程序设计与算法基础

程红蓉

信息与软件工程学院

课程主要内容

- 第1章 C语言概述
- 第2章 C语言基本概念
- 第3章 格式化输入/输出
- 第4章 表达式
- 第5章 选择语句
- 第6章 循环
- 第7章 基本类型
- 第8章 数组
- 第9章 函数
- 第10章 程序结构
- 第11章 指针
- 第12章 指针和数组
- 第13章 字符串
- 第14章 预处理
- 第15章 编写大型程序
- 第16章 结构、联合和枚举
- 第22章 输入/输出

本章要点

- 结构变量
- 结构类型
- 嵌套的数组和结构
- 联合
- 枚举
- 链表

16.1 结构变量

数组	结构
✓同类型数据的有序集合	✓相关数据的有序集合
✓数据项：元素，类型相同	✓数据项：成员，类型不尽相同
✓通过下标访问元素	✓通过名字访问成员

16.1.1 结构变量的声明

- 结构类型的定义

```
struct  结构标记{
        成员的定义;
}
```

注意：

1. 结构标记不是数据类型，仅作标记使用。
2. 成员的定义与变量声明类似。

16.1.1 结构变量的声明

- 举例1：为存储零部件，声明两个结构变量 教材P.267

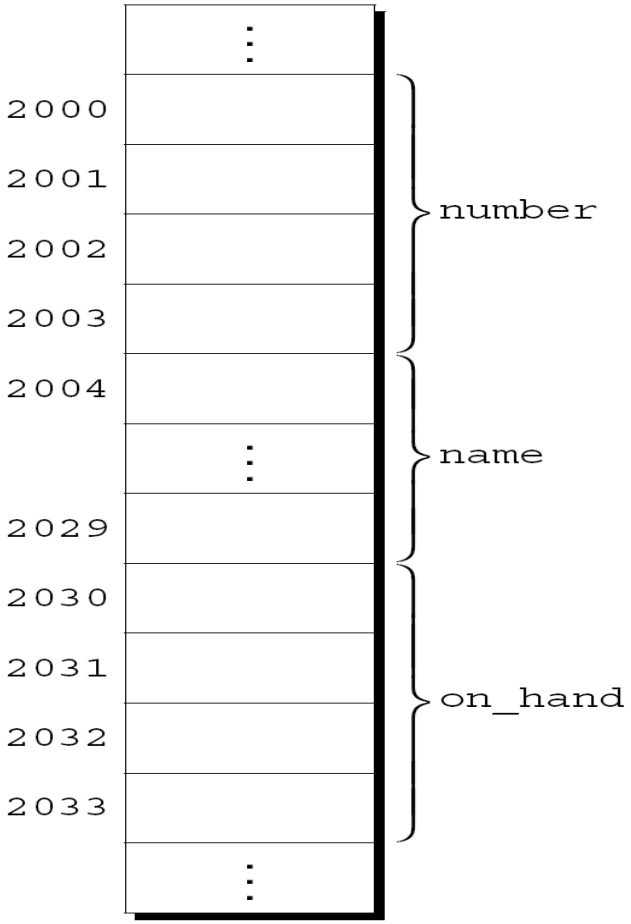
```
#define NAME_LEN 25
struct part{
        int number;   //编号
        char name[NAME_LEN+1];
        int on_hand;  //数量
} part1, part2;
```

```
#define NAME_LEN 25
struct {
        int number;
        char name[NAME_LEN+1];
        int on_hand;
} part1, part2;
```

