11 结构体与联合体

固定记录与备选值的最小应用

本讲学习目标

- 表示与处理记录
  - 定义、初始化结构体，访问和复制成员。
  - 用固定数组和循环处理多条记录。
- 函数访问
  - 区分结构体副本与原对象，写简单修改函数。
- 备选值
  - 认识联合体共用存储，按标记选择成员。

复合数据：从单项数据到记录组合

程序 = 数据对象 + 操作 + 控制 + 组合

- 结构体将不同类型的数据成员组合为整体，使程序统一表示和处理一条记录。
- 联合体通过共享存储表示不同的备选值，进一步扩展程序的数据表示能力。

1 结构体记录

用命名成员组成一条记录，区分类型与具体对象

一条记录的不同字段

```c
int id = 101;
char name[20] = "Li Ming";
int score = 80;
printf("%d %s %d\n", id, name, score);
```

同一名学生

学号101、姓名Li Ming、成绩80，输出同一条记录的字段。

已有表示

三个变量类型不同，用名称约定它们属于同一名学生。

数组要求元素类型相同，不能直接把这三个字段作为元素。

本讲问题

怎样把这些字段组成一条记录，统一保存和复制?

结构体类型与成员

```
1 struct Student {
2     int id;
3     char name[20];
4     int score;
5 };
```

类型定义

本例用struct定义类型struct Student。

定义规定记录由哪些成员组成，还未创建学生变量。

成员

id、name、score是成员，各有名称和类型。

name是内部固定数组，保存字符而非地址。

写法

右花括号后保留分号；姓名容量包含终止符。

结构体变量的初始化

```
1 struct Student s = {
2     101, "Li Ming", 80
3 };
4 struct Student t = {
5     102, "Chen Yu", 60
6 };
```

具体变量

s、t是同一类型的两个独立结构体对象。

初始化项依次对应id、name、score。

固定数组成员

"Li Ming"及终止符保存在s.name中。

s.name容量20，当前字符串长度7。

类型与对象

类型规定组成；s、t各自保存成员值。

访问和修改结构体成员

```
1 struct Student s = {101, "Li Ming", 80};
2 s.score = 85;
3 printf("%d %s %d\n",
4         s.id, s.name, s.score);
5 strcpy(s.name, "Lin Tao");
6 printf("%s\n", s.name);
```

成员访问运算符.

s.score访问s的成绩，可像int变量一样赋值。

第一段输出101 Li Ming 85及换行。

数组成员

s.name像已有字符数组一样使用。

strcpy声明在string.h中，目标容量足够。

最后输出Lin Tao。

数组不能用s.name = "Lin Tao"赋值。

练习：复制记录后的变化

```
1 struct Student a = {101, "Li Ming", 80};
2 struct Student b;
3 b = a;
4 b.score = 90;
5 b.name[0] = 'X';
```

独立判断

写出a、b各自的姓名与成绩。

b.name[0]修改的是哪个对象的数组?

调用前提

Student成员为int与固定char数组。

姓名均有终止符，修改下标0有效。

解析：复制记录后的变化

```
1 struct Student a = {101, "Li Ming", 80};
2 struct Student b;
3 b = a;
4 b.score = 90;
5 b.name[0] = 'X';
6 printf("%s %d\n", a.name, a.score);
7 printf("%s %d\n", b.name, b.score);
```

核对输出

第一行Li Ming 80；第二行Xi Ming 90。

整体赋值

相同类型结构体复制各成员值，固定数组成员也随记录复制。

两个对象

a.name与b.name是各自对象内的独立数组。

本例修改b，不改变a。

练习：定义一条商品记录

- 字段要求
  - 编号为int，名称最多19个短ASCII字符，价格为double。
- 独立编写
  - 定义struct Product，成员名为id、name、price。
  - 创建并初始化变量item：编号201，名称Pen，价格2.5。
  - 写出把价格改为3.0的语句。
- 可用支持
  - 可以查struct语法；名称容量必须包含终止符。

解析：定义一条商品记录

```
1 struct Product {
2     int id;
3     char name[20];
4     double price;
5 };
6 struct Product item = {201, "Pen", 2.5};
7 item.price = 3.0;
```

字段与类型

19个字符再加终止符，name容量至少20。

商品价格按题意使用double。

初始化与修改

初始化项与成员顺序对应。

用item.price访问该商品的价格。

核对自己的代码

允许更大固定容量；成员名与题意一致即可。

2 固定记录数组

用已有数组与循环处理多条记录

结构体数组中的记录

```
struct Student classroom[3] = {
    {101, "Li Ming", 80},
    {102, "Chen Yu", 60},
    {103, "Wang Lei", 90}
};
printf("%s %d\n", classroom[1].name,
       classroom[1].score);
```

