LECTURE 4 指针和链表

工学院18-19学年秋季学期计算概论（邓习峰班）课后辅导

讲师：陈婉雯

日期：2018/10/27

目录

- 课堂讲义讲解：
- 复习指针和结构体
- 链表

- 上周部分作业讲解与示例

结构体

- 自定义的数据格式，能够更加灵活、高效地表示数据
- 定义结构：描述对象由什么组成（没有创建实际对象）

```
struct stuff {
    int number;
    char code[4];
    float cost;
};
```

struct: 关键字
stuff: 标记（可选）

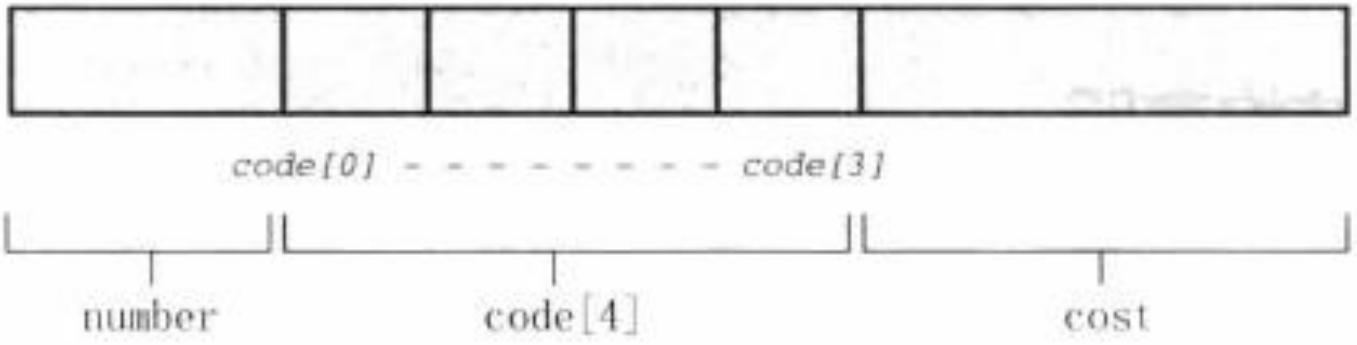


图 14.1 一个结构的内存分配

结构体

- 定义结构变量：创建结构体，分配存储空间
- struct stuff Alex; ← Alex 才是我们定义的结构变量的名称

- 初始化结构：
- 除了直接访问结构成员，还能够这样初始化：

```
struct stuff Alex = {
    20,
    abcd,
    4.5
};
```

