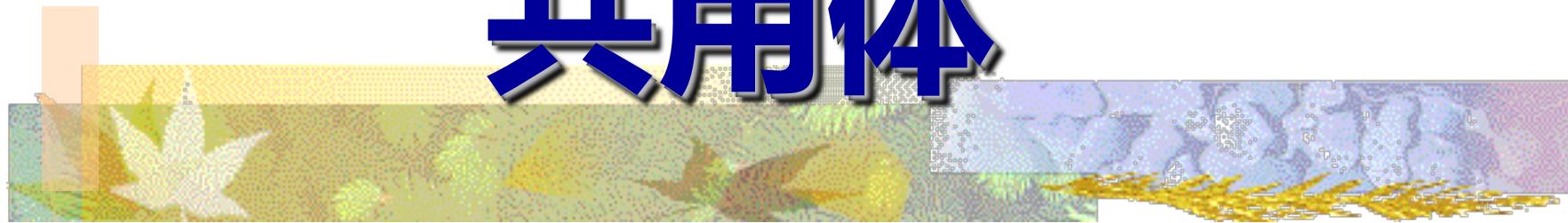




规格严格 功夫到家

第12章 结构体和 共用体



哈尔滨工业大学

计算机科学与技术学院

苏小红

sxh@hit.edu.cn

本章学习内容

- ❖ 结构体数据类型，共用体数据类型，枚举数据类型，定义数据类型的别名
- ❖ 结构体变量、结构体数组、结构体指针的定义和初始化
- ❖ 结构体成员的引用，成员选择运算符，指向运算符
- ❖ 向函数传递结构体变量、结构体数组、结构体指针
- ❖ 动态数据结构、动态链表

12.1 从基本数据类型到抽象数据类型

■ 二进制数——类型本不存在

- 内存里存的内容，你认为它是什么，它就是什么
- 在早期的机器指令及汇编语言中，数据对象均用二进制数表示，没有类型的概念

■ 一般的CPU只支持两种类型

- 整数、浮点数

12.1 从基本数据类型到抽象数据类型

- 在高级语言引入了基本数据类型
 - 整型、浮点型、字符型等
 - 不同语言会定义不同的基本类型
 - 基本数据类型并不能方便地解决所有问题
 - 有些语言（如PL/I）中试图规定较多的类型，如数组、树、栈等，但实践证明不是个好办法

12.1 从基本数据类型到抽象数据类型

- 用户自己构造数据类型-复合数据类型
 - 由基本数据类型迭代派生而来，表示复杂的数据对象
 - 典型的代表就是“结构体”
- 抽象数据类型（Abstract Data Type，简称ADT）
 - 在复合数据类型基础上增加了对数据的操作
- 抽象数据类型进而进化为“类(Class)”
 - 这是一个跨时代的进步
 - Class是Object-Oriented的一个重要概念

12.2.1 为什么要定义结构体类型

- 在程序里表示一个人（姓名、年龄、性别...），怎么表示？
- 想表示多个人呢？
- 如何用计算机程序实现下述表格的管理？



学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
...							

数组的解决方法

```
long studentId[30];           /* 学号 */  
char studentName[30][10];     /* 姓名 */  
char studentSex[30];          /* 性别 */  
int yearOfBirth[30];          /* 出生年 */  
int scoreMath[30];            /* 数学课的成绩 */  
int scoreEnglish[30];         /* 英语课的成绩 */  
int scoreComputer[30];        /* 计算机原理课的成绩 */  
int scoreProgramming[30];     /* 程序设计课的成绩 */
```



学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
...							

数组的解决方法

```
long studentId[30] = {100310121, 100310122, 100310123, 100310124};  
char studentName[30][10] = {"王刚", "李小明", "王丽红", "陈莉莉"};  
char studentSex[30] = {'M', 'M', 'F', 'F'};  
int yearOfBirth[30] = {1991, 1992, 1991, 1992};  
int scoreMath[30] = {72, 88, 98, 87};  
int scoreEnglish[30] = {83, 92, 72, 95};  
int scoreComputer[30] = {90, 78, 89, 78};  
int scoreProgramming[30] = {82, 78, 66, 90};
```



学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
...							

数组的解决方法

数据的内存管理方式

100310121
100310122
100310123
100310124
.....

王刚
李小明
王丽红
陈莉莉
.....

'M'
'M'
'F'
'F'
.....

1991
1992
1991
1992
.....

72
88
98
87
.....

83
92
72
95
.....

90
78
89
78
.....

82
78
66
90
.....

分配内存不集中，寻址效率不高
对数组赋初值时，易发生错位
结构显得零散，不易管理

希望的内存分配图

100310121	100310122	100310123	100310124
王刚	李小明	王丽红	陈莉莉
'M'	'M'	'F'	'F'
1991	1992	1991	1992
72	88	98	87
83	92	72	95
90	78	89	78
82	78	66	90



结构体类型的声明

```
struct student
{
    long studentID;           /* 学号 */
    char studentName[10];     /* 姓名 */
    char studentSex;          /* 性别 */
    int yearOfBirth;          /* 出生年 */
    int scoreMath;            /* 数学课的成绩 */
    int scoreEnglish;         /* 英语课的成绩 */
    int scoreComputer;        /* 计算机原理课的成绩 */
    int scoreProgramming;     /* 程序设计课的成绩 */
};
```

声明了一个结构体类型

结构体的名字
称为结构体标
签(Structure Tag)

构成结构体的变量
称为结构体的成员
(Structure Member)

学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
...							

结构体类型的声明

```
struct student
{
    long studentID;                      /* 学号 */
    char studentName[10];                 /* 姓名 */
    char studentSex;                     /* 性别 */
    int yearOfBirth;                    /* 出生年 */
    int scoreMath;                      /* 数学课的成绩 */
    int scoreEnglish;                   /* 英语课的成绩 */
    int scoreComputer;                  /* 计算机原理课的成绩 */
    int scoreProgramming;               /* 程序设计课的成绩 */
};
```

Don't forget the
semicolon!!

```
struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    int yearOfBirth;
    int score[4];                         /* 4 门课程的成绩 */
};
```

结构体模板
(Structure Template)

形成一个类型声明的样板
用于生成结构体变量
但并未声明结构体变量
因而编译器不为其分配内存

12.2.2 结构体变量的定义

(1) 先定义结构体类型，再定义变量名

```
struct student stu1;
```

(2) 在定义类型的同时定义变量

(3) 直接定义结构体变量（不指定结构体标签）

```
struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    int yearOfBirth;
    int score[4];
} stu1;
```

```
struct
{
    long studentID;
    char studentName[10];
    char studentSex;
    int yearOfBirth;
    int score[4];
} stu1;
```

12.2.3 用typedef定义数据类型

```
typedef struct student STUDENT;
```

```
typedef struct student  
{  
    long studentID;  
    char studentName[10];  
    char studentSex;  
    int yearOfBirth;  
    int score[4];  
} STUDENT;
```

关键字**typedef**为一种
已存在的类型定义一个
别名，并未定义新类型

STUDENT与**struct**
student类型是同义词

struct student stu1, stu2; /*It works*/	
student stu1, stu2;	/*Can this work?*/
struct stu1, stu2;	/*Can this work?*/
STUDENT stu1, stu2;	/*It works!*/



12.2.4 结构体变量的初始化

stu1:

100310121	王刚	M	1991	72	83	90	82
-----------	----	---	------	----	----	----	----

```
STUDENT stu1 = {100310121, "王刚", 'M', 1991, {72,83,90,82}};
```

等价于

```
struct student stu1 = {100310121, "王刚", 'M', 1991, {72,83,90,82}};
```

等价于

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    int yearOfBirth;
    int score[4];
} STUDENT;
```

