数

C. 占用内存最大成员所需的字节数

D. 最后一个成员占用的内存字节数

答案：A

解析：定义结构体类型变量，系统才为其分配内存，所分配的内存字节数等于该结构体类型所有成员占用的字节数之和。

结构体类型变量的定义和初始化

用户自定义的结构体类型，与系统定义的基本数据类型（如int、char等）一样，可用来定义该类型的变量。

定义结构体类型变量的三种方法：

1、先定义结构体类型、后定义结构体类型的变量（间接定义法）

定义结构体类型变量的例子。 程序段代码如下：

```
struct std_info                    /*定义结构体类型*/
{
    int no;
    char name[10];
    char sex;
    int score[3];
};
…
struct std_info a,b;               /*定义结构体类型的变量 a 和 b*/
```

结构体类型变量的定义和初始化

2、在定义结构体类型的同时，定义结构体类型变量（直接定义法）

定义结构体类型的同时定义结构体类型变量并赋初值的例子。

程序段代码如下：

```
struct std_info                          /*定义结构体类型*/
{
    int no;
    char name[10];
    char sex;
    int score[3];
}a={1,"ZhangYi",'F',{80,78,88}};
                                         /*定义结构体类型的同时定义结构体类型的变量并赋初值*/
…
struct std_info b={2,"LiYang",'M',{89,91,85}};   /*该语句是正确的，因为前面已定义了结构体类型
                                                   std_info*/
```

结构体类型变量的定义和初始化

3、省略结构体类型名，直接定义结构体类型变量

```
struct
{ …
}a,b;
```

第二种和第三种定义结构体类型变量的一般格式：

```
struct [结构体类型名]
{
    …
}结构体类型变量表;
```

结构体类型变量初始化的格式与一维数组相似。

结构体类型变量={初值表};

★声明结构体类型的同时定义结构体变量

```
struct   student
{ int num;
   char  name[20];
   char sex;
   int age;
   float score;
   char addr[30];
} stu1,stu2 ;
```

```
struct     结构体名
{ 类型标识符    成员名;
  类型标识符    成员名;
        ……
}变量名表列;
```

定义结构体类型

定义结构体变量

注 只有在定义了结构体变量后系统才为其分配内存。

❖结构体成员名与程序中变量名可相同，两者不代表同一个对象。

```
struct  student
  {   int  num;
      char name[20];
      float score;
  }stu;
int  num;
```

stu. num

num

结构体变量的引用

★引用规则

- ❖ 结构体变量不能整体引用,只能引用变量成员

结构体变量名.成员名

成员(分量)运算符
优先级: 1
结合性:从左向右

```
struct   student
{ int num;
   char  name[20];
   char sex;
   int age;
   float score;
   char addr[30];
}stu1,stu2;
```

stu1.num=10;

stu1.age++;

stu1.score=85.5;
stu1.score+=stu2.score;

```
main()
{
  struct   student
  { int No;
     float score;
  } stu1,stu2;
   scanf("%d,%f",&stu1);   (×)
   scanf("%d,%f",&stu1.No, &stu1.score); ( √ )
   printf("%d,%f",stu1);      (×)
   printf("%d,%f" , stu1.No, stu1.score); ( √ )
   stu2=stu1; ( √ )
}
```

