第8章 结构体与共用体

《C语言程序设计新编教程》

定义一个整形变量a，赋值5给a，然后使用a：a=a-1。如何编写该代码?

整形变量int数据类型（数据结构），是系统自带的，那么想自己构造一个数据是否可以?

函数可以自定义，数据结构是否可以?

能力要求

CAPACITY

- 掌握结构体的类型和变量的定义
- 掌握结构体的变量、数组指针的应用
- 熟悉共用体类型的定义
- 掌握共用体类型的使用方式

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

课程导入

❖**数据**的基本类型：整、实、字符。

❖**数组**是构造类型：每个元素为同一类型

❖有些问题仅用基本类型和数组来描述，无法反映其内在联系，如学生情况：

num	name	sex	age	score	addr
int	char	char	int	int	char
17001	Zhan xi	m	21	92. 5	Guang Dong
17001	Wang li	f	24	87.6	Bei jing

上述数据互相独立又相互关联，如：均与学号和姓名关联。需要将其组合成一个有机的整体，C语言可以将由**不同类型数据**组成的这种**数据结构**组织成一个组合项，称为**结构体**（structure）。

在C语言中，结构体(struct)指的是一种数据结构，是C语言中聚合数据类型(aggregate data type)的一类。结构体可以被声明为变量、指针或数组等，用以实现较复杂的数据结构。结构体同时也是一些元素的集合，这些元素称为结构体的成员(member)，且这些成员可以为不同的类型，成员一般用名字访问。

格式：

```
struct 结构体名
    {
      类型名 成员名1;
      类型名 成员名2;
      …
     };
```

格式：struct 结构体名

{

类型名 成员名1;

类型名 成员名2;

…

};

struct是关键字，不能省略

；不能省略

功能：定义一个结构体类型。

说明：

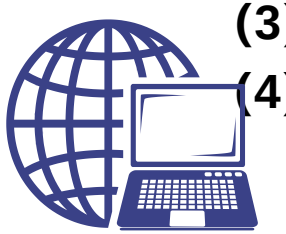
（1）struct是关键字，结构体名是用户自定义的标识符，其命名规则与变量相同。

（2）花括号“{}”中是组成该结构体类型的数据项，或称为结构体类型中的成员。每个类型名后面可以定义多个不同类型的成员。

（3）结构体成员的数据类型可以是简单类型、数组、指针或已定义过的结构体类型等。

（4）结构体类型的定义部分一般放在函数外，整个定义以分号结束。

注意：这只是声明一种数据类型并没有定义变量。



例如：在描述学生信息时，通常需要了解他们的学号、姓名、性别、成绩、家庭住址等信息。比如一个学生的情况可以如表8-1所示。

这样，就可以定义一个名叫studinf的结构体类型来描述一个学生的情况。格式如下：

```
struct studinf
{
    int num;
    char name[20];
    char sex;
    float score;
    char addr[30];
 };
```

结构体类型定义仅描述结构体的组成,不分配内存空间

以上定义了一个结构体类型studinf，我们就可以象使用int定义变量一样，使用这个名叫studinf的“结构体类型”来定义变量了。

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

方法一：先定义结构体类型，再定义结构体变量

声明结构体类型时不分配存储单元，使用该类型定义变量时才分配存储单元。定义变量方法3种：

★ 先声明结构体类型，再定义结构体变量

例如，前面已经定义了studinf结构体类型，语句：

struct studinf student1, student2;

定义student1和student2为struct studinf类型变量，即它们是具有struct studinf类型的结构，如表8-2所示。

stu1					stu2				
num	name	sex	KC	score	num	name	sex	KC	score
20172101	Li xiaoming	M	Java	87	20172102	Zhangli	F	PHP	92

方法二：定义了一个结构体类型，同时定义结构体变量

格式：

```
struct 结构体名
{
    成员表;
  }变量名表;
```

例如:

```
struct student
{
  int num;
  char name[10];
  char sex;
  char KC[20];
  int score[2];
 }s1,s2;
```