```
stu1.studentID = 100310121;
strcpy(stu1.studentName, "王刚");
stu1.studentSex = 'M';
stu1.birthday.year = 1991;
stu1.birthday.month = 5;
stu1.birthday.day = 19;
stu1.score[0] = 72;
stu1.score[1] = 83;
stu1.score[2] = 90;
stu1.score[3] = 82;
```



12.2.5 嵌套的结构体

■ 嵌套的结构体（Nested Structure）就是在
一个结构体内包含了另一个结构体作为其成员

```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

学号	姓名	性别	出生日期			数学	英语	计算机原理	程序设计
			年	月	日				

```
STUDENT stu1 = {100310121, "王刚", 'M', {1991,5,19}, {72,83,90,82}};  
STUDENT stu1 = {100310121, "王刚", 'M', {1991,"May",19}, {72, 83, 90, 82}};
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```

```
typedef struct date
{
    int year;
    char month[10];
    int day;
} DATE;
```

结构体定义
可以嵌套

12.2.6 结构体变量的引用

- 访问结构体变量的成员必须使用成员选择运算符（也称圆点运算符）

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

结构体变量名 . 成员名

```
stu1.studentID = 100310121;
```

- 当出现结构体嵌套时，必须以级联方式访问结构体成员

```
stu1.birthday.year = 1991;
stu1.birthday.month = 5;
stu1.birthday.day = 19;
```

12.2.6 结构体变量的引用

【例12.1】演示结构体变量的赋值和引用方法

```
16 int main()
17 {
18     STUDENT stu1 = {100310121, "王刚", 'M', {1991,5,19}, {72,83,90,82}};
19     STUDENT stu2;
20     stu2 = stu1;                                /* 同类型的结构体变量之间的赋值操作 */
21     printf("stu2:%10ld%8s%3d%6d/%02d/%02d%4d%4d%4d%4d\n",
22             stu2.studentID, stu2.studentName, stu2.studentSex,
23             stu2.birthday.year, stu2.birthday.month, stu2.birthday.day,
24             stu2.score[0], stu2.score[1], stu2.score[2], stu2.score[3]);
25     return 0;
26 }
```

按结构体的成员顺序逐一对相应成员进行赋值

stu2: 100310121 王刚 M 1991/05/19 72 83 90 82

格式符%02d中2d前面的前导符0表示输出数据时，若左边有多余位，则补0

【例12.1】若要从键盘输入结构体变量stu1的内容，那么程序如何修改？



```
16 int main()
17 {
18     STUDENT stu1, stu2;
19     int i;
20     printf("Input a record:\n");
21     scanf("%ld", &stu1.studentID);
22     scanf("%s", stu1.studentName); /* 输入学生姓名，无需加& */
23     scanf(" %c", &stu1.studentSex); /* %c 前有一个空格 */
24     scanf("%d", &stu1.birthday.year);
25     scanf("%d", &stu1.birthday.month);
26     scanf("%d", &stu1.birthday.day);
27     for (i=0; i<4; i++)
28     {
29         scanf("%d", &stu1.score[i]);
30     }
31     stu2 = stu1; /* 同类型的结构体变量之间的赋值操作 */
32     printf("&stu2 = %p\n", &stu2); /* 打印结构体变量 stu2 的地址 */
33     printf("%10ld%8s%3c%6d/%02d/%02d%4d%4d%4d%4d\n",
34            stu2.studentID, stu2.studentName, stu2.studentSex,
35            stu2.birthday.year, stu2.birthday.month, stu2.birthday.day,
36            stu2.score[0], stu2.score[1], stu2.score[2], stu2.score[3]);
37     return 0;
38 }
```

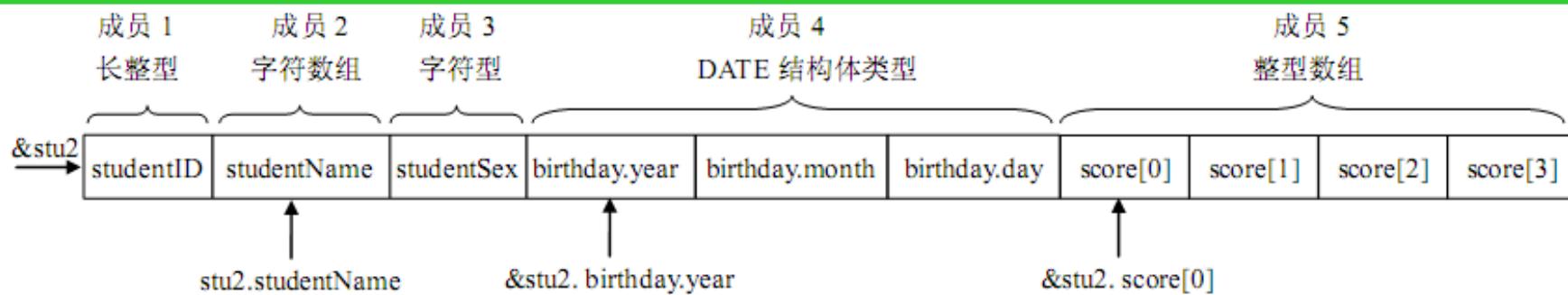
两个地址有何不同？

【例12.1】
若要从键盘输入结构体变量stu1的内容，那么程序如何修改？

```
16 int main()
17 {
18     STUDENT stu1, stu2;
19     int i;
20     printf("Input a record.\n");
21     scanf("%ld", &stu1.studentID);
22     scanf("%s", stu1.studentName); /* 带入字符串时，先输入空格 */
23     scanf(" %c", &stu1.studentSex); /* %~ 前有一个空格 */
24     scanf("%d", &stu1.birthday.year);
25     scanf("%d", &stu1.birthday.month);
26     scanf("%d", &stu1.birthday.day);
27     for (i=0; i<4; i++)
28     {
29         scanf("%d", &stu1.score[i]);
30     }
31     stu2 = stu1; /* 同类型的结构体变量之间的赋值操作 */
32     printf("&stu2 = %p\n", &stu2); /* 打印结构体变量 stu2 的地址 */
```

结构体成员的地址与该成员在结构体中所处的位置及其所占内存的字节数相关

结构体变量的地址
&stu2是该变量所占内存空间的首地址



12.2.7 结构体所占内存的字节数

struct 类型用内存字节数 = ?

【例12.2】

是所有成员变量的内存总和吗？

用运算符**sizeof**获得结构体大小

sizeof(变量或表达式)

sizeof(类型)