结构体类型变量成员的引用方法

结构体类型变量是通过成员运算符“.”逐一访问其成员，一般形式：

结构体类型变量名.成员名

说明：

1、“.”是成员运算符，它是双目运算符，前面的运算对象必须是结构体类型的变量名（或结构体类型数组元素），后面的运算对象是该结构体类型的成员名。

2、成员运算符“.”的优先级最高，与圆括号运算符（）、下标运算符[]的优先级相同，结合性是从左至右。

3、C语言规定不能对一个结构体类型变量整体地进行输入和输出操作，只能对结构体类型变量的成员进行输入和输出操作。

4、结构体类型变量的成员可以像普通变量一样进行赋值和运算。

5、相同类型的结构体类型变量之间可以互相赋值。

6、两个结构体类型变量不能用关系运算符“==”或“！=”进行比较操作。

结构体类型变量成员地址的引用方法

结构体类型变量成员地址的引用方法：

&结构体类型变量名.成员名　　/*成员是变量形式的变量地址*/

结构体类型变量名.成员名　　/*成员是数组形式的数组首地址*/

&结构体类型变量名.成员数组[下标]　/*成员是数组，其数组元素的地址*/

需要注意的是，存放结构体类型变量成员地址的指针变量的数据类型必须与该成员的数据类型相同。

结构体类型变量地址的引用方法

结构体类型变量地址的引用方法：

&结构体类型变量名

需要注意的是，存放结构体类型变量地址的指针变量必须是相同类型的结构体类型指针变量。

第三节 结构体类型数组

结构体类型数组的定义和初始化：

定义结构体类型数组的三种方法：

1、先定义结构体类型，再定义结构体类型数组并初始化。

2、定义结构体类型的同时定义结构体类型数组并初始化。

3、定义无名称的结构体类型的同时定义结构体类型数组并初始化。

结构体类型数组定义的一般格式：

```
struct  结构体类型名
{
    成员列表;
};
```

结构体类型数组的定义和初始化

struct 结构体类型名 结构体类型数组名[数组长度];

与普通数组一样，结构体类型数组也可以在定义时进行初始化，其初始化的格式：

结构体类型数组名[n]={{初值表1},{初值表2},...,{初值表n}};

定义结构体类型数组并初始化的例子。

方法一：

```
struct std_info          /*先定义结构体类型*/
{
    int no;
    char name[10];
    char sex;
};
…
struct std_info s[3]={{100,"ZhangYi",'f'},{101,"WangHong",'m'},{102,"LiSan",'f'}};
/*后定义一个结构体类型数组 s，共有三个元素，即 s[0]～s[2]，每个数组元素都是
结构体类型，即每个数组元素都包括结构体类型 std_info 的三个成员*/
```

方法二：

```
struct std_info          /*定义结构体类型的同时定义结构体类型数组 s 并赋初值*/
{
    int no;
    char name[10];
    char sex;
}s[3]={{100,"ZhangYi",'f'},{101,"WangHong",'m'},{102,"LiSan",'f'}};
```

方法三：

```
struct                   /*定义无名称的结构体类型的同时定义结构体类型数组 s 并赋初值*/
{
    int no;
    char name[10];
    char sex;
}s[3]={{100,"ZhangYi",'f'},{101,"WangHong",'m'},{102,"LiSan",'f'}};
```

结构体类型数组元素成员的引用方法

结构体类型数组元素成员的引用方法：

结构体类型数组名[下标].成员名

程序代码如下：

```
#include <stdio.h>
struct s
{
    int num;
    char color;
    char type;
} car[ ]={ {101,'G','c'},
           {210,'Y','m'},
           {105,'R','l'},
           {222,'B','s'},
           {308,'P','b'}
         };                    /*定义结构体类型数组 car，同时进行初始化*/
int main(void)
{
    int i;
    printf("number color type\n");
    for(i=0;i<5;i++)
        printf("%-9d%-6c%c\n",car[i].num,car[i].color,car[i].type);  /*依次输出结构体类型数组的每
                                                                      个元素 car[i]的各个成员值*/
    return 0;
}
```

程序运行结果如下：

```
number color type
101      G    c
201      Y    m
105      R    l
222      B    s
308      P    b
```