上面定义了两个结构体变量 s1，s2，它们是已定义的 student 结构体类型，系统为 每个结构体变量分配存储单元。使用 student 结构体类型定义结构体变量时，要在前面加上 struct 关键字。

方法三：直接定义结构体类型变量

如果直接定义了 stu1 和 stu2 两个结构体变量，结构体类型的名字可以缺省。在内存中，stu1 占连续的一片存储单元。

格式：struct

{

成员表;

}变量名表;

注意：

这种情况下定义的结构体变量没有结构体类型名，所以该类型的结构体变量只能使用一次（该方法又称为无名定义结构体类型）。

例如：

```
struct
{
  int num;
  char name[10];
  char sex;
  char KC[20];
  int score[2];
}stu1,stu2;
```

❖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

内容导航

CONTENTS

结构体变量的赋值

方法1：结构变量的赋值就是给各成员赋值，可用输入语句或赋值语句来完成。格式如下：

```
struct 结构体名
{
  成员表;
}变量名={数据项表};
```

```
struct student
{
int id;
char name[32];
int age;
char sex;
float score;
}stu1={1, “Zhang3”,21, ‘F’,99};
```

结构体变量的赋值

或：

struct 结构体名 变量名={数据项表};

```
struct student
{
int id;
char name[32];
int age;
char sex;
float score;
}stu1;
struct student stu1={1, “Zhang3”,21,
‘F’,99};
```

例如：给结构体变量赋值并输出其值。

程序代码：

```
#include <stdio.h>
main()
{
struct stu  /*定义结构体类型stu */
    {
      int num;
      char *name;
      char sex;
      float score;
    } boy1,boy2;
  boy1.num=102;
  boy1.name="Zhang ping";
  printf("input sex and score\n");
  scanf("%c %f",&boy1.sex,&boy1.score);  /*通过输入语句实现结构体变量赋值*/
  boy2=boy1;
  printf("Number=%d\nName=%s\n",boy2.num,boy2.name);
  printf("Sex=%c\nScore=%f\n",boy2.sex,boy2.score);
}
```

例如：给结构体变量赋值并输出其值。

程序代码：

```
#include <stdio.h>
main()
{
struct stu  /*定义结构体类型stu */
    {
      int num;
      char *name;
      char sex;
      float score;
    } boy1,boy2;
  boy1.num=102;
  boy1.name="Zhang ping";
  printf("input sex and score\n");
  scanf("%c %f",&boy1.sex,&boy1.score);  /*通过输入语句实现结构体变量赋值*/
  boy2=boy1;
  printf("Number=%d\nName=%s\n",boy2.num,boy2.name);
  printf("Sex=%c\nScore=%f\n",boy2.sex,boy2.score);
}
```

```
input sex and score
m
80
Number=102
Name=Zhang ping
Sex=m
Score=80.000000
```

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

结构体变量引用

对结构体变量引用，结构体变量不能整体引用,只能引用变量成员，其形式如下：

结构体变量名.成员名

其中：“.”是结构体的成员运算符，它在所有运算符中优先级最高，而且是左结合，因此，上述引用结构体成员的写法，在程序中被作为一个整体看待。

这样，结构体变量birthday中的三个成员可分别表示为：

birthday.month

birthday.day

birthday.year

如果成员本身又属于一个结构体类型，则要用若干个成员运算符，一级一级地找到最低一级（基层）的成员。只能对最低一级的成员进行赋值、存取或运算。

结构体变量的引用规则

在定义了结构体变量以后，就可以使用这个变量进行操作。但应遵守以下规则：

（1）只能对结构体变量中的各成员分别进行输入和输出：

```
printf("%d，%s，%c，%d，%s", student1.num，student1.name，student1.sex, student1.age, student1.addr);
```