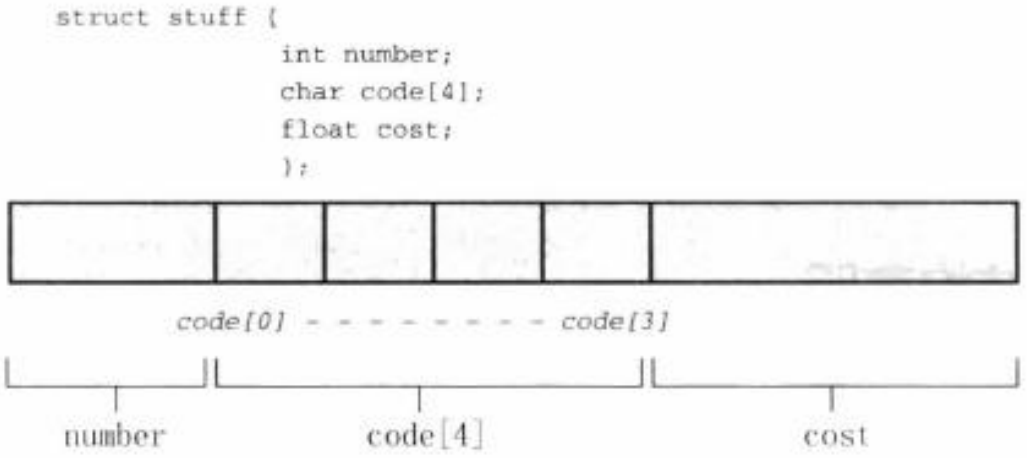


图 14.1 一个结构的内存分配

指针

- 指针：值为内存地址的变量

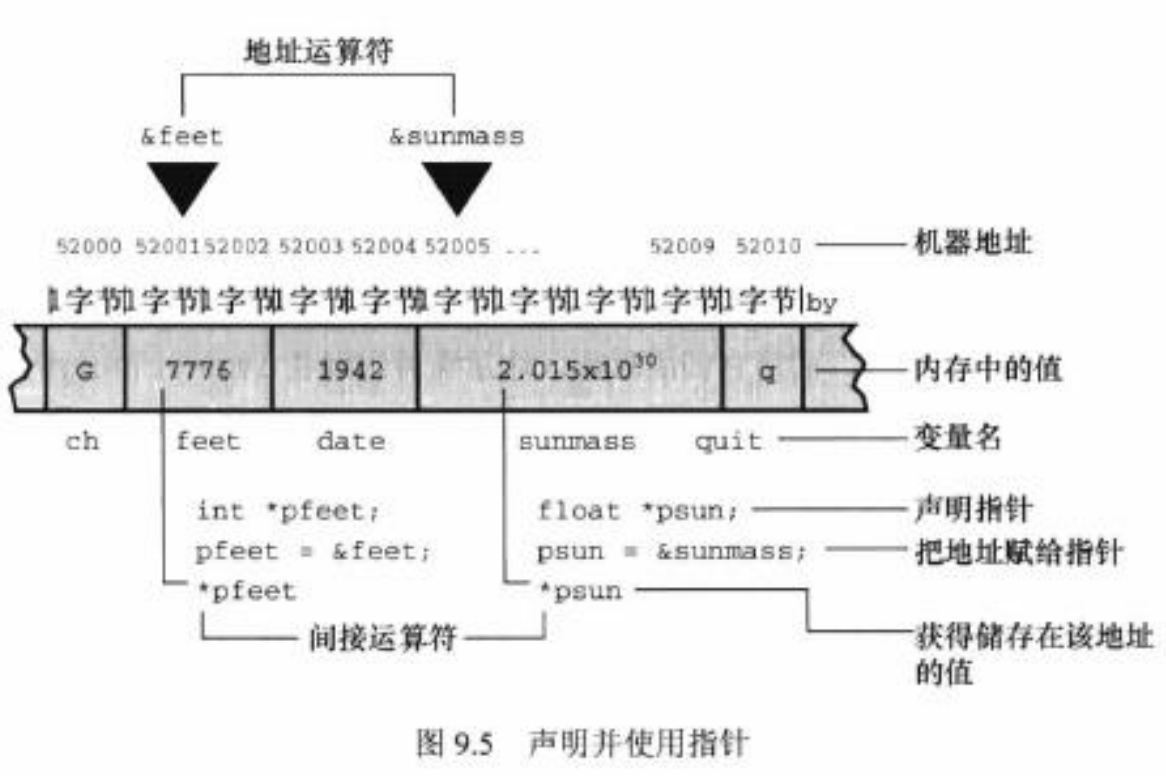


图 9.5 声明并使用指针

指针与内存分配

- int value;
  - 系统自动预留相应的内存空间存储value
- int *add;
  - 我们只知道add会是一个地址（无符号整型），但系统不会自动预留内存空间，add也没有指向特定的内存位置
  - 所以我们需要给add分配内存！

- malloc(size):
  - 找到合适的、大小为size字节的空闲内存块
  - 返回动态分配内存块的首字节地址
  - 要把这个地址赋给指针变量，通过指针访问
  - 要记得强制类型转换（为什么？）

```
double * ptd;
ptd = (double *) malloc(30 * sizeof(double));
```