```
2  typedef struct sample
3  {
4      char m1;
5      int m2;
6      char m3;
7 }SAMPLE;
```

`printf("%d\n", sizeof(struct sample));`

`printf("%d\n", sizeof(SAMPLE));`

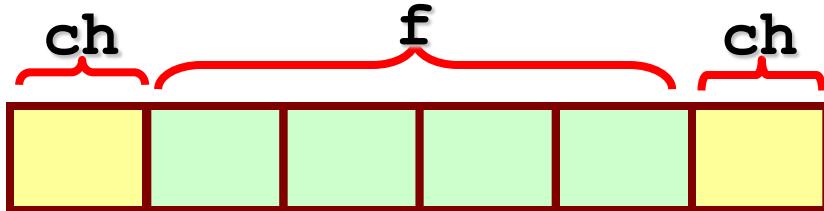


12

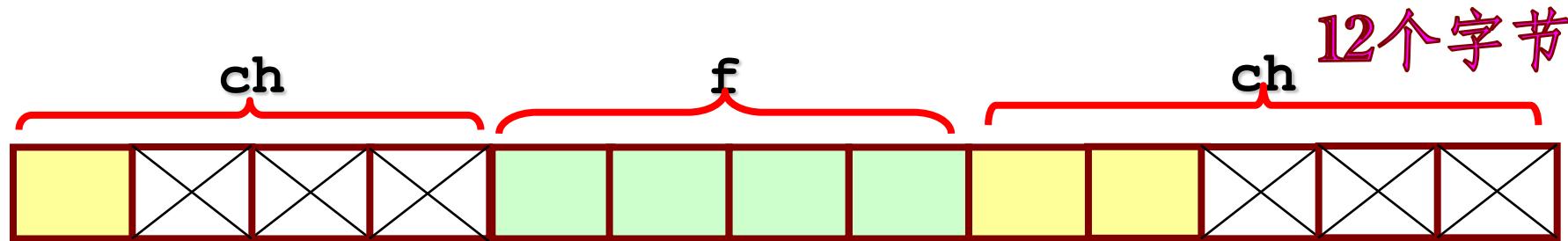
Why?



12.2.7 结构体所占内存的字节数



非所有成员变量的内存总和



事实上，所有数据类型在内存中都是从**偶数**地址开始存放的
且结构所占的实际空间一般是按照机器字长对齐的
不同的编译器、平台，对齐方式会有变化
结构体变量的成员的存储**对齐规则**是与机器相关的
具有特定数据类型的**数据项大小**也是与机器相关的
所以一个结构体在内存中的存储格式也是与机器相关的

12.3 结构体数组的定义和初始化

学号	姓名	性别	出生日期			数学	英语	计算机原理	程序设计
			年	月	日				

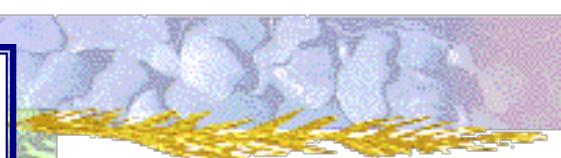
```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```

```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

12.3 结构体数组的定义和初始化

```
STUDENT stu[30] = {{100310121, "王刚", 'M', {1991, 5, 19}, {72, 83, 90, 82}},  
                     {100310122, "李小明", 'M', {1992, 8, 20}, {88, 92, 78, 78}},  
                     {100310123, "王丽红", 'F', {1991, 9, 19}, {98, 72, 89, 66}},  
                     {100310124, "陈莉莉", 'F', {1992, 3, 22}, {87, 95, 78, 90}}}  
};
```

学号	姓名	性别	出生日期			数学	英语	计算机原理	程序设计
			年	月	日				
1	王刚	M	1991	5	19	72	83	90	82
2	李小明	M	1992	8	20	88	92	78	78
3	王丽红	F	1991	9	19	98	72	89	66
4	陈莉莉	F	1992	3	22	87	95	78	90



【例12.3】利用 结构体数组计 算每个学生的 平均分

```
16 int main()
17 {
18     int i, j, sum[30];
19     STUDENT stu[30] = {{100310121,"王刚",'M',{1991,5,19},{72,83,90,82}},
20                             {100310122,"李小明",'M',{1992,8,20},{88,92,78,78}},
21                             {100310123,"王丽红",'F',{1991,9,19},{98,72,89,66}},
22                             {100310124,"陈莉莉",'F',{1992,3,22},{87,95,78,90}}
23                         };
24     for (i=0; i<4; i++)
25     {
26         sum[i] = 0;
27         for (j=0; j<4; j++)
28         {
29             sum[i] = sum[i] + stu[i].score[j];
30         }
31         printf("%10d%8s%3c%6d/%02d/%02d%4d%4d%4d%4d%6.1f\n",
32                 stu[i].studentID,
33                 stu[i].studentName,
34                 stu[i].studentSex,
35                 stu[i].birthday.year,
36                 stu[i].birthday.month,
37                 stu[i].birthday.day,
38                 stu[i].score[0],
39                 stu[i].score[1],
40                 stu[i].score[2],
41                 stu[i].score[3],
42                 sum[i]/4.0);
43     }
44     return 0;
45 }
```

100310121	王刚	M	1991/05/19	72	83	90	82	81.8
100310122	李小明	M	1992/08/20	88	92	78	78	84.0
100310123	王丽红	F	1991/09/19	98	72	89	66	81.3
100310124	陈莉莉	F	1992/03/22	87	95	78	90	87.5

12.4 结构体指针的定义和初始化

- 如何定义指向结构体变量的指针？

```
STUDENT stu1;
```

```
STUDENT *pt;
```

```
pt = &stu1;
```

等价于

```
STUDENT *pt = &stu1;
```



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```



12.4 结构体指针的定义和初始化

- 如何访问结构体指针变量所指向的结构体成员呢？

```
STUDENT stu1;
```

```
STUDENT *pt = &stu1;
```

- 通过stu1和成员选择运算符访问结构体成员

```
stu1. studentID = 1;
```

- 通过pt和指向运算符访问结构体成员

```
(*pt). studentID = 1;
```

```
pt -> studentID = 1;
```



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```



12.4 结构体指针的定义和初始化

当结构体嵌套时，如何访问结构体指针变量所指向的结构体成员？

```
STUDENT stu1;
```

```
STUDENT *pt = &stu1;
```

```
stu1. birthday. year = 1999;
```

```
(*pt). birthday. year = 1999;
```

```
pt -> birthday. year = 1999;
```

stu1
成员1
成员2
成员3
成员4
成
员
5



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```

12.4 结构体指针的定义和初始化

■ 如何定义指向结构体数组的指针？

```
STUDENT stu[30];
```

```
STUDENT *pt;
```

```
pt = stu;
```

等价于

```
STUDENT *pt = stu;
```

等价于

```
STUDENT *pt = &stu[0];
```



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```

stu[30]
pt → stu[0]

stu[1]

stu[2]

stu[3]

stu[4]

stu[5]

.....

stu[29]

12.4 结构体指针的定义和初始化

- 如何访问结构体数组指针指向的结构体成员？

```
STUDENT stu[30];
```

```
STUDENT *pt = stu;
```

使用 `pt++`, 使 `pt` 指向 `stu[1]`

`pt -> studentID`

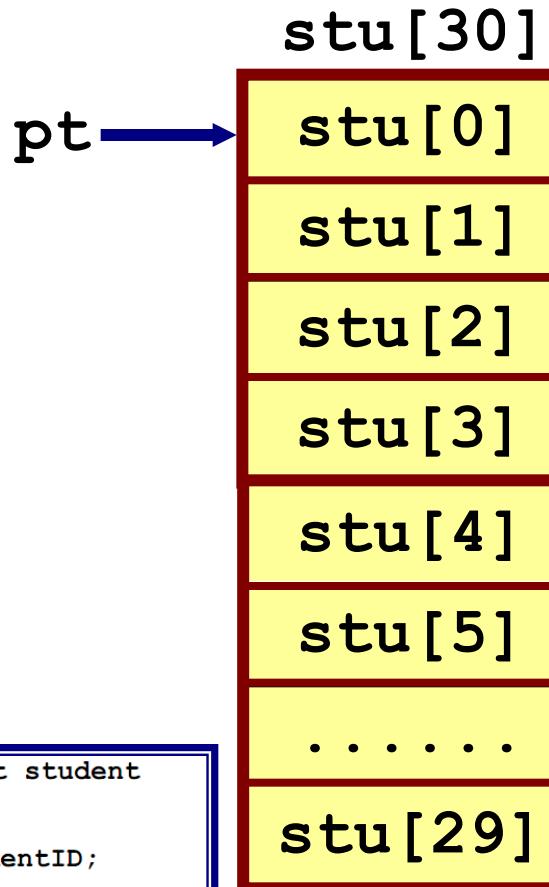
等价于

`stu[1]. studentID`