（2）结构体变量中的每个成员都可以像普通变量一样进行各种运算，例如：student1.num表示student1变量中的num成员，即学号项，可以用下面的语句对该变量成员赋值：

```
student1.num=2008001;
student1.age=20;
```

结构体变量的引用规则

(3) 可以引用成员的地址，也可以引用结构体变量的地址。如：

scanf("%d", &student1.num); (输入student1.num的值，引用成员的地址)

printf("%d", &student1); (输出student1的首地址，引用结构体变量的地址)

结构体变量的地址主要用作函数参数，这样做比直接传递结构体变量可以提高程序运行效率。

(4) 同类型的结构体变量可以整体赋值。

例如，对于前面定义的结构体变量student1和student2，可以有：

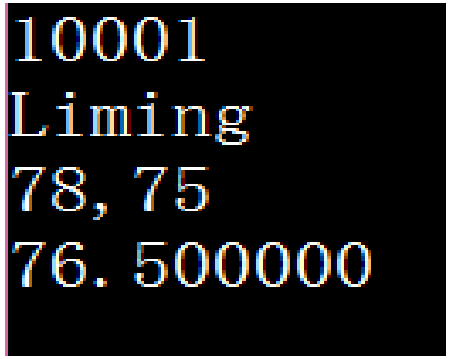
```
student2=student1;
```

其作用是将结构体变量student1的各成员值在student2中复制一份。

【例 8.3】结构体成员的使用

程序代码：

```
#include <stdio.h>
#include <string.h>
struct student
{
int num;
char name[10];
int score[2];
float aver;
};
void main()
{
struct student stu;
stu.num=10001;
strcpy(stu.name,"Liming");
stu.score[0]=78;
stu.score[1]=75;
stu.aver=(stu.score[0]+ stu.score[1])/2.0;
printf("%d\n",stu.num);
printf("%s\n",stu.name);
printf("%d,%d \n", stu.score[0], stu.score[1]);
printf("%f\n", stu.aver);
}
```



内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

结构体数组的定义

一个结构体变量只能存放一个对象的一组相关信息，结构体数组可以存放多个同类型对象的信息。

方法一：直接定义法，其一般格式如下：

```
struct
{
   成员列表;
}数组名[元素个数];
```

例如：

```
struct
{
  int num;
  char name[10];
  int score[2];
  float aver;
}stu[30];
```

结构体数组的定义

一个结构体变量只能存放一个对象的一组相关信息，结构体数组可以存放多个同类型对象的信息。

方法二：先定义结构体类型再定义结构体变量法，其一般格式如下：

```
struct 结构体名
{
    成员列表;
};
struct 结构体名 数组名[元素个数];
```

例如：

```
struct student
 {
   int num;
   char name[10];
   int score[2];
   float aver;
};
struct student stu[30]; //定义了一个数组stu，其元素为struct student类型，数组有30个元素。
```

结构体数组的定义

一个结构体变量只能存放一个对象的一组相关信息，结构体数组可以存放多个同类型对象的信息。

方法三：同时定义结构体类型和结构体数组，其一般格式如下：

```
struct 结构体名
{
   成员列表;
}数组名[元素个数];
```

例如：

```
struct student
{
int num;
char name[10];
int score[2];
float aver;
}stu[30];
```

结构体数组的初始化

结构体数组初始化的一般格式为：

```
struct 结构体名
{
  成员列表;
}数组名[元素个数]={{数据项表1},{数据项表2},…};
```

或者：

```
Struct 结构体名 数组名[元素个数]= {{数据项表1},{数据项表2},…};
```

结构体数组的初始化

例如：

```
struct student /*定义结构体类型*/
{
  int num;
  char name[10];
  int score[2];
  float aver;
}stu[2] ={{101,"Liming",{75,87},0},{102,"Wangli",{70,80},0}};//结构体数组初始化
```

或者：

```
struct student stu[2]={{101,"Liming",{75,87},0},{102,"Wangli",{70,80},0}};
```