指向结构体的指针

- 使用办法和指向普通数据类型的指针一样
- 先声明：
  - struct stuff *him;
- 再初始化：
  - struct stuff barney;
  - him=&barney;
  - him=(struct stuff *)malloc(sizeof(struct stuff))

此时并未创建一个新的结构体

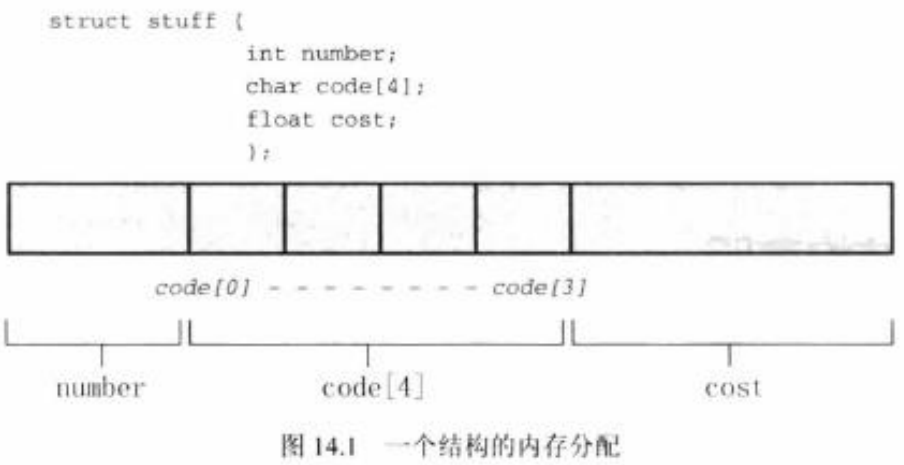


图 14.1 一个结构的内存分配

用指针访问结构体成员

- him=&barney;

- him->number 即是 (*him).number 即是 barney.number
- 必须使用圆括号，因为.运算符比*运算符优先级要高

链式结构

- 通常，一个结构体里面有几个数据项和一两个指向其他同类型结构的指针
- 这些指针把一个结构和另一个结构链接起来
- 提供一种路径能够遍历整个彼此链接的结构

- 例子：
- 队列、栈、二叉树.....

从数组到链表

- 添加数据的量不确定
- 不想让程序分配多余的空间
- 但每次malloc()时分配的内存块不一定连续

链表：

- 为新的指针分配空间

一个例子

```
struct film{
    char title[TSIZE];
    int rating;
    struct film * next;
};
struct film * head;
```