```
typedef struct date
{
    int year;
    int month;
    int day;
} DATE;
```

```
typedef struct student
{
    long studentID;
    char studentName[10];
    char studentSex;
    DATE birthday;
    int score[4];
} STUDENT;
```



12.5 向函数传递结构体

■ 向函数传递结构体的单个成员

- 复制单个成员的内容
- 函数内对结构内容的修改不影响原结构

■ 向函数传递结构体的完整结构

■ 向函数传递结构体的首地址

```
struct date
{
    int year;
    int month;
    int day;
};

void Func(struct date p)
{
    p.year = 2000;
    p.month = 5;
    p.day = 22;
}
```

结构体变量 作函数参数

【例12.4】

Before function call:1999/04/23

After function call:1999/04/23

```
14 int main()
15 {
16     struct date d;
17     d.year = 1999;
18     d.month = 4;
19     d.day = 23;
20     printf("Before function call:%d/%02d/%02d\n", d.year, d.month, d.day);
21     Func(d);          /* 结构体变量作函数实参，传值调用 */
22     printf("After function call:%d/%02d/%02d\n", d.year, d.month, d.day);
23     return 0;
24 }
```

```

struct date
{
    int year;
    int month;
    int day;
};

void Func(struct date *p)
{
    p->year = 2000;
    p->month = 5;
    p->day = 22;
}

```

```

14 int main()
15 {
16     struct date d;
17     d.year = 1999;
18     d.month = 4;
19     d.day = 23;
20     printf("Before function call:%d/%d/%d\n", d.year, d.month, d.day);
21     Func(&d);          /* 结构体变量的地址作函数实参，传地址调用 */
22     printf("After function call:%d/%d/%d\n", d.year, d.month, d.day);
23     return 0;
24 }

```

结构体指针 作函数参数

【例 12.5】

Before function call:1999/04/23

After function call:2000/05/22



指针作函数形参
实参必须为地址值

```

struct date
{
    int year;
    int month;
    int day;
};

struct date Func(struct date p)
{
    p.year = 2000;
    p.month = 5;
    p.day = 22;
    return p;
}

```

结构体变量 作函数返回值

【例12.6】

Before function call:1999/04/23
After function call:2000/05/22

```

15 int main()
16 {
17     struct date d;
18     d.year = 1999;
19     d.month = 4;
20     d.day = 23;
21     printf("Before function call:%d/%d/%d\n", d.year, d.month, d.day);
22     d = Func(d);           /* 函数返回值为结构体变量的值 */
23     printf("After function call:%d/%d/%d\n", d.year, d.month, d.day);
24     return 0;
25 }

```

12.5 向函数传递结构体

■ 向函数传递结构体的完整结构

- 复制整个结构体成员的内容，多个值
- 函数内对结构内容的修改不影响原结构
- 内容传递更直观，但开销大

■ 向函数传递结构体的首地址

- 用结构体数组/结构体指针作函数参数
- 仅复制结构体的首地址，一个值
- 修改结构体指针所指向的结构体的内容
- 指针传递效率高

12.5 向函数传递结构体

【例12.7】修改例12.3程序，用结构体数组作函数参数编程并输出计算学生的平均分

```
17 void InputScore(STUDENT stu[], int n, int m);
18 void AverScore(STUDENT stu[], float aver[], int n, int m);
19 void PrintScore(STUDENT stu[], float aver[], int n, int m);
20 int main()
21 {
22     float aver[N];
23     STUDENT stu[N];
24     int n;
25     printf("How many student?");
26     scanf("%d", &n);
27     InputScore(stu, n, 4);
28     AverScore(stu, aver, n, 4);
29     PrintScore(stu, aver, n, 4);
30     return 0;
31 }
```

12.5 向函数传递结构体

【例12.7】修改例12.3程序，用结构体数组作函数参数编程并输出计算学生的平均分

```
32  /* 输入n个学生的学号、姓名、性别、出生日期以及m门课程的成绩到结构体数组stu中 */
33  void InputScore(STUDENT stu[], int n, int m)
34  {
35      int i, j;
36      for (i=0; i<n; i++)
37      {
38          printf("Input record %d:\n", i+1);
39          scanf("%d", &stu[i].studentID);
40          scanf("%s", stu[i].studentName);
41          scanf(" %c", &stu[i].studentSex); /* %c前有一个空格 */
42          scanf("%d", &stu[i].birthday.year);
43          scanf("%d", &stu[i].birthday.month);
44          scanf("%d", &stu[i].birthday.day);
45          for (j=0; j<m; j++)
46          {
47              scanf("%d", &stu[i].score[j]);
48          }
49      }
50  }
```

12.5 向函数传递结构体

【例12.7】修改例12.3程序，用结构体数组作函数参数编程并输出计算学生的平均分

```
51  /* 计算n个学生的m门课程的平均分，存入数组aver中 */
52  void AverScore(STUDENT stu[], float aver[], int n, int m)
53  {
54      int i, j, sum[N];
55      for (i=0; i<n; i++)
56      {
57          sum[i] = 0;
58          for (j=0; j<m; j++)
59          {
60              sum[i] = sum[i] + stu[i].score[j];
61          }
62          aver[i] = (float)sum[i]/m;
63      }
64 }
```

12.5 向函数传递结构体

【例12.7】修改例12.3程序，用结构体数组作函数参数编程并输出计算学生的平均分

```
65  /* 输出n个学生的学号、姓名、性别、出生日期以及m门课程的成绩 */
66  void PrintScore(STUDENT stu[], float aver[], int n, int m)
67  {
68      int i, j;
69      printf("Results:\n");
70      for (i=0; i<n; i++)
71      {
72          printf("%10d%8s%3c%6d/%02d/%02d",
73                 stu[i].studentID,
74                 stu[i].studentName,
75                 stu[i].studentSex,
76                 stu[i].birthday.year,
77                 stu[i].birthday.month,
78                 stu[i].birthday.day);
79          for (j=0; j<m; j++)
80          {
81              printf("%4d", stu[i].score[j]);
82          }
83          printf("%6.1f\n", aver[i]);
84      }
}
```

用户自定义的数据类型

■ 结构体（Struct）

- 把关系紧密且逻辑相关的多种不同类型的变量，组织到统一的名字之下
- 占用相邻的一段内存单元

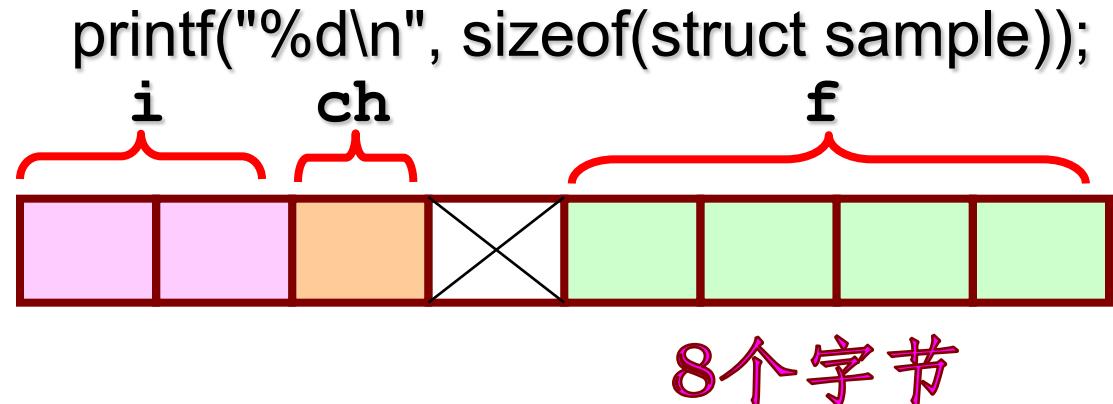
■ 共用体，也称联合（Union）

- 把情形互斥但逻辑相关的多种不同类型的变量，组织到统一的名字之下
- 占用同一段内存单元，每一时刻只有一个数据起作用

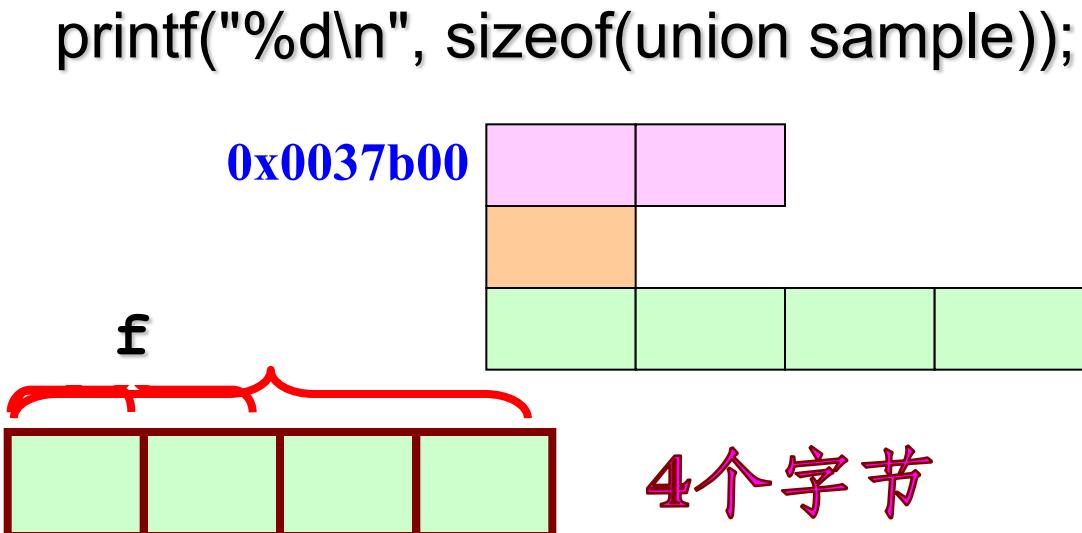
12.6 共用体

【例12.8】