注意：定义了结构体数组以后，要通过结构体数组元素访问其成员。例如，结构体数组 stu中 第二名学生的平均成绩为 stu[1].aver。

结构体数组的初始化

【例 8.4】计算全班每个学生两门课的平均考试成绩，并在屏幕上显示学生学号、姓名 及其平均成绩。假设全班共有 5 名学生。

程序代码：

```
#include <stdio.h>
#define N 5
void main()
{
struct student
{
    int num;
    char name[10];
int score[2];
float aver;
}stu[N];
int i;
printf("输入%d名学生姓名及2门考试成绩。\n",N);
for(i=0;i<N;i++)
{
printf("学号： ");
scanf("%d",&stu[i].num);
printf("姓名： ");
scanf("%s",stu[i].name);
printf("成绩1，成绩2: ");
scanf("%d,%d",&stu[i].score[0], &stu[i].score[1]);
stu[i].aver=(stu[i].score[0]+ stu[i].score[1])/2.0;
}
```

结构体数组的初始化

```
    for(i=0;i<N;i++)
    printf("%d,%s,%f\n",
stu[i].num,stu[i].name,stu[i].aver);
    }
```

```
输入5名学生姓名及2门考试成绩。
学号： 20171227
姓名： 李明生
成绩1， 成绩2： 87,93
学号： 20171228
姓名： 林雨明
成绩1， 成绩2： 76,84
学号： 20171229
姓名： 张捐
成绩1， 成绩2： 84,94
学号： 20171230
姓名： 莫哲明
成绩1， 成绩2： 93,87
学号： 20171231
姓名： 陈晓
成绩1， 成绩2： 93,78
20171227,李明生,90.000000
20171228,林雨明,80.000000
20171229,张捐,89.000000
20171230,莫哲明,90.000000
20171231,陈晓,85.500000
```

结构体数组的初始化

【例 8.5】使用C编写程序，实现对候选人选票的统计。假设有三个候选人，每次输入一个候选人的名字，要求最后统计输出每人得票结果。

程序代码：

```
#include <stdio.h>
#include <string.h>
struct person
{
char name[20];
int count;
}leader[3]={"li", 0, "han", 0, "ma", 0};  /*初始化*/
```

```
main()
{
 int i, j;
char name[20];
printf("请输入候选人名单（ma；han；li）：\n");
for(i=1;i<=10;i++)                /*假设共10张票*/
{
printf("%2d:",i);
scanf("%s", name);
for(j=0;j<3;j++)                  /*计票*/
if(!strcmp(strlwr(name), leader[j].name))  /*输入名与候选人名比较*/
leader[j].count++;          /*票数累加*/
}
printf("\n");
for(i=0;i<3;i++)
printf("%5s: %d\t", leader[i].name, leader[i].count);
}
```

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

结构体的合法操作只有几种：作为一个整体复制和赋值，通过&运算符取地址，访问其成员。其中，复制和赋值包括向函数传递参数以及从函数返回值。结构体之间不可以进行比较，但可以用一个常量成员值列表初始化结构体，结构体也可以通过赋值进行初始化。首先来看一下函数：

程序代码：

```
#include <stdio.h>
struct tree
{
        int x;
        int y;
} t;
void func(struct tree t)
{
        t.x = 10;
        t.y = 20;
}
main()
{
        t.x = 1;
        t.y = 2;
        func(t);
        printf("%d,%d\n", t.x, t.y);
}
```

运行结果：1,2

为了更进一步理解结构体与函数，我们编写一个对点和矩形进行操作的函数。至少可以通过3种可能的方法传递结构体：一是分别传递种个结构体成员，二是传递整个结构体，三是传递指向结构体的指针。

以下函数makepoint，它带有两个整型参数，并返回一个point类型的结构：

```
/* makepoint: 通过x、y值确定一个点*/
struct point makepoint(int x, int y)//makepoint()是结构体point的变量，此变量为函数。
{
struct point temp;//定义一个结构体point 变量temp
temp.x = x;
temp.y = y;
return temp;
}
```