注意：part是结构标记，而part1, part2是结构变量名。

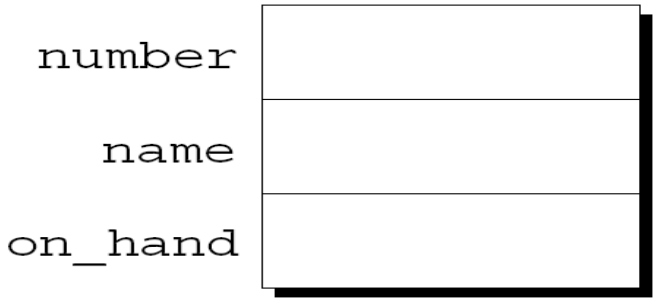
16.1.1 结构变量的声明

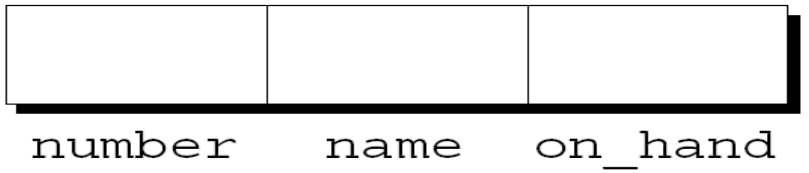
- 结构的成员按声明时的顺序存储在内存中，成员之间没有空隙。 教材P.268
- 举例2：part1 的存储示意



16.1.1 结构变量的声明

- **结构的抽象表示** **教材P.268**





注意：结构的成员名字不可重复。

16.1.1 结构变量的声明

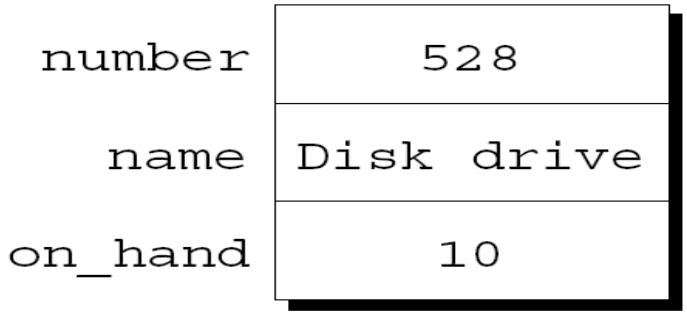
- 每个结构的成员有单独的名字空间 教材P.268
- 举例3

```
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;
```

16.1.2 初始化结构变量

- 声明结构变量时可以初始化　教材P.269

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



16.1.2 初始化结构变量

- 用于结构初始化的表达式**必须是常量**。
- 初始化的成员数可以比结构的成员数少，任何剩余的成员都用**0作为初始值**。

16.1.3 指定初始化(C99)

教材P.269

这部分内容自学

16.1.4 对结构的操作

- **访问结构成员** **教材P.270**
  - ✓ 结构名.成员名（句点.是运算符）
  - ✓ 结构成员可作变量使用

16.1.4 对结构的操作

- 举例4 教材P.270

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

方法一：通过结构变量.成员名的方式。

```
#include <stdio.h>
#define NAME_LEN 25
struct {
        int number;
        char name[NAME_LEN+1];
        int on_hand;
} part1 = {528, "Disk drive", 10};

int main (void) {
    printf("Part number: %d\n", part1.number);
    printf("Part name: %s\n", part1.name);
    printf("Quantity on hand: %d\n", part1.on_hand);
    return 0;
}
```

16.1.4 对结构的操作

- 句点运算符优先级高于其它运算符。
- 举例5

```
scanf("%d", &part1.on_hand);
part1.number = 258;
part1.on_hand++;
```

16.1.4 对结构的操作

- 结构间的赋值（实现结构内容拷贝）
- 举例6

```
struct {
        int number;
```

注意：

1. 只有赋值运算可操作整个结构。
2. 同种类型的结构变量之间才可以赋值。

值给

```
part2 = part1;
```

part1.name 赋值给 part2.name

16.1.4 对结构的操作

- 数组间不能用赋值运算实现内容拷贝

```
int a1[10]={1,2,3,4,5,6,7,8,9,10}, a2[10]={0,0,0};
```

```
a2 = a1;
```