结构体类型数组元素成员地址的引用方法

结构体类型数组元素成员地址的引用方法：

&结构体类型数组名[下标].成员名

存放结构体类型数组元素成员地址的指针变量的数据类型必须与结构体类型中该成员的数据类型相同。

结构体类型数组元素成员地址的引用例子

```
#include <stdio.h>
#define NAMESIZE   20
struct std_info
{
    char name[NAMESIZE];
    int age;
    char sex;
}s[10];                          /*定义结构体类型数组 s*/
int main(void)
{
    int i;
    printf("Please input student data\n");
    i=0;
    while(i<4)                   /*从键盘输入四个学生的信息分别存入结构体类型数组 s*/
    {
        printf("name:   age:   sex: ");
        scanf("%s",s[i].name);
        scanf("%d",&s[i].age);
        getchar( );
        scanf("%c",&s[i].sex);
        i=i+1;
    }
    return 0;
}
```

程序运行结果如下：

```
Please input student data
name:   age:   sex: David 23 M↵
name:   age:   sex: Elliott 22 M↵
name:   age:   sex: Evan 23 M↵
name:   age:   sex: Dennis 21 F↵
```

第四节 结构体类型指针

指向结构体类型变量的指针变量：

定义指向结构体类型变量的指针变量和定义结构体类型变量的方法基本相同，区别是在结构体类型指针变量名的前面加一个“*”。

一般地，如果指针变量pointer已指向结构体类型变量var，则以下三种形式等价：

方式一：var.成员

方式二：pointer->成员

方式三：(*pointer).成员 /*“*pointer”外面的一对圆括号不能省略*/

❖使用结构体指针变量引用成员形式

例　int　n;
　　int　*p=&n;
　　*p=10; ⇔ n=10

struct　student　stu1;
struct　student　*p=&stu1;
stu1.num=101; ⇔ (*p).num=101

以下三种形式等价：

结构体变量名.成员名　　stu.num =101;

(*结构体指针名).成员名　(*p).num=101

结构体指针名->成员名　　p->num =101

( )不能少！

指向运算符
优先级: 1
结合方向：从左向右

指向结构体类型变量的指针变量

需要注意以下几点：

1、方式一中，成员运算符“.”左侧的运算对象只能是结构体类型变量。

2、方式二中的“->”称为指向成员运算符或简称为指向运算符。该运算符是双目运算符，其左侧的运算对象必须是结构体类型变量（或数组元素）的地址，也可以是已指向结构体类型变量（或数组元素）的指针变量，否则会出错；右侧的运算对象必须是该结构体类型的成员名。指向运算符的优先级和“()”“[]”“.”是同级的，结合性是从左至右。

3、方式三中的“*指针变量”代表了它所指向的结构体类型变量，利用成员运算符“.”来引用，等价于“结构体类型变量.成员名”。“*指针变量”必须用圆括号括起来，因为运算符“*”的优先级低于运算符“.”，若没有圆括号则优先处理运算符“.”，将出现“指针变量.成员名”会造成语法错误。

指针变量指向结构体类型数组元素

如果一个结构体类型数组元素的地址已赋予相同结构体类型的指针变量（即指针变量指向结构体类型数组元素），可以使用以下两种方式引用该元素的成员，其作用完全相同。

方式一：(*指针变量).成员名

方式二：指针变量->成员名

这里的指针变量必须已经指向某一结构体类型数组元素。

指针变量指向结构体类型数组

当一个结构体类型数组的首地址已经赋予相同结构体类型的指针变量（即该指针变量已经指向结构体类型数组），可以使用以下两种方式引用下标为i的数组元素的成员，其作用完全相同。

- 方式一：(*(指针变量+i)).成员名
- 方式二：(指针变量+i)->成员名

这里的指针变量必须已经指向某一结构体类型数组，则上述两种引用方式均等价于“数组名[i].成员名”。

指针变量指向结构体类型数组

说明：

1、如果指针变量p已经指向一个结构体类型数组，则p+1指向结构体类型数组的下一个元素，而不是当前元素的下一个成员。

2、如果指针变量p已经指向一个结构体类型变量（或结构体类型数组），则不能再使之指向结构体类型变量（或结构体类型数组元素）的某一成员。

利用全局变量传递结构体类型数据

利用全局变量传递结构体类型数据时，只要在程序的开头定义结构体类型及其变量或数组，则可以在其后面定义的函数间传递结构体类型数据。

利用返回值传递结构体类型数据

利用返回值传递结构体类型数据时，只能返回一个结构体类型数据，定义时函数的类型必须是已定义的结构体类型，利用语句return(表达式);返回的表达式值也必须是同一结构体类型的数据。调用函数后的返回值必须用同一结构体类型变量或数组元素接收。