```
struct sample
{
    short i;
    char ch;
    float f;
};
```



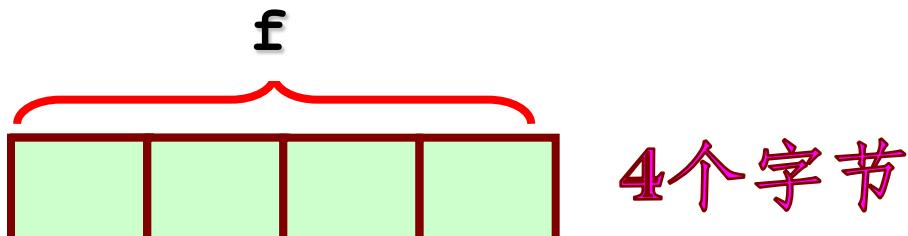
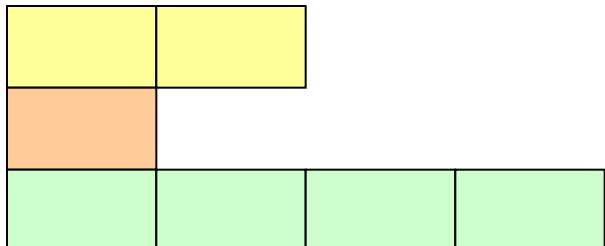
```
union sample
{
    short i;
    char ch;
    float f;
};
```



12.6 共用体

- `sizeof(union number)` 取决于占空间最多的那个成员变量
- 同一内存单元在每一瞬时只能存放其中一种类型的成员
- 起作用的成员是最后一次存放的成员，不能作为函数参数
- 不能进行比较操作，只能对第一个成员初始化

0x0037b00



12.6 共用体

姓名	性别	年龄	婚姻状况						婚姻状况 标记	
			未婚	已婚			离婚			
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量		

```
24 struct person           /* 定义职工个人信息结构体类型 */
25 {
26     char name[20];        /* 姓名 */
27     char sex;             /* 性别 */
28     int age;              /* 年龄 */
29     union maritalState marital; /* 婚姻状况 */
30     int marryFlag;        /* 婚姻状况标记 */
31 }
```

```
18 union maritalState      /* 定义婚姻状况共用体类型 */
19 {
20     int single;           /* 未婚 */
21     struct marriedState married; /* 已婚 */
22     struct divorceState divorce; /* 离婚 */
23 }
```

12.6 共用体

姓名	性别	年龄	婚姻状况						婚姻状况 标记	
			未婚	已婚			离婚			
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量		

```
7   struct marriedState           /* 定义已婚结构体类型 */  
8   {  
9       struct date marryDay;      /* 结婚日期 */  
10      char spouseName[20];        /* 配偶姓名 */  
11      int child;                /* 子女数量 */  
12  };  
13  struct divorceState          /* 定义离婚结构体类型 */  
14  {  
15      struct date divorceDay;    /* 离婚日期 */  
16      int child;                /* 子女数量 */  
17  };  
18  union maritalState           /* 定义婚姻状况共用体类型 */  
19  {  
20      int single;               /* 未婚 */  
21      struct marriedState married; /* 已婚 */  
22      struct divorceState divorce; /* 离婚 */  
23  };  
1   struct date  
2   {  
3       int year;  
4       int month;  
5       int day;  
6   };
```

12.7.1 枚举数据类型

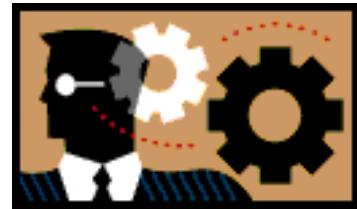
■ 枚举 (*Enumeration*) 数据类型

- 描述的是一组整型值的集合
- 用于当某些量仅由有限个数据值组成时

```
enum weeks {SUN, MON, TUE, WED, THU, FRI, SAT};  
enum weeks today;  
enum response {no, yes, none};  
enum response answer;  
  
today = TUE;  
answer = yes;  
  
enum response {no = -1, yes = 1, none = 0};
```



问题的提出



- 下面的结构是什么意思？

```
struct temp
{
    int          data;
    struct temp pt;
};
```

结构体声明时不能包含本结构体类型成员，系统将无法为这样的结构体类型分配内存

- CB下的错误提示：

- field 'pt' has incomplete type

- VC下的错误提示：

- 'pt' uses undefined struct 'temp'

- 下面的结构是什么意思呢？

```
struct temp
{
    int          data;
    struct temp *pt;
};
```

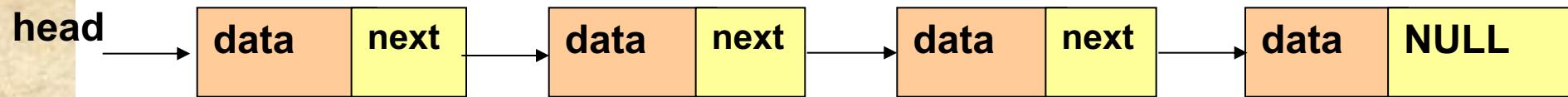
可包含指向本结构体类型的指针变量

12.7.2 动态数据结构——单向链表

■ 链表 (*Linked Table*) : 线性表的链式存储结构

- 特点: 用一组任意的存储单元存储线性表的数据; 存储单元可以是连续的, 也可是不连续的