注意，参数名和结构成员同名不会引起冲突。事实上，使用重名强调了两者之间的关系。

现在可以用makepoint动态初始化任意结构，也可以向函数提供结构类型的参数。例如：

```
struct rect screen;
struct point middle;
struct point makepoint(int, int);
screen.pt1 = makepoint(0, 0);
screen.pt2 = makepoint(XMAX, YMAX);
middle = makepoint((screen.pt1.x + screen.pt2.x) / 2,
(screen.pt1.y + screen.pt2.y)/2);
```

下面通过一系列的函数对点进行算术运算。例如：

```
/* addpoint: 将两个点相加*/
struct point addpoint(struct point p1, struct point p2)
{
p1.x += p2.x;
p1.y += p2.y;
return p1;    }
```

其中，函数的参数和返回值都是结构体类型。之所以直接将相加所得的结果赋值给p1，而没有使用显式的临时变量存储，是为了强调结构类型的参数和其他类型的参数一样，都是通过值传递的。

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

1．指向结构变量的指针

结构指针变量说明的一般形式为：

struct 结构名 *结构指针变量名

例如，在前面的例题中定义了student这个结构，如要说明一个指向 student的指针变量 pstu，可写为：

struct student *pstu;

结构名和结构变量是两个不同的概念，不能混淆。结构名只能表示一个结构形式，编译 系统并不对它分配内存空间。只有当某变量被说明为这种类型的结构时，才对该变量分配存 储空间。

1．指向结构变量的指针

有了结构指针变量，就能更方便地访问结构变量的各个成员。其访问的一般形式为：

(*结构指针变量).成员名

或为：

结构指针变量->成员名

例如：

(*pstu).num

或者：

pstu->num

应该注意(*pstu)两侧的括号不可少，因为成员符“.”的优先级高于“*”。如去掉括号写 作*pstu.num则等效于*(pstu.num)，这样，意义就完全不对了。下面通过例子来说明结构指针变量的具体说明和使用方法。

1. 指向结构变量的指针

【例8.6】应用结构体指针处理学生的基本信息

程序代码：

```
#include<stdio.h>
struct stu
 {
   int num;
   char *name;
   char sex;
   float score;
}boy1={102,"Zhang ping",'M',78.5},*pstu;
```

1. 指向结构变量的指针

```
main()
{
 pstu=&boy1;
 /*下面分别使用三种结构体成员运算符输出数据*/
 printf("Number=%d\nName=%s\n",boy1.num,boy1.name);
 printf("Sex=%c\nScore=%f\n\n",boy1.sex,boy1.score);
 printf("Number=%d\nName=%s\n",(*pstu).num,(*pstu).name);
 printf("Sex=%c\nScore=%f\n\n",(*pstu).sex,(*pstu).score);
 printf("Number=%d\nName=%s\n",pstu->num,pstu->name);
 printf("Sex=%c\nScore=%f\n\n",pstu->sex,pstu->score);
}
```

```
Number=102
Name=Zhang ping
Sex=M
Score=78.500000

Number=102
Name=Zhang ping
Sex=M
Score=78.500000

Number=102
Name=Zhang ping
Sex=M
Score=78.500000
```

2．指向结构数组的指针

指针变量可以指向一个结构数组，这时结构指针变量的值是整个结构数组的首地址。结 构指针变量也可指向结构数组的一个元素，这时结构指针变量的值是该结构数组元素的首地 址。

【例 8.7】用指针变量输出结构数组。

程序代码：

```
#include<stdio.h>
struct stu
{
int num;
char *name;
char sex;
float score;
}boy[5]={
        {101,"Zhou ping",'M',45},
        {102,"Zhang ping",'M',62.5},
        {103,"Liou fang",'F',92.5},
        {104,"Cheng ling",'F',87},
        {105,"Wang ming",'M',58},
      };
```

