第 十 六 讲

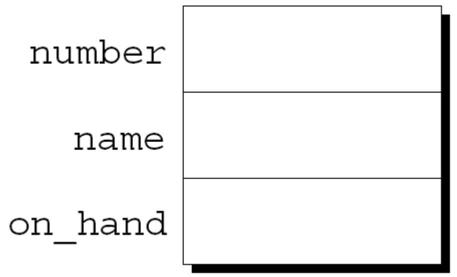
结 构 、 联 合 和 枚 举

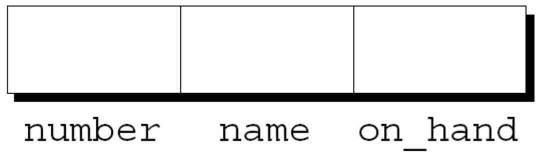
结 构 变 量

数组：所有成员类型相同

结构：所有成员类型可以不同　　每个成员有自己的**名字**　　**成员**：字段

```
struct {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
} part1, part2;
```





内存中紧挨着存储，
每个成员有自己的空间

结 构 变 量

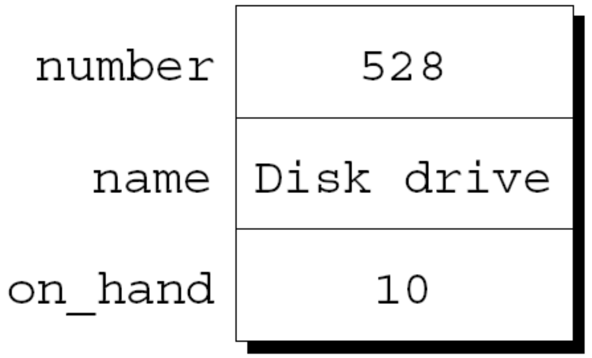
独立命名空间

```
struct {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
} part1, part2;
```

```
struct {
    char name[NAME_LEN+1];
    int number;
    char sex;
} employee1, employee2;
```

初始化

```
struct {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
} part1 = {528, "Disk drive", 10},
  part2 = {914, "Printer cable", 5};
```



```
{.number = 528, .name = "Disk drive", .on_hand = 10}
```

结 构 变 量

访问

```c
printf("Part number: %d\n", part1.number);
printf("Part name: %s\n", part1.name);
printf("Quantity on hand: %d\n", part1.on_hand);
```

```c
part1.number = 258;
  /* changes part1's part number */
part1.on_hand++;
  /* increments part1's quantity on hand */
```

- . 的优先级几乎高于其他所有运算符

```c
scanf("%d", &part1.on_hand);
```

```c
part2 = part1;
```

复制

数组能复制吗?

```c
int a[10],b[10];
a=b;
```

```c
struct { int a[10]; } a1, a2;
a1 = a2;
```

结 构 类 型

结构标记

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
};
```

```
struct part part1, part2;
```

```
part part1, part2;   /*** WRONG ***/
```

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
} part1, part2;
```

结构标记的定义

```
typedef struct {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
} Part;
```

```
Part part1, part2;
```

结 构 类 型

作为参数

```
void print_part(struct part p)
{
  printf("Part number: %d\n", p.number);
  printf("Part name: %s\n", p.name);
  printf("Quantity on hand: %d\n", p.on_hand);
}
```

复合字面量

```
print_part((struct part) {528, "Disk drive", 10});
```

作为返回值

```
struct part build_part(int number,
                       const char *name,
                       int on_hand)
{
  struct part p;

  p.number = number;
  strcpy(p.name, name);
  p.on_hand = on_hand;
  return p;
}
```

```
part1 = build_part(528, "Disk drive", 10);
```

嵌 套 的 数 组 和 结 构

嵌套的结构

```c
struct person_name {
  char first[FIRST_NAME_LEN+1];
  char middle_initial;
  char last[LAST_NAME_LEN+1];
};

struct student {
  struct person_name name;
  int id, age;
  char sex;
} student1, student2;


strcpy(student1.name.first, "Fred");
```

结构数组

```c
struct part inventory[100];

inventory[i].number = 883;

inventory[i].name[0] = '\0';
```

嵌 套 的 数 组 和 结 构

初始化

```
struct dialing_code {
 char *country;
 int code;
};
```

```
const struct dialing_code country_codes[] =
 {{"Argentina",            54}, {"Bangladesh",      880},
  {"Brazil",               55}, {"Burma (Myanmar)",  95},
  {"China",                86}, {"Colombia",         57},
  {"Congo, Dem. Rep. of", 243}, {"Egypt",            20},
  {"Ethiopia",            251}, {"France",           33},
  {"Germany",              49}, {"India",            91},
  {"Indonesia",            62}, {"Iran",             98},
  {"Italy",                39}, {"Japan",            81},
  {"Mexico",               52}, {"Nigeria",         234},
  {"Pakistan",             92}, {"Philippines",      63},
  {"Poland",               48}, {"Russia",            7},
  {"South Africa",         27}, {"South Korea",      82},
  {"Spain",                34}, {"Sudan",           249},
  {"Thailand",             66}, {"Turkey",           90},
  {"Ukraine",             380}, {"United Kingdom",   44},
  {"United States",         1}, {"Vietnam",          84}};
```