利用形参和实参结合传递结构体类型数据

1、地址传递方式

地址传递方式中，形参可以是结构体类型数组或指针变量，对应的实参必须是同一结构体类型的变量地址、数组名、数组元素地址或已赋值的结构体类型的指针变量。

2、值传递方式

值传递方式中，形参为结构体类型变量，对应的实参必须是同一结构体类型的变量或数组元素。

利用形参和实参结合传递结构体类型数据

1、地址传递方式

地址传递方式中，形参可以是结构体类型数组或指针变量，对应的实参必须是同一结构体类型的变量地址、数组名、数组元素地址或已赋值的结构体类型的指针变量。

2、值传递方式

值传递方式中，形参为结构体类型变量，对应的实参必须是同一结构体类型的变量或数组元素。

课堂练习

2、设struct{int a;char c;}m,*p=&m;，用指针引用成员a的形式是(*p).a或_____。

课堂练习

2、设struct{int a;char c;}m,*p=&m;，用指针引用成员a的形式是(*p).a或_____。

答案：p->a

解析：一般地，如果指针变量pointer已指向结构体类型变量var，则三种形式等价：var.成员、pointer->成员、(*pointer).成员，即用指针引用成员a的形式是(*p).a也可以为p->a。

课堂练习

3、设

```
struct student
{ char name[10];
int sex;
int num;
}s,*p=&s;
```

以下对结构型变量s中成员num的非法引用是（　　）

A:p->num　　B:(*p).num

C:s.num　　D:student.num

课堂练习

3、设

```
struct student
{ char name[10];
int sex;
int num;
}s,*p=&s;
```

以下对结构型变量s中成员num的非法引用是（　　）

A:p->num　　B:(*p).num

C:s.num　　D:student.num

答案：D

解析：指向结构体类型变量的指针变量的三种等价形式：结构体类型变量.成员、指针变量->成员、（*指针变量）.成员。student.num中的student不是结构体类型变量。

第六节 自定义类型

typedef定义类型别名的一般形式：

【格式】typedef 原类型符 新类型符;

【参数】“原类型符”可以是基本类型符，也可以是用户自定义的无名结构体类型。“新类型符”是用户自定义的一个标识符，通常采用大写字母。

【功能】将“原类型符”定义为用户自定义的“新类型符”，以后可以使用“新类型符”定义相应数据类型的变量、数组、指针变量、结构体类型和函数。

【说明】

1、typedef只能定义类型名，不能定义变量名，即定义类型名的别名，并不能创造一个新的类型。

2、typedef与宏定义#define具有相似之处，但二者是不同的。前者是在编译时完成的；后者是在编译预处理时完成的，且只能进行简单的字符串替换。

基本数据类型符的自定义

【格式】typedef 基本数据类型符 用户类型符;

【功能】将“基本数据类型符”定义为“用户类型符”。

例如，为实型float定义一个别名REAL的过程如下：

1、按照定义实型变量的方法，写出定义体，即float f;

2、将变量名换成别名，即float REAL;

3、在定义体最前面加上typedef，即typedef float REAL;

4、已定义完新类型名，可用此新类型名定义变量，即REAL x,y;

数组类型的自定义

【格式】typedef 基本数据类型符 用户类型符[数组长度];

【功能】以后可以使用“用户类型符”定义“基本数据类型符”的数组，其长度为定义时确定的数组长度。

例如，typedef int ARRAY[10];/*定义ARRAY为整型、长度为10的数组的用户类型符*/

ARRAY a,b,c;/*等价于int a[10],b[10],c[10];*/

ARRAY是长度为10的整型数组，可以用ARRAY定义多个相同类型和长度的数组。

结构体类型的自定义

【格式】typedef struct

{

数据类型符 成员名1;