2．指向结构数组的指针

```
main()
{
  struct stu *ps;
  printf("No\tName\t\t\tSex\tScore\t\n");
  for(ps=boy;ps<boy+5;ps++)
  printf("%d\t%s\t\t%c\t%f\t\n",ps->num,ps->name,ps->sex,ps->score);
}
```

```
No      Name                    Sex     Score
101     Zhou ping               M       45.000000
102     Zhang ping              M       62.500000
103     Liou fang               F       92.500000
104     Cheng ling              F       87.000000
105     Wang ming               M       58.000000
```

2．指向结构数组的指针

注意：一个结构指针变量虽然可以用来访问结构变量或结构数组元素的成员，但是，不能使它指向一个成员。也就是说不允许取一个成员的地址来赋予它。因此，下面的赋值是错误的。

ps=&boy[1].sex;

而只能是：

ps=boy;(赋予数组首地址)或者是

ps=&boy[0];(赋予0号元素首地址)

3.用指向结构体的指针作函数参数

结构体类型的数据也可以作为实参传递到另一个函数中。结构体类型的数据作实参，通常有以下几种形式：

（1）结构体变量的成员作实参

结构体变量的成员作实参如同普通变量作实参的情况一样，属于“值传递”。下面的函数调用语句，显示了结构体变量的成员作实参的使用方法：

```
add(student[i].score[0], student[i].score[1], student[i].score[2]);
```

3.用指向结构体的指针作函数参数

（2）结构体变量作实参

用结构体变量作实参，形参应与实参类型相同，参数传递时，按顺序把实参的各个成员依次传递给形参对应的成员，属于值传递。在函数调用期间，形参也要占用内存单元。这种传递方式在空间和时间上开销较大，如果结构体的规模很大时，开销是很可观的。一般较少用这种方法。

（3）指向结构体变量的指针（或数组名）作实参

该形式属于地址传递。函数被调用时，不仅可以访问实参的结构体变量各个成员的数据，而且被调用函数还可以修改实参的数据。这种参数传递方式效率较高，比较常用。

3.用指向结构体的指针作函数参数

【例 8.8】计算一组学生的平均成绩和不及格人数。

【解题思路】用结构指针变量作函数参数编程。

程序代码：

```
struct stu
{
 int num;
 char *name;
 char sex;
 float score;
 }boy[5]={
 {101,"Li ping",'M',45},
 {102,"Zhang ping",'M',62.5},
 {103,"He fang",'F',92.5},
 {104,"Cheng ling",'F',87},
 {105,"Wang ming",'M',58},
 };
main()
{
 struct stu *ps;
 void ave(struct stu *ps);
 ps=boy;
 ave(ps);
 }
```

3.用指向结构体的指针作函数参数

```
void ave(struct stu *ps)
{
  int c=0,i;
  float ave,s=0;
  for(i=0;i<5;i++,ps++)
    {
      s+=ps->score;
      if(ps->score<60) c+=1;
    }
printf("s=%f\n",s);
ave=s/5;
printf("average=%f\ncount=%d\n",ave,c);
}
```

```
s=345.000000
average=69.000000
count=2
```

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

1.共用体类型的定义和共用体变量的说明

在进行某些C语言程序设计时，可能会需要使几种不同类型的变量存放到同一段内存单元中，使几个变量互相覆盖，也就是使用覆盖技术。比如，可以把int、char、float类型的数据存放在同一个地址开始的内存单元，这种几个不同类型的变量共同占用一段内存的结构，在C语言中，被称作共用体类型结构，简称共用体。

共用体类型的定义形式为：

```
union [共用体名]  /*共用体名可以省略*/
{
成员项表;
};
```

1.共用体类型的定义和共用体变量的说明

在定义共用体的同时，也可以定义共用体变量形式如下：

```
union [共用体名] /*共用体名可以省略*/
{
成员项表;
}共用体变量名表;
或
union 共用体名
{
成员项表;
};
union 共用体名 共用体变量名表;
```

