

C程序设计语言

第6章 结构

孙志岗
sun@hit.edu.cn
<http://sunner.cn>

从基本数据类型、复合数据类型到抽象数据类型

- 类型本不存在
 - 内存里存储的内容，你认为它是什么，它就是什么
- 高级语言设计了基本数据类型：整型、浮点型、字符型等。不同的语言也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
- 复合数据类型是基本数据类型迭代派生而来
 - 典型的代表就是“结构”，数组、指针也可算作此类
- 抽象数据类型(ADT)在复合数据类型的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

2004-12-19

Structures

2

一个问题

- 在程序里表示一个人（姓名、年龄、性别、身高、体重……），怎么表示？
 - `char name[12];`
`unsigned int age;`
`char sex;`
- 想表示多个人呢？
 - 定义多个数组？（有些搞笑了）



2004-12-19

Structures

3

C语言的解决办法

- `struct person`
 - {
`char name[12];`
`unsigned int age;`
`char sex;`
};
 - `struct person`是一个类型
- `struct person students[4];`
- `students[0].name`
`students[0].age`
`students[0].sex`
 - 它们都是变量，一般称为结构的成员变量



2004-12-19

Structures

4

结构 (Structure) 的内存占用

- 一个结构变量的成员变量在内存中是相邻的
- 整个结构变量的将占用多少内存呢?
 - 是所有成员变量的内存总和吗?
 - 我们应该用 `sizeof` 获得结构的大小
 - 事实上, 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、不同的平台、不同的编译参数, 对齐方式会有变化。

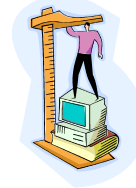
2004-12-19

Structures

5

`sizeof` 到底是什么?

- 它是一个C语言的关键字, 并不是函数
- 可以用两种形式使用
 - `sizeof(表达式)`
 - 一般都使用 `sizeof(变量名)`
 - `sizeof(类型)`
- 求出的结果为表达式值所属类型或者类型占用的字节数



2004-12-19

Structures

6

`struct` 类型的特点

- 一个普通的类型
 - 所以可以定义该类型的变量、数组、指针.....
 - 它的成员可以是任意类型
 - 基本类型、数组、指针、结构.....
 - 可以做函数的参数类型和返回值类型
- `struct` 类型的变量
 - 可以互相赋值
 - 可以 `&`
 - 不可能参与运算
- 它的成员个个也都是如假包换的变量
- 面向对象和数据库是 `struct` 的思想的发展

2004-12-19

Structures

7

结构指针

- `struct point`

```
{
    int x;
    int y;
};
```
- `struct point pt;`
`struct point* ppt;`
- `ppt = &pt;`
- 怎样通过 `ppt` 访问 `pt` 的成员?
 - `(*ppt).x = 0;`
 - `ppt->x = 0;` /* 更常用 */

2004-12-19

Structures

8

思考题

- `struct point`
{
 `int x;`
 `int y;`
};
`struct rect`
{
 `struct point pt1;`
 `struct point pt2;`
};
- `struct rect rt, *rp=&rt;`
- 下面表达式哪些合法?
 - `rt.pt1.x`
 - `(*rp).pt1.x`
 - `rp->pt1.x`
 - `rt->pt1.x` ✗
- 上面合法的表达式都是等价的吗?



2004-12-19

Structures

9

再思考

- 下面的结构什么意思?
- `struct something`
{
 `struct something obj1;`
 `struct something obj2;`
};
- 下面的的呢?
- `struct something`
{
 `char name[10];`
 `struct something* pOtherObj;`
};
- 第一个结构无限循环了，第二个没问题



2004-12-19

Structures

10

定义自己的类型名

- `struct person student; /* It works */`
- `person student; /* Can this work? */`
- `typedef struct`
{
 ;
 ;
} `person;`

`person student; /* It works! */`

2004-12-19

Structures

11

结构做参数和返回值

- 结构变量类型是一个结构，其值不好描述：
`typedef struct{int x; int y;} point;`
`point mid, begin, end = {0, 0};`
`begin = end;`
- 所以对代码：
`point MidPt(point pt1, point pt2)`
{
 `point pt;`
 `pt.x = (pt1.x+pt2.x)/2;`
 `pt.y = (pt1.y+pt2.y)/2;`
 `return pt;`
}
`mid = MidPt(begin, end);`
- `MidPt`中`begin`和`end`会得到什么？`mid`得到什么？