数据类型符 成员名2;

…

数据类型符 成员名n;

}用户类型符;

【功能】以后可以使用“用户类型符”定义含有n个成员的结构体类型变量、数组和指针变量等。

结构体类型的自定义

例如，为结构体类型date定义一个别名DATE。

struct date

{ int year，month，day; };

1、按照定义结构体类型变量的方法，写出定义体，即struct date{…}d;

2、将变量名换成别名，即struct date{…}DATE;

3、在定义体最前面加上typedef，即typedef struct date{…}DATE;

4、已定义完新类型名，可用此新类型名定义变量，即DATE s1,s2;

指针型的自定义

【格式】typedef 基本数据类型符 *用户类型符;

【功能】以后可以使用“用户类型符”定义“基本数据类型符”类型的指针变量或数组等。

自定义指针类型的例子：

程序段代码如下：

```
typedef int *P_I;          /*定义 P_I 为整型指针的用户类型符*/
typedef char *P_C[5];      /*定义 P_C 为长度是 5 的字符型指针数组的用户类型符*/
int main(void)
{
    P_I   p1,p2;           /*等价于 int *p1,*p2;*/
    P_C   p3,p4;           /*等价于 char *p3[5],*p4[5];*/
    …
}
```

指针型的自定义

【格式】typedef 基本数据类型符 *用户类型符;

【功能】以后可以使用“用户类型符”定义“基本数据类型符”类型的指针变量或数组等。

自定义指针类型的例子：

程序段代码如下：

```
typedef int *P_I;          /*定义 P_I 为整型指针的用户类型符*/
typedef char *P_C[5];      /*定义 P_C 为长度是 5 的字符型指针数组的用户类型符*/
int main(void)
{
    P_I   p1,p2;           /*等价于 int *p1,*p2;*/
    P_C   p3,p4;           /*等价于 char *p3[5],*p4[5];*/
    …
}
```

课堂练习

4、设typedef float REAL；则REAL是（　　）

A. 变量名

B. 常量名

C. 函数名

D. 类型名

课堂练习

4、设typedef float REAL；则REAL是（　　）

A. 变量名

B. 常量名

C. 函数名

D. 类型名

答案：D

解析：自定义类型语句：typedef 基本数据类型符　用户类型符。因此typedef float REAL；中的REAL是类型名。

课堂练习

5、设typedef int ARRAY[10]；，与ARRAY a，b；等价的是（　　）

A. int a，b；

B. int a[10]，b[10]；

C. int a[10]，b；

D. int a，b[10]；

课堂练习

5、设typedef int ARRAY[10]；，与ARRAY a，b；等价的是（　　）

A. int a，b；

B. int a[10]，b[10]；

C. int a[10]，b；

D. int a，b[10]；

答案：B

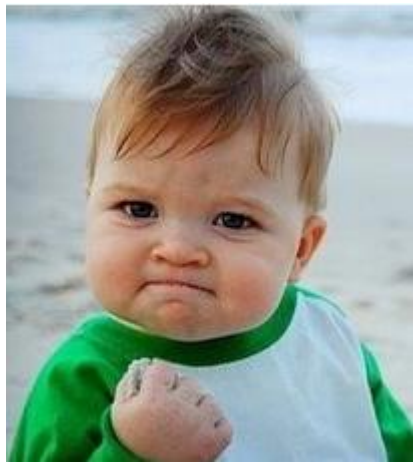
解析：typedef int ARRAY[10]；定义ARRAY为整型、长度为10的数组的用户类型符，ARRAY a，b；等价于int a[10]，b[10]；。

附录A ASCII码表

L \ H	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

其中，0～31 为控制字符。

32～127 为可打印字符。



祝大家顺利通过考试！