1.共用体类型的定义和共用体变量的说明

例如：

```
union student
{
 int num;
 char name[8];
 char sex;
 int age;
 float score;
}stu1,stu2;
```

或者无名定义：

```
union
{
 int num;
 char name[8];
 char sex;
 int age;
 float score;
} stu1,stu2;
```

1.共用体类型的定义和共用体变量的说明

也可以将定义共用体的类型的同时分开定义相应的变量，如下：

```
union student   /*先定义共用体类型*/
{
 int num;
 char name[8];
 char sex;
 int age;
 float scroe;
};
union student stu1,stu2;  /*后定义共用体类型变量*/
```

1.共用体类型的定义和共用体变量的说明

共用体概念

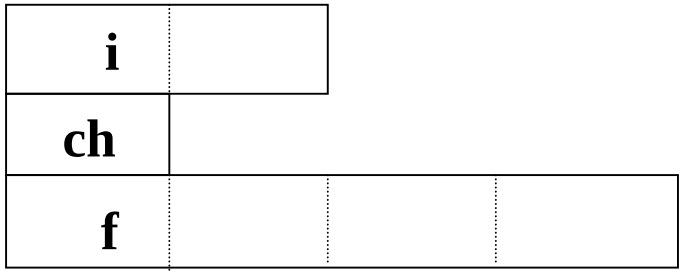
- 构造数据类型,也叫联合体
- 用途：使几个不同类型的变量共占一段内存(相互覆盖)
- 类型定义形式：

```
union  共用体名
{ 类型标识符   成员名;
  类型标识符   成员名;
        ……
};
```

类型定义不分配内存

例

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

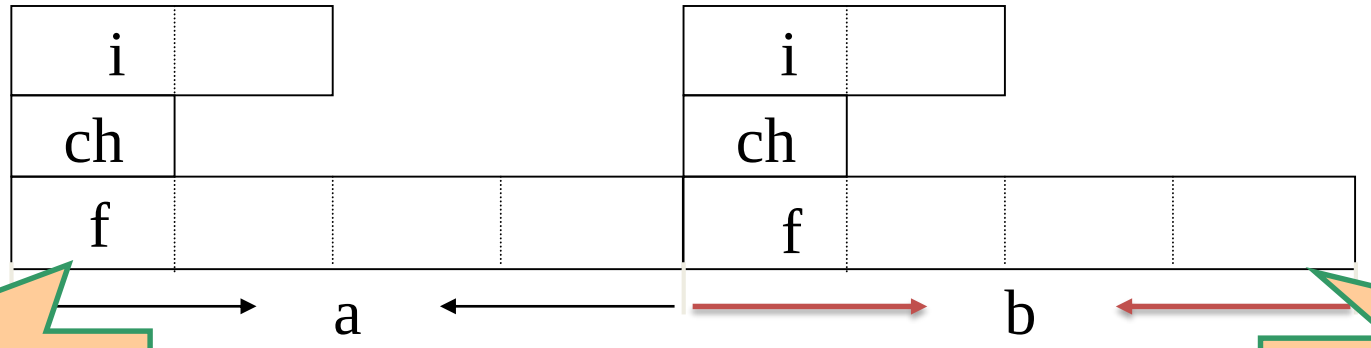


共用体类型的定义

形式一:
```
union data
  {short int i;
    char ch;
    float f;
  } a,b;
```

形式二:
```
union data
  { short int i;
    char ch;
    float f;
   };
   union data a,b,c,*p,d[3];
```