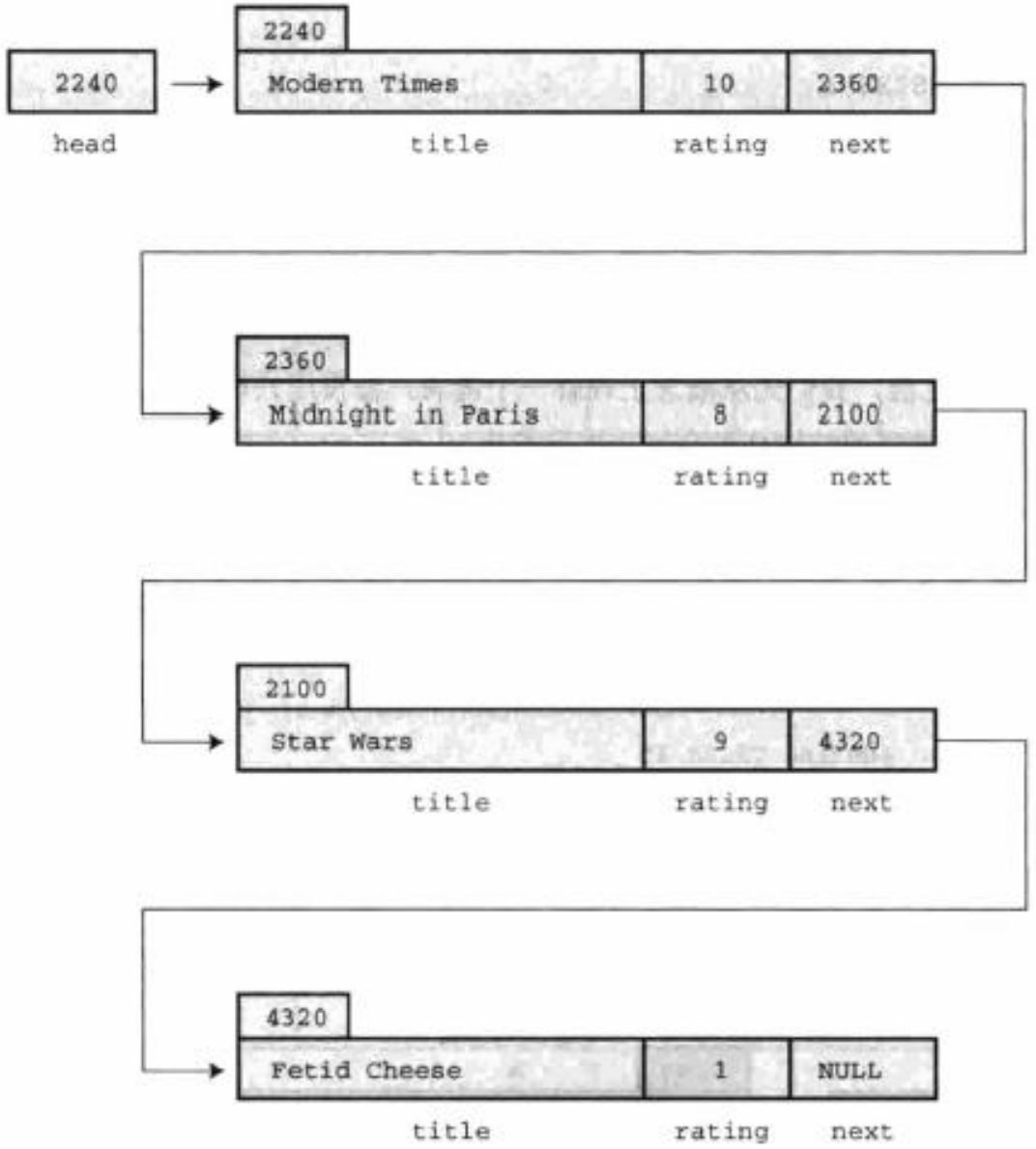


图 17.4 链表中的多个项

使用链表

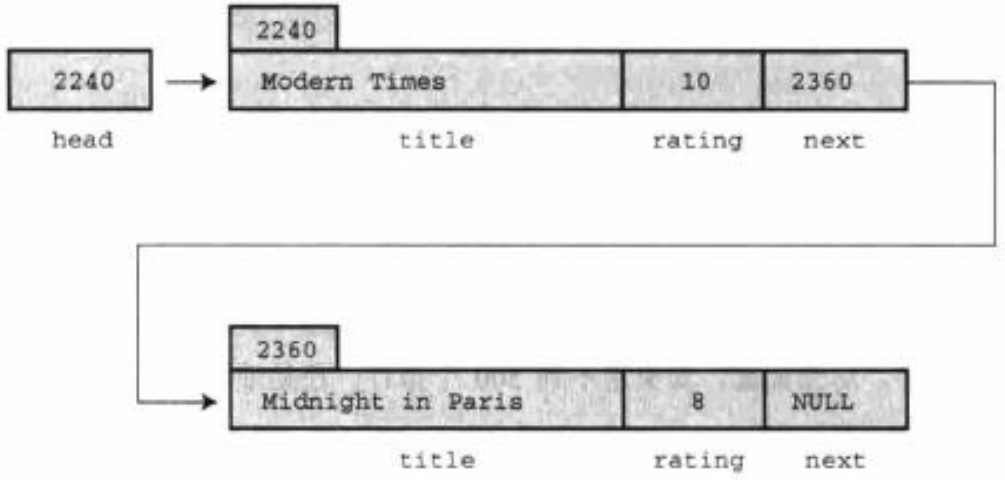
- 创建链表
- 往链表里添加、删除成员
- 显示（遍历）链表
- 删除（释放）链表