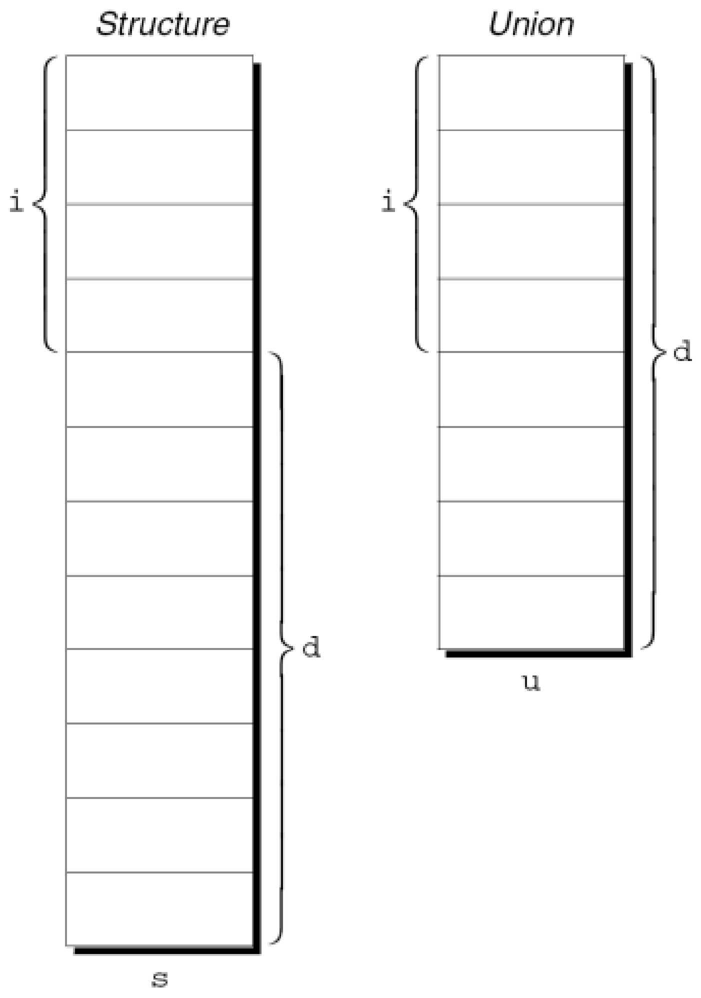
示例：维护零件库　　　01-inventory.c

联 合

多个成员构成

```
union {
    int i;
    double d;
} u;
```

为最大的那个成员分配足够空间



空间共享

```
u.i = 82;

u.d = 74.8;

union {
    int i;
    double d;
} u = {0};

union {
    int i;
    double d;
} u = {.d = 10.0};
```

联 合

用联合来节省空间

```
struct catalog_item {
int stock_number;
double price;
int item_type;
union {
  struct {
    char title[TITLE_LEN+1];
    char author[AUTHOR_LEN+1];
    int num_pages;
  } book;
  struct {
    char design[DESIGN_LEN+1];
  } mug;
  struct {
    char design[DESIGN_LEN+1];
    int colors;
    int sizes;
  } shirt;
} item;
};
```

```
printf("%s", c.item.book.title);
```

```
strcpy(c.item.mug.design, "Cats");
```

```
printf("%s", c.item.shirt.design);
/* prints "Cats" */
```

枚 举

```
enum {CLUBS, DIAMONDS, HEARTS, SPADES} s1, s2;
```

```
enum suit {CLUBS, DIAMONDS, HEARTS, SPADES};        enum suit s1, s2;
```

```
typedef enum {CLUBS, DIAMONDS, HEARTS, SPADES} Suit;     Suit s1, s2;
```

```
typedef enum {FALSE, TRUE} Bool;
```

作为整数来处理

```
enum {CLUBS, DIAMONDS, HEARTS, SPADES} s1, s2;
         0       1         2       3
```

```
int i;
enum {CLUBS, DIAMONDS, HEARTS, SPADES} s;

i = DIAMONDS;     /* i is now 1              */
s = 0;            /* s is now 0 (CLUBS)      */
s++;              /* s is now 1 (DIAMONDS)   */
i = s + 2;        /* i is now 3              */
```