形式三:
```
union
  { short int i;
    char ch;
    float f;
   } a,b,c;
```



共用体变量任何时刻只有一个成员存在

共用体变量定义分配内存,长度=最长成员所占字节数

★共用体变量的引用方式

❖3种方式等价：

- 共用体变量名.成员名
- 共用体指针名->成员名
- (*共用体指针名).成员名

❖引用规则

- 不能引用共用体变量，只能引用其成员

```
printf("%d",a) ;     (×)

printf("%d",a.i);   (√)
```

```
union data
 { int i;
   char ch;
   float f;
 };
 union data a,b,c,*p,d[3];
```

a.i a.ch a.f

p->i p->ch p->f

(*p).i (*p).ch (*p).f

d[0].i d[0].ch d[0].f

★共同体类型数据的特点

(1) 同一个内存段可以用来存放几种不同类型的成员，但在每 一瞬时只能存放其中一种，而不是同时存放几种。

(2) 共用体变量中起作用的成员是最后一次存放的成员

```
例   a.i=1;
     a.ch='a';
     a.f=1.5;
     printf("%d",a.i);        ( × 编译通过，运行结果不对)
     printf("%f",a.f);        (√)
```

(3) 共用体变量和它的各成员的地址都是同一地址。

(4) 不能对共用体变量名赋值，也不能在定义共用体变量时初始化。但可以用一个共用体变量为另一个变量赋值

```
例：union
      { int i;
        char ch;
        float f;
      }a={1,'a',1.5};     (×)
    a=1;                  (×)
    m=a;                  (×)
```

```
例：float  x;
    union
     { int i;   char ch;  float f;
     }a,b;
    a.i=1;   a.ch='a';   a.f=1.5;
    b=a;      (√)
    x=a.f;    (√)
```

内容导航

CONTENTS

结构体

- 结构体的定义
- 结构体变量的定义
- 结构体变量的赋值和初始化
- 结构体变量的引用
- 结构体与数组
- 结构体与函数
- 结构指针变量的说明和使用

共用体

- 共用体类型的定义和变量的说明
- 共用体类型变量的赋值和使用

2.共用体类型变量的赋值和使用

可以引用共用体变量的成员，其用法与结构体完全相同。引用时不能直接引用共用体变量，只能引用变量的成员。若定义共用体类型为：

```
union data  /*共用体*/
{
 int a;
 float b;
 double c;
 char d;
}n,m ;
```

其成员引用为：n.a, m.b, n.c, m.d

注意:不能同时引用四个成员，在某一时刻，只能使用其中之一的成员。

2.共用体类型变量的赋值和使用

例如：

程序代码：

```
main()
{
 union student
 {
  int a;
  float b;
  double c;
  char d;
}stu;
stu.a=6;
printf("stu.a=%d\n",stu.a);
stu.c=67.2;
printf("stu.c=%5.1lf\n",stu.c);
stu.d='W';
stu.b=34.2;
printf("stu.b=%5.1f,stu.d=%c\n",stu.b,stu.d);
}
```

```
stu.a=6
stu.c= 67.2
stu.b= 34.2,stu.d=?
```

2.共用体类型变量的赋值和使用

【例 8.9】结构体和共用体的混合使用

程序代码：

```
#include <stdio.h>
struct ctag
{
  char low;
  char high;
};
union utag
{
  struct ctag bacc;
  short wacc;
}uacc;
int main()
{
  uacc.wacc=(short)0x1234;
  printf("word value is:%04x\n",uacc.wacc);
  printf("high byte is:%02x\n",uacc.bacc.high);
  printf("low byte is:%02x\n",uacc.bacc.low);
  uacc.bacc.high=(char)0xFF;
  printf("word value is : %04x\n",uacc.wacc);
}
```

```
word value is:1234
high byte is:12
low byte is:34
word value is : ffffff34
```

共用体的主要用途有：

1. 节省内存空间：当多个成员变量不会同时使用，但需要共享同一块内存空间时，可以使用共用体来节省内存。
2. 数据类型转换：共用体可以用于不同类型之间的转换，通过存储一个成员变量，然后通过另一个成员变量来读取转换后的值。

需要注意的是，由于共用体的成员共享同一块内存空间，因此在使用共用体时需要谨慎处理成员的赋值和读取操作，以免出现意外的结果。


THANKS