数组元素

classroom的每个元素是一整条Student记录。

有效下标是0、1、2。

先定位记录，再访问成员

classroom[1]是第二条记录。

classroom[1].score是它的成绩，输出Chen Yu 60。

与平行数组的区别

姓名和成绩属于同一个数组元素。

遍历记录并统计成绩

```
1 int count = 0;
2 for (int i = 0; i < 3; ++i) {
3     if (classroom[i].score >= 80) {
4         ++count;
5     }
6 }
7 printf("%d\n", count);
```

判断成员

依次取80、60、90，与80比较。

下标0与2满足条件，最终输出2。

复用已学循环

循环仍访问数组元素，只是条件使用其score成员。

只访问i < 3的记录；恰好80也应计入。

练习：自主统计达标记录

```
int count_at_least(
    const struct Student data[],
    int n, int threshold);
struct Student data[3] = {
    {101, "Li Ming", 72},
    {102, "Chen Yu", 85},
    {103, "Wang Lei", 60}
};
```

独立写完整函数

统计前n条记录中score >= threshold的数量。

data含至少n条记录，n非负。

函数不修改记录。

核对输入

成绩为72、85、60，n为3，threshold为80。

再检查threshold为60及n为0时的返回值。

可用支持

接口和条件已给，循环与成员判断由你编写。

解析：自主统计达标记录

```
1  int count_at_least(
2      const struct Student data[],
3      int n, int threshold) {
4      int count = 0;
5      for (int i = 0; i < n; ++i) {
6          if (data[i].score >= threshold) {
7              ++count;
8          }
9      }
10     return count;
11 }
```

三项结果

threshold为80时返回1；为60时返回3。

n为0时不进入循环，返回0。

核对自己的实现

访问data[i].score，循环不超过有效记录数。

用>=保留等于阈值的记录；接受等价写法。

const

接口限定通过data不修改记录。

用三次调用核对统计函数

```
struct Student data[3] = {
    {101, "Li Ming", 72},
    {102, "Chen Yu", 85},
    {103, "Wang Lei", 60}
};
printf("%d\n", count_at_least(data, 3, 80));
printf("%d\n", count_at_least(data, 3, 60));
printf("%d\n", count_at_least(data, 0, 80));
```

函数与调用

沿用前页函数，三次都传入data。

核对结果

输出依次为1、3、0；第二次把60也计入。

第三次n为0，不访问任何数组元素。

对照自己的实现

用自己的函数运行这些调用，核对结果。

3 结构体与函数

区分参数副本与原对象，修改已有记录

指向已有结构体的指针

```
1 struct Student s = {101, "Li Ming", 80};
2 struct Student *p = &s;
3 p->score = 85;
4 printf("%d\n", s.score);
5 (*p).score = 90;
```

保存地址

p保存s的地址，未创建另一条Student记录。

p必须指向有效对象，才可访问其成员。

间接成员访问运算符->

p->score访问p指向的记录的成绩。

输出85；最后s.score变为90。

等价写法

p->score等价于(*p).score，括号不能省略。

按值参数中的记录副本

```
1 void change_copy(struct Student x) {
2     x.score = 100;
3 }
4 // 独立调用片段
5 struct Student s = {101, "Li Ming", 80};
6 change_copy(s);
7 printf("%d\n", s.score);
```

参数副本

x获得s各成员值的副本。

函数内x.score变为100，s.score仍为80。

访问方式

x是结构体变量，使用x.score。

返回调用处

最后输出80；按值修改不能更新原记录。

通过指针参数修改原记录

```c
void change_original(struct Student *p) {
    p->score = 100;
}
// 独立调用片段
struct Student s = {101, "Li Ming", 80};
change_original(&s);
printf("%d\n", s.score);
```

参数保存地址

&s把原记录的地址传入函数，p指向s。

p->score修改原记录，最后输出100。

传值规则保持

指针参数本身也接收地址值的副本。

通过该地址访问的仍是同一个s。

前提

本例调用保证p指向有效可写对象。

练习：两次调用修改谁

```c
void change_copy(struct Student x) {
    x.score = 100;
}
void change_original(struct Student *p) {
    p->score = 100;
}
struct Student t = {102, "Chen Yu", 60};
change_copy(t);
printf("%d\n", t.score);
change_original(&t);
printf("%d\n", t.score);
```

独立判断

两行输出各是什么?