```
struct Link  
{  
    int          data;  
    struct Link *next;  
};
```



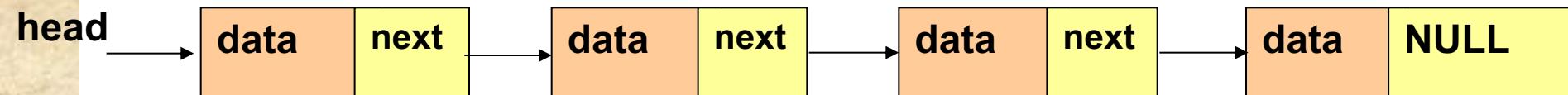
链表的定义

■ 链表 (*Linked table*) : 线性表的链式存储结构

- 为表示每个元素与后继元素的逻辑关系，除存储元素本身信息外，还要存储其直接后继信息

```
struct Link  
{  
    int data;  
    struct Link *next;  
};
```

两部分信息组
成一个节点



链表的定义

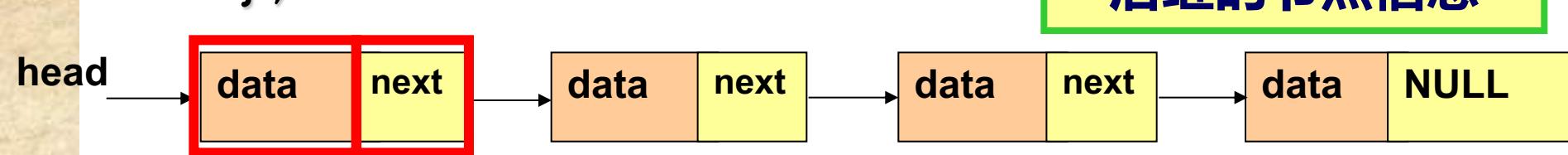
■ 链表 (*Linked Table*) : 线性表的链式存储结构

- 为表示每个元素与后继元素的逻辑关系，除存储元素本身信息外，还要存储其直接后继信息

```
struct Link  
{  
    int data;  
    struct Link *next;  
};
```

数据域：存储数据元素信息

指针域：存储直接后继的节点信息

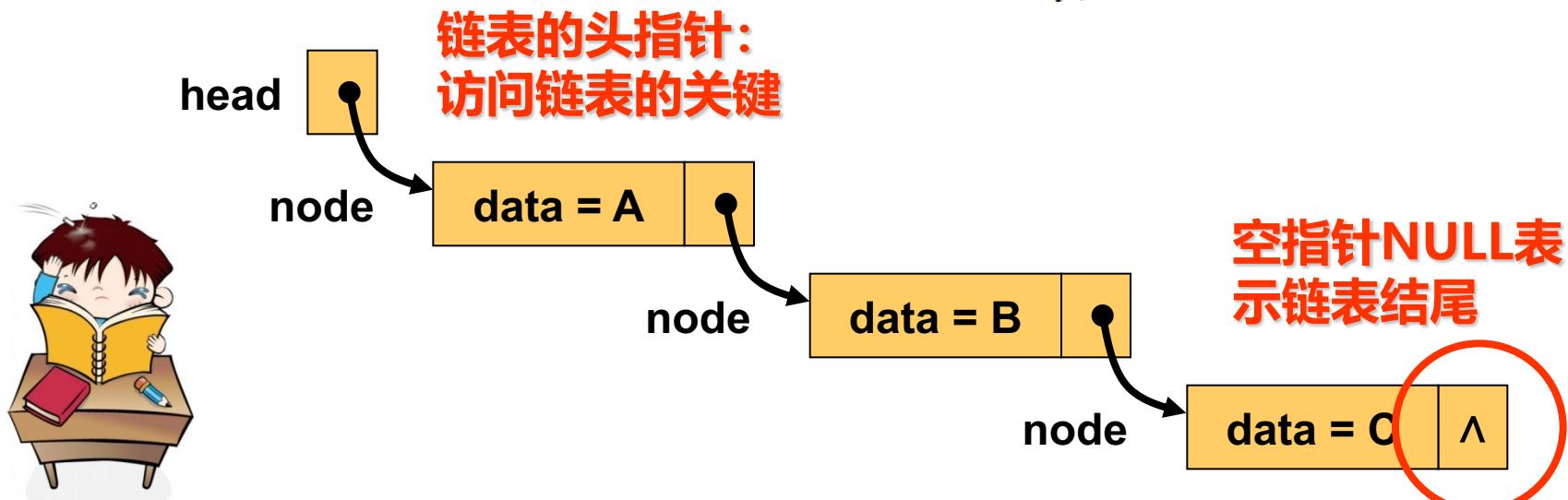


n个节点链接成一个链表 (因为只包含一个指针域，故又称线性链表或单向链表)

链表的建立

向链表中添加一个新节点

```
struct link  
{  
    int data;  
    struct link *next;  
};
```

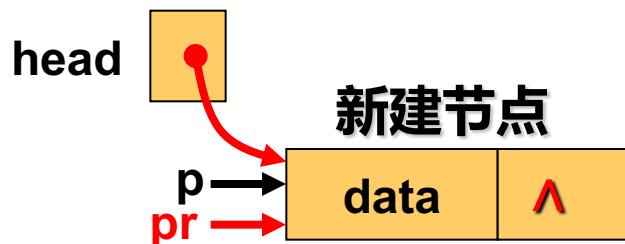


链表的建立

- 若原链表为空表 (`head == NULL`) , 则将新建节点 `p` 置为头节点

(1) `head = p`

```
struct link  
{  
    int data;  
    struct link *next;  
};
```



(2) `pr = p`

(3) `pr->next = NULL`

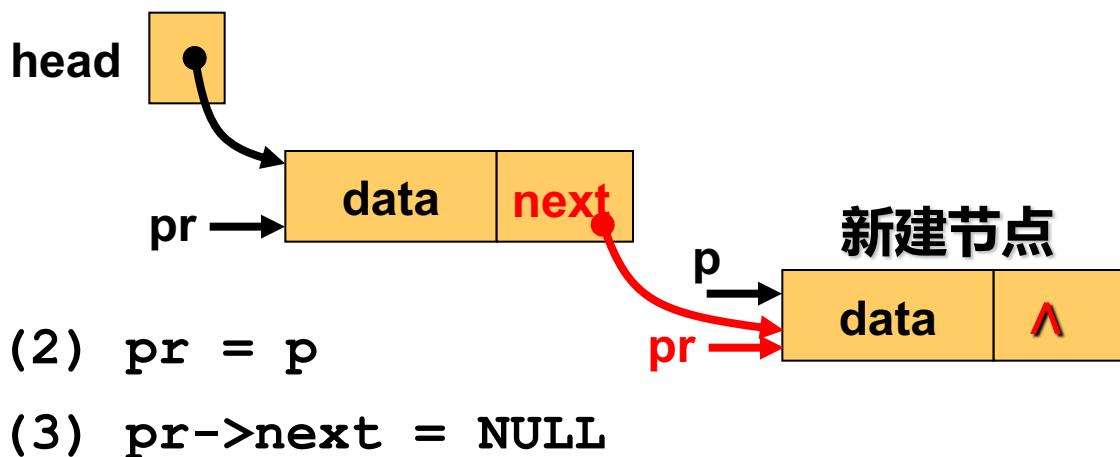


链表的建立

- 若原链表为非空，则将新建节点p添加到表尾

```
struct link  
{  
    int data;  
    struct link *next;  
};
```

(1) `pr->next = p`

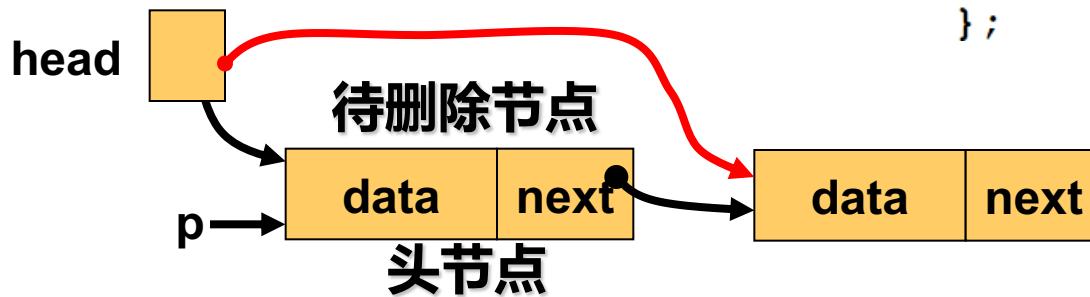


链表的删除操作

- 若原链表为空表，则退出程序
- 若待删除节点p是头节点，则将head指向当前节点的下一个节点即可删除当前节点

(1) `head = p->next`
(2) `free(p)`