易错情况

- 可借助结构赋值实现数组间内容拷贝

16.1.4 对结构的操作

- 举例7

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

```
……
a2 = a1;
```

注意：a1的成员a赋值给a2的成员a，实现数组间赋值。

16.1.4 对结构的操作

- ==和!= 运算符不能用于结构

- 举例

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

```
if (part1 == part2)
    ……;
```

```
if (part1 != part2)
    ……;
```

易错情况

本章要点

- 结构变量
- 结构类型
- 嵌套的数组和结构
- 联合
- 枚举
- 链表

16.2.1 声明一个结构标记

- 结构标记的声明

```
#define NAME_LEN 25
struct part{
        int number;
        char name[NAME_LEN+1];
        int on_hand;
};  //注意分号不能少
```

易错情况
part是标记，不是数据类型

- 用结构标记声明若干变量

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

```
part part1, part2;
```

16.2.1 声明一个结构标记

- 举例1

```
#define NAME_LEN 25
struct part{
        int number;
        char name[NAME_LEN+1];
        int on_hand;
};  //注意分号不能少
```

```
struct part part1= {528, "Disk drive", 10};
struct part part2;
```

```
part2 = part1;
```

16.2.2 结构类型的定义

- 使用 typedef 定义一个真正的数据类型名。

- 举例2

```
#define NAME_LEN 25
typedef struct {
        int number;
        char name[NAME_LEN+1];
        int on_hand;
} part;  //此时part为类型名
```

```
part part1, part2;
```

教材P.272

16.2.3 结构作为参数和返回值

- 结构作为函数的参数和返回值。
- 举例3 教材P.272

```
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);
}
```

```
print_part(part1);
```

```
#include <stdio.h>
#define NAME_LEN 25
struct {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
} part1 = {528, "Disk drive", 10};
void print_part(struct part p);
int main (void) {
    print_part(part1);
    return 0;
}
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);
}
```

16.2.3 结构作为参数和返回值

- 举例4

```
void f(struct part part1) {
    struct part part2 = part1;
    …
}
```

16.2.3 结构作为参数和返回值

- 举例5 教材P.272

```
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);
```

结构指针

- 声明结构指针

```
#define NAME_LEN 25
struct part{
        int number;
        char name[NAME_LEN+1];
        int on_hand;
} part1;
```

```
struct part  *p1;
```

```
p = &part1;
```

访问结构成员

```
p->number = 202;
```

```
(*p).number = 202;
```

```
#include <stdio.h>
#define NAME_LEN 25
struct part{
        int number;
        char name[NAME_LEN+1];
        int on_hand;
} part1 = {528,” \0” ,10};
int main (void) {
    struct part *p;
    p = &part1;
    printf("Please Enter name\n");
      scanf("%s",p->name);
    printf("Part number: %d\n", p->number);
    printf("Part name: %s\n", p->name);
    printf("Quantity on hand: %d\n", p->on_hand);
    return 0;
}
```

16.2.4 复合字面量(C99)

教材P.273

这部分内容自学

本章要点

- 结构变量
- 结构类型
- 嵌套的数组和结构
- 联合
- 枚举
- 链表

16.3 嵌套数组和结构

- 结构和数组可以无约束地结合。
- 数组可以有结构成员，结构也可以含有数组和结构成员。

16.3.1 嵌套的结构

- 一个结构可以嵌套进另一个结构
- 举例1 教材P.274

```
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;
```

16.3.1 嵌套的结构

- 访问student1的名需两次使用句点. 运算符

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

16.3.1 嵌套的结构

- 举例2 教材P.274

```
display_name(student1.name);

struct person_name  new_name;
    …
student1.name = new_name;
```

16.3.2 结构数组

- **结构数组**是数组和结构一种常见的结合形式，可以作为简单的数据库。教材P.274

- 举例3：存储100种零件的结构数组

```
struct part inventory[100];
```

16.3.2 结构数组

- 对结构数组中的某个元素i可以如下操作：

```
inventory[i].number = 883;
```

```
inventory[i].name[0] = ‘\0’;
```

```
printf_part(inventory[i]);
```

回顾

```
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);
}
```