2004-12-19

Structures

12

结构做参数和返回值

- 这样呢？

```
void MidPt(const point* pPt1,
          const point* pPt2,
          point* pMid)
{
    pMid->x = (pPt1->x + pPt2->x)/2;
    pMid->y = (pPt1->y + pPt2->y)/2;
}
```

```
MidPt(&begin, &end, &mid);
```

- 指针传递效率更高，内容传递更直观

2004-12-19

Structures

13

位字段

- 想表达人的姓名、性别、肤色、出生年、月、日，都定义什么类型的成员变量？

```
struct person
{
    char name[12];
    char sex;
    char color;
    int year;
    char month;
    char day;
};
```

- 这样有很多的空间浪费，比如month只可能取值1-12

2004-12-19

Structures

14

位字段

- struct person

```
{
    char name[12];
    unsigned int sex : 2;
    unsigned int color : 2;
    int year;
    unsigned int month : 4;
    unsigned int day : 5;
};
```

成员所占位数

- 调整成员顺序可以让结构更紧凑，占用内存更少
- 每个位段都可以当作一个无符号整型数使用
 - 表达范围当然受限，而且当然不能取地址

2004-12-19

Structures

15

结构的一个高级用法

- 记录成绩，但成绩个数不定（选修课）

- struct person

```
{
    char studentNo[12];
    char name[12];
    int numberOfScores;
    float score[];
};
```

int好还是char[]好？

怎么是空的？

- 最后一个数组的定义可以不指定元素个数，此时它只是一个符号，不占用内存

2004-12-19

Structures

16

结构的一个高级用法

```
struct person* pStudent;
int numberOfScores, size, i;

scanf("%d", &numberOfScores);

size = sizeof(struct person)
      + numberOfScores * sizeof(float);
pStudent = (struct person*)malloc(size);

pStudent->numberOfScores=numberOfScores;
for (i=0; i<numberOfScores; i++)
{
    printf("%d:", i+1);
    scanf("%f", &(pStudent->score[i]));
}
```

struct.c

2004-12-19

Structures

17

联合 (Union)

- `union u_tag`

```
{
    int ival;
    float fval;
    char* sval;
} u;
```
- 使用上和`struct`一样
- 特点:
 - `u.ival`、`u.fval`、`u.sval`的地址相同
- `sizeof(union xxx)`取决于占空间最多的那个成员变量

2004-12-19

Structures

18

C语言的核心学习到此结束

- 教材P168列出的32个关键字和围绕它们的语法、符号构成了C语言的核心
 - 26个字母以及围绕它们的构词法、语法构成了英语的核心
- 本课程对C语言核心的讲述并非面面俱到
 - 一些比较“偏”，很少用到的没有涉及
- 学会原理，掌握思维
 - 复杂的表面都是简单的原理的外在表现
- 仅掌握语言的核心当然不能熟练运用语言
 - 背下英语的所有单词和语法，你就能写出莎士比亚一样的诗句了吗？

2004-12-19

Structures

19

C语言中的三大定律

- 表达式定律
 - 任何能产生数值结果的运算、操作都可以作为表达式，并可以放到任何需要数值结果的地方，只要数值类型能够匹配
 - 常见的可以产生数值结果的运算和操作
 - 算术、逻辑、位运算等
 - `?`、`:`、`&`、`*`等
 - 有返回值的函数
 - 赋值
 - 常见的需要数值的地方有：
 - 赋值
 - 条件判断
 - 函数调用

2004-12-19

Structures

20

C语言中的三大定律

■ 类型定律

- 任何类型都可以在任何需要类型的地方使用；用任何类型定义的变量都要占用内存
- 已知特例
 - 函数返回值不能定义为数组类型（只能是指针的形式）
 - 函数参数定义为数组类型，此时该参数不占用内存
- 常用类型
 - 基本数据类型、指针、数组、结构.....
- 常见的需要类型的地方
 - 定义变量
 - 定义指针、数组和结构
 - 函数参数和返回值
 - `sizeof`

2004-12-19

Structures

21

C语言中的三大定律

■ 参数传递定律

- 函数调用时的参数传递永远都是传值调用，把实参的值拷贝给形参
 - 实参：调用者提供的参数
 - 形参：函数定义的参数
 - 基本数据类型无容置疑
 - `struct`也无容置疑
 - 指针作为参数时，把指针变量的内容（就是其指向的内存地址）做了拷贝
 - 数组名作为参数时，把它等同于指针看待

2004-12-19

Structures

22