```
struct link  
{  
    int data;  
    struct link *next;  
};
```



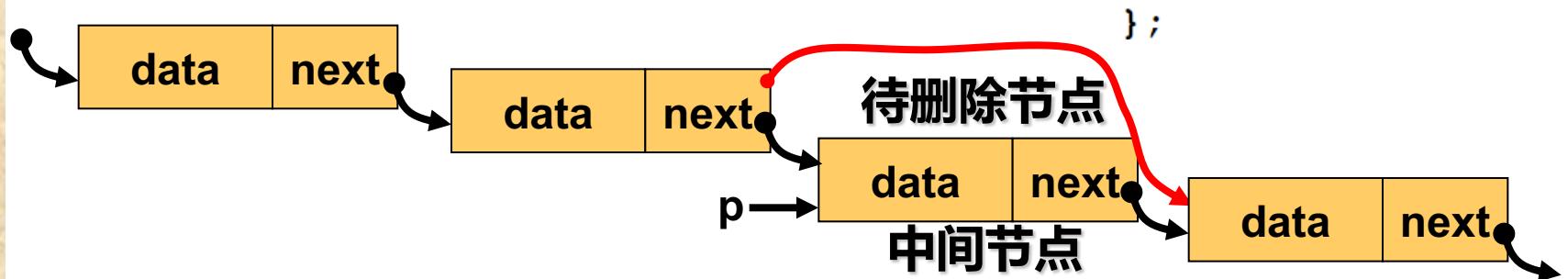
链表的删除操作

- 若待删除节点不是头节点，则将前一节点的指针域指向当前节点的下一节点即可删除当前节点

(1) `pr->next = p->next`

(2) `free(p)`

```
struct link  
{  
    int data;  
    struct link *next;  
};
```

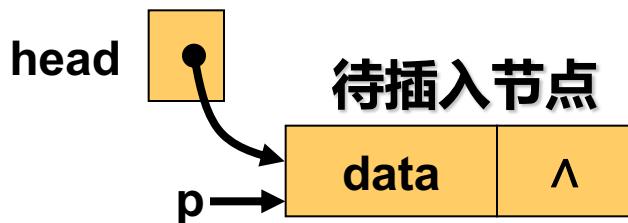


- 若已搜索到表尾 (`p->next == NULL`) 仍未找到待删除节点，则显示“未找到”

链表的插入操作

- 若原链表为空表，则将新节点p作为头节点，让head指向新节点p

(1) head = p

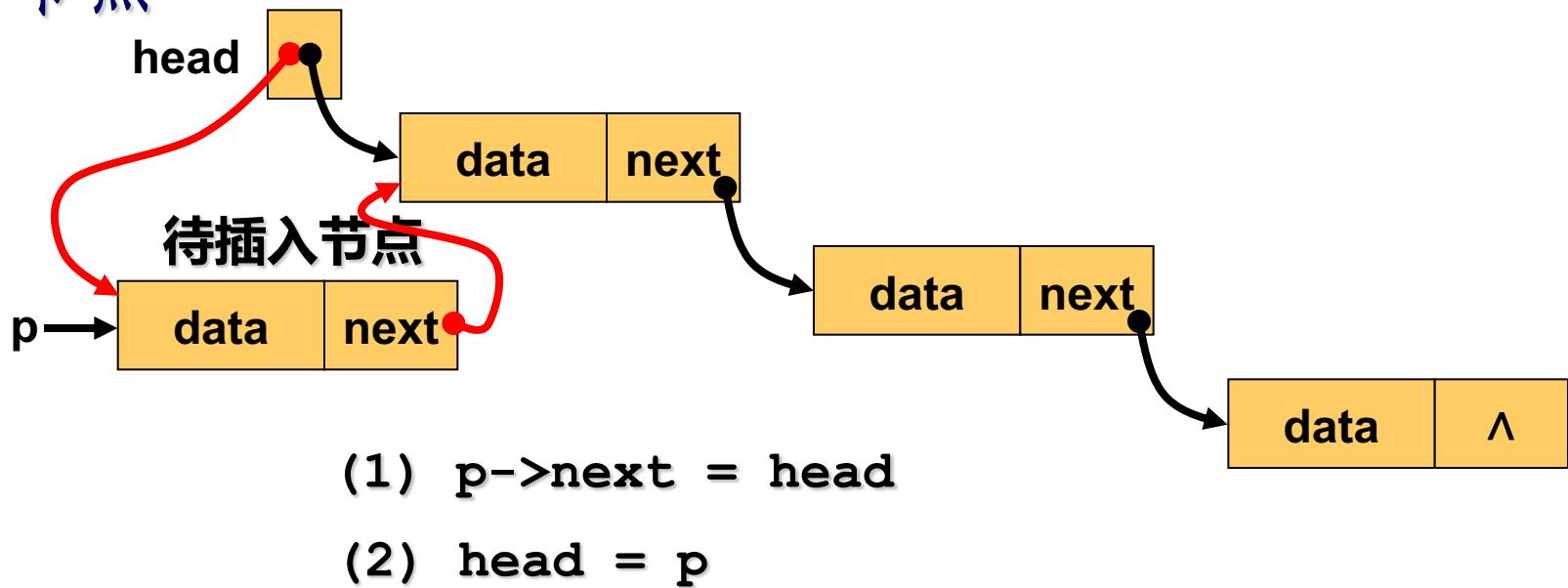