创建链表

```
struct film * CreateFilm(int val){
    struct film *head;
    head = (struct film*)malloc(sizeof(struct film));
    head->next=NULL;
    return head;
}
```

增加成员

```
struct film * AddFilm(struct film *pNowFilm, char *filmTitle, int myRating){
    struct film *pTmp;
    pTmp=(struct film*)malloc(sizeof(struct film));
    pTmp->title=filmTitle;
    pTmp->rating=myRating; //存数
    if pNowFilm==NULL{ //在链表结尾添加成员
        pTmp->next=NULL;
        pNowFilm->next=pTmp;
    }
    else{//在链表中间添加成员
        pTmp->next=pNowFilm->next;
        pNowFilm-pTmp;
    }
    return pTmp;}
```



遍历链表

```
void PrintFilm(struct film *head){
    struct film *current;
    current=head->next;
    while(current!=NULL){
        printf("Title is %s, Rating is %d\n",current->title, current->rating);
        current=current->next;
}
```

为什么我们之前一直让链表最后一个元素的指针指向NULL?
-> 方便我们判断链表什么时候结束

删除一个成员、删除整个链表

```
void DelFilmNext(struct film *pNow){  //删除pNow后面的一个成员
    struct film *pDel=pNow->next;
    pNow->next=pDel->next;
    free(pDel);
}

struct film *ClearFilm(struct film *pFirst){
    struct film *pTmp=pFirst;
    struct film *pNow;
    while(pTmp!=NULL){
        pNow=pTmp->next;
        free(pTmp);
        pTmp=pNow;
    }
    return NULL;
}
```

作业思路讲解

- 约瑟夫问题：有n只猴子，按顺时针方向围成一圈选大王（编号从1到n），从第1号开始报数，一直数到m，数到m的猴子退出圈外，剩下的猴子再从1开始报数。直到圈内只剩下一只猴子，这只猴子就是猴王。要求输入n,m之后，输出最后猴王的编号。
- 输入：每行为用空格分开的两个整数，第一个是n，第二个是m，输入0 0时停止。
- 输出：除了最后一行外，输出数据为最后猴王的编号。

- 例子输入：
- 6 2
- 12 4
- 8 3
- 0 0
- 例子输出：
- 5
- 1
- 7

思路

- 每次输入一组n,m，判断是否输入结束
- 如果没有结束输入：对n,m求约瑟夫问题
  - 我们可以把约瑟夫问题单独写成一个函数。
  - 函数输入：n和m，函数输出：猴王编号