指出哪次调用只改副本，哪次改t。

可用支持

沿用前两页函数，各把score设为100。

参数分别为结构体值与对象地址。

解析：两次调用修改谁

```
1 struct Student t = {102, "Chen Yu", 60};
2 change_copy(t);
3 printf("%d\n", t.score); // 输出60
4 change_original(&t);
5 printf("%d\n", t.score); // 输出100
```

按值调用

change_copy(t)修改参数x的副本，t.score仍为60。

通过地址访问

change_original(&t)使p指向t，修改原对象，输出100。

指针参数也传值

p保存地址副本；通过它访问原对象，不复制新记录。

练习：自主修改记录成员

```
1 void set_score(struct Student *p,
2                 int value);
3 struct Student t = {102, "Chen Yu", 60};
```

独立写完整函数

把p所指记录的score设为value，其他成员保持。

调用者保证p指向有效可写的Student对象。

写出调用并判断

t的学号102，姓名Chen Yu，成绩60。

把成绩设为75，写出调用及修改后的各成员。

可用支持

接口已给；成员访问与赋值由你完成。

解析：自主修改记录成员

```
1 void set_score(struct Student *p,
2                 int value) {
3     p->score = value;
4 }
5 // 独立调用片段
6 struct Student t = {102, "Chen Yu", 60};
7 set_score(&t, 75);
```

修改目标

p->score访问原对象指定成员。

(*p).score = value也是等价写法。

调用结果

t仍为102、Chen Yu，score变为75。

不为p申请空间，也不释放已有记录。

4 联合体

用共用存储保存备选值，保持标记与成员一致

联合体成员共用存储

```c
union Value {
    int count;
    double reading;
};
union Value v;
v.count = 3;
printf("%d\n", v.count);
v.reading = 2.5;
printf("%.1f\n", v.reading);
```

备选成员

union定义联合体，成员共用同一块存储。

本例先保存整数计数，再改为小数测量值。

当前值

两行输出为3和2.5；后一次保存改变当前值类别。

不能把v.count当作仍独立保存着3。

本讲读取约定

读取与当前保存值相对应的成员。

结构体与联合体的存储关系

比较项	结构体	联合体
成员存储	各成员保存各自字段	各成员共用存储
当前应用	同时保存学号、姓名和成绩	保存整数计数或小数测量
成员更新	指定成员变更，其他字段保持	改变共用存储中的当前值
存储大小	为所有成员及必要填充提供空间	容纳最大成员，并考虑布局要求

语法相似不表示存储关系相同；具体sizeof由实现决定，不直接相加成员大小。

标记与联合体成员

```
1 struct Measurement {
2     char kind;
3     union Value value;
4 };
5 struct Measurement m;
6 m.kind = 'r';
7 m.value.reading = 2.5;
```

应用约定

kind为'c'表示整数count，为'r'表示小数reading。

kind和value是两个独立成员，同时保存。

保存一项测量

本例标记'r'，对应reading为2.5。

m.value先定位联合体，再用.reading访问其成员。

更新责任

程序同时设置标记和对应值，联合体不自动记类别。

按标记读取对应成员

```c
if (m.kind == 'c') {
    printf("%d\n", m.value.count);
} else if (m.kind == 'r') {
    printf("%.1f\n", m.value.reading);
} else {
    puts("Unknown kind");
}
```

分支对应

'c'读取整数count，'r'读取小数reading。

上页的m选择'r'分支，输出2.5。

保存与读取一致

只有标记和值对应，分支才读取预期成员。

未知标记输出提示，避免猜测值的类别。

练习：修正标记与成员

```
1 struct Measurement sample;
2 sample.kind = 'c';
3 sample.value.reading = 7.0;
```

已知目标

要保存整数计数7，'c'约定对应count。

独立修正

改写最后一行，使保存成员与标记一致。

写出按标记读取时应使用的成员和输出。

边界

不运行失配原版，也不猜测其整数输出。

解析：修正标记与成员

```
1 struct Measurement sample;
2 sample.kind = 'c';
3 sample.value.count = 7;
4 printf("%d\n", sample.value.count);
```

修正保存成员

'c'对应count，因此写入整数成员count。

按标记选择同一成员读取，输出7。

解释原版问题

标记宣称保存计数，实际却写入reading。

联合体不自动把7.0转换并保存为整数7。

应用边界

跨成员解释存储字节不进入本讲。

本讲总结

- 记录表示
  - 结构体类型规定成员；具体变量保存一条记录。
  - 固定数组成员随结构体整体赋值复制。
- 记录处理
  - 数组下标定位记录，.访问其成员。
  - 按值参数改副本，指针参数可用->改原对象。
  - 能写出固定数组统计与简单成员修改函数。
- 备选值
  - 联合体成员共用存储；标记与当前保存成员相配。
  - 按标记读取对应成员。

教材对应与阅读要求

本讲内容	C Primer Plus第六版	阅读要求
类型、变量与成员	§14.2 ~ 14.3	重点阅读基本定义和访问
固定记录数组	§14.4	阅读下标与成员定位
指针、函数及复制	§14.6 ~ 14.7.4	重点区分副本与原对象
联合体及简单标记	§14.10 ~ 14.10.1	了解备选存储与用法

本讲成员使用标量与固定数组；动态成员、链表与文件不进入当前学习要求。