```
struct link
{
    int data;
    struct link *next;
};
```

```
p = (struct link *)malloc(sizeof(struct link));
p->next = NULL;
p->data = nodeData;
```

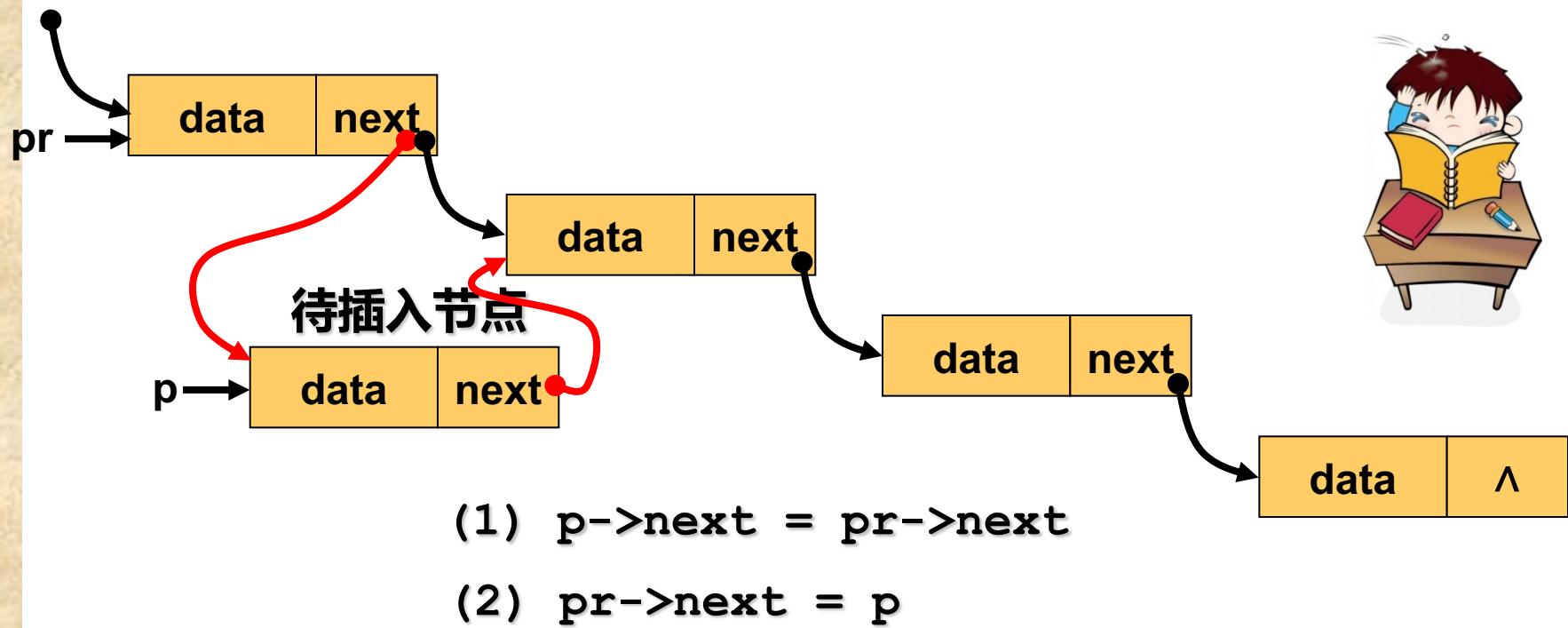
链表的插入操作

- 若原链表为非空，则按节点值（假设已按升序排序）的大小确定插入新节点的位置
- 若在头节点前插入新节点，则将新节点的指针域指向原链表的头节点，且让**head**指向新节点



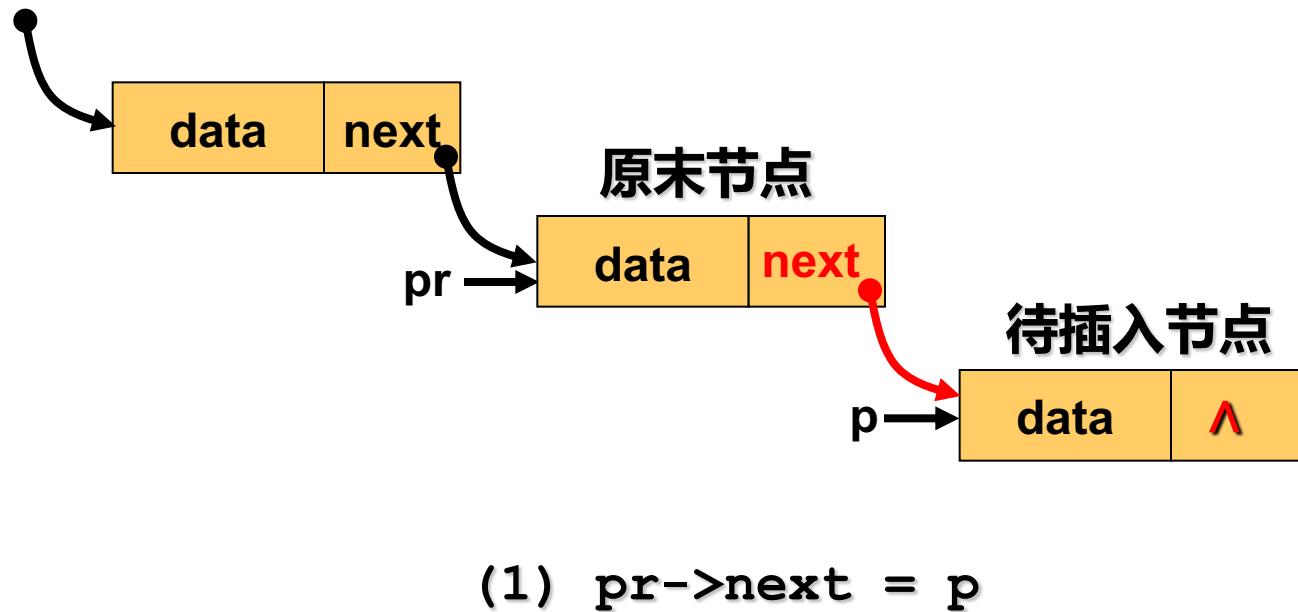
链表的插入操作

- 若在链表中间插入新节点，则将新节点的指针域指向下一节点且让前一节点的指针域指向新节点



链表的插入操作

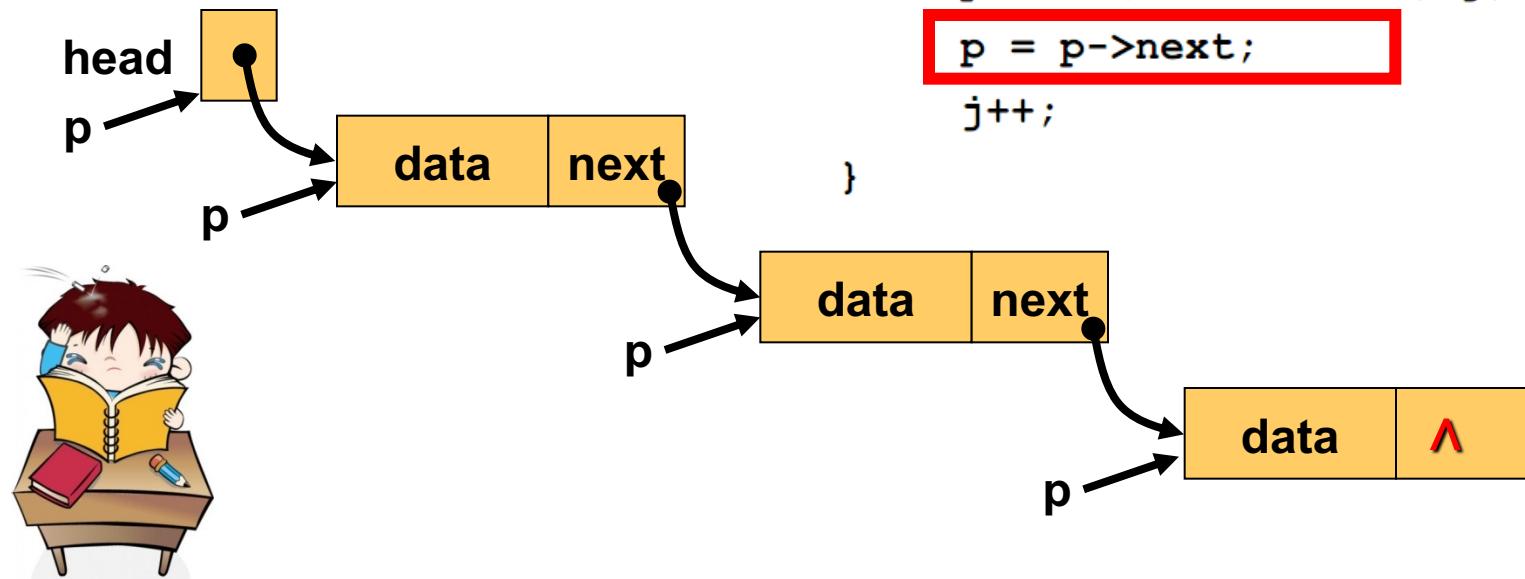
- 若在表尾插入新节点，则末节点指针域指向新节点



链表的输出

遍历链表的所有节点

```
struct link *p = head;  
int j = 1;  
while (p != NULL)  
{  
    printf("%5d%10d\n", j, p->data);  
    p = p->next;  
    j++;  
}
```





■ Questions and answers