- 如何求解约瑟夫问题？
  - 数组法
  - 链表法

作业思路讲解

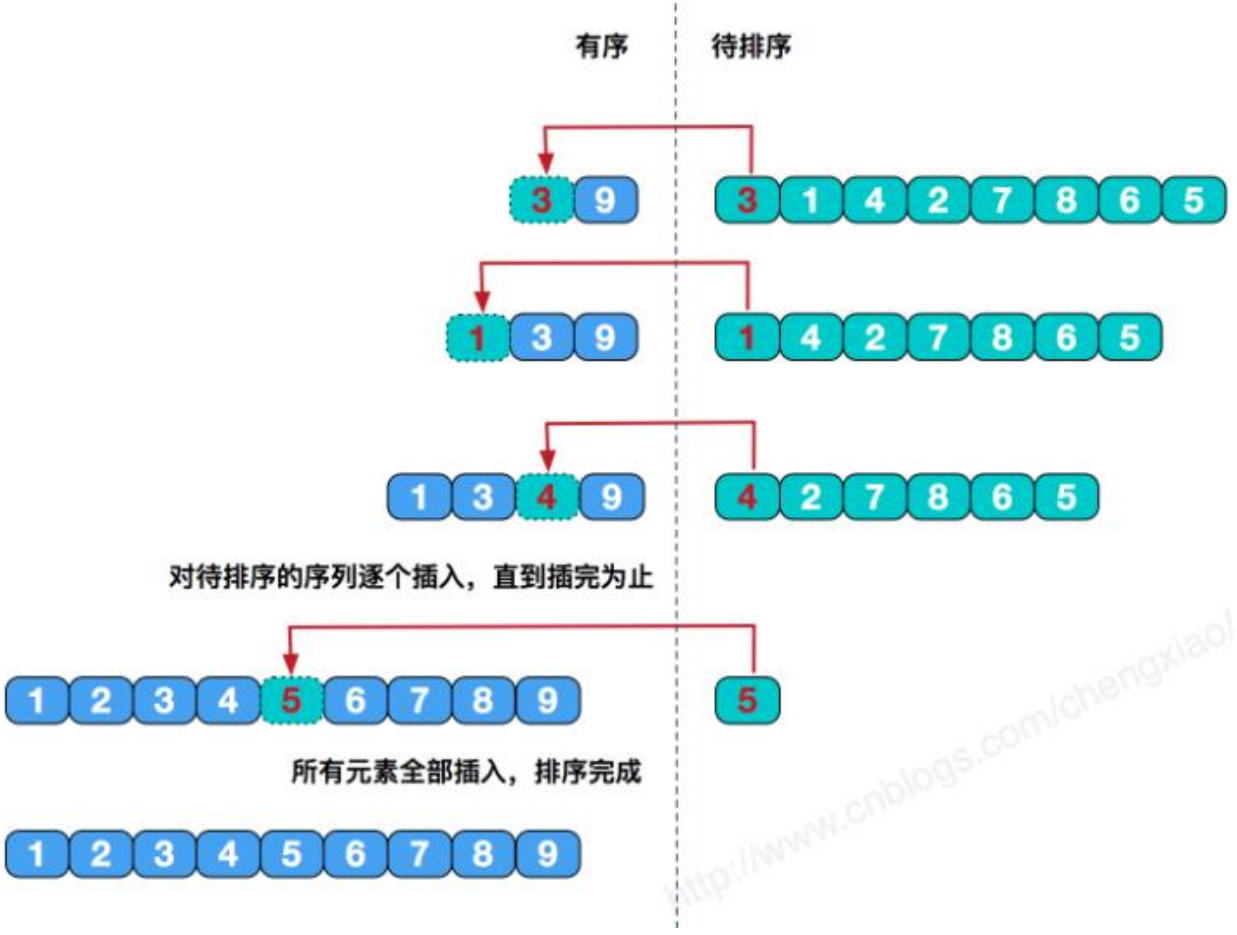
- 折半法查找可能重复的数值：
- 一个长度为N的整数数组，从键盘输入N个可能重复的数据，用插入排序算法对输入数据实现插入排序（升序）。编写折半法查找函数实现查找，返回第一个和最后一个查找值位置。如果值唯一，则第一个和最后一个位置值相同。
- 输入：
- 首先输入N，然后输入N个数值，最后输入查找值。
- 输出：
- 输出折半法查找的数值出现的第一个位置和最后一个位置，中间用<-->隔开。

思路

- 题目很长，但可以分成三个主要部分：
  - 对数组进行插入排序
  - 对某个值用折半法查找第一个位置
  - 对某个值用折半法查找最后一个位置

- 难点：对于有重复元素，应该如何用折半法找第一、最后一个位置？
  - 普通折半法：在找到元素之后就停止；查找区间没有固定的停止长度
  - 修改：我们要把查找区间缩小至长度为1，保证区间内两个数一个为我们要查找的元素，另外一个比我们要查找的元素小/大

- 插入排序：每一步将一个待排序的记录，插入到前面已经排好序的有序序列中去，直到插完所有元素为止。



- 折半法：搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。