School of Economics ang Management, Beihang University

Introduction to C Programming

Jichang Zhao

jichang@buaa.edu.cn

Structures

Objectives

- **Single Structures**
- **Arrays of Structures**
- **Passing and Returning Structures**
- **Unions**
- Common Programming and Compiler Errors
  - 阅读参考教材P465
- 枚举类型
  - 阅读参考教材P503

Introduction

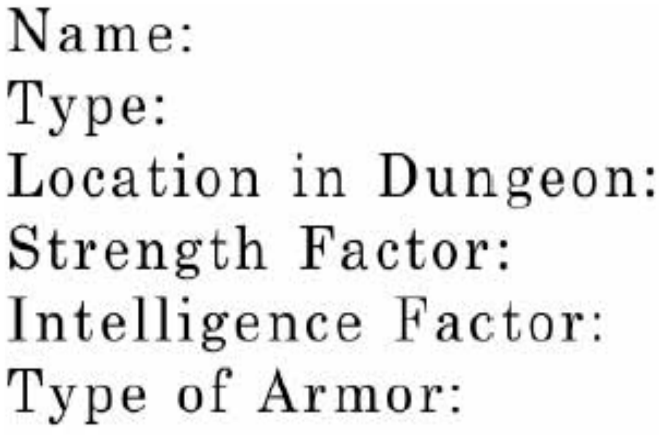

Name:
Type:
Location in Dungeon:
Strength Factor:
Intelligence Factor:
Type of Armor:

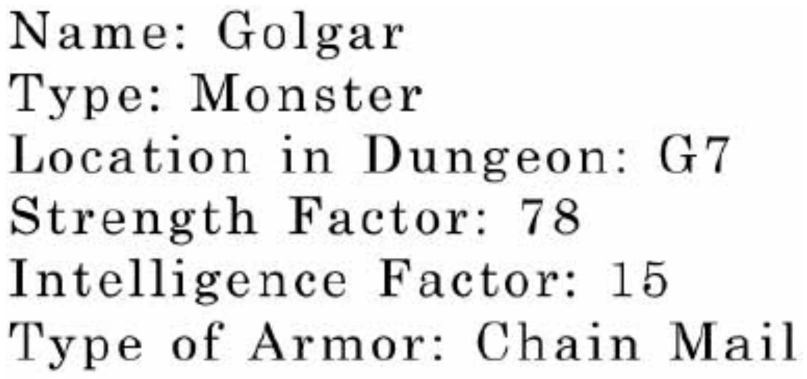
Figure 12.1 Typical components of a video game character

Introduction

- Each data item is an entity by itself, called a **data field**
- Together, all the data fields form a single unit called a **record**
  - In C, a record is referred to as a **structure**
  - Advantage of structures is when the same structure type is used in a list many times
    - An **inferior alternative** to structures (especially arrays of structures) is parallel arrays
    - **Parallel arrays** are two or more arrays, where each array has the same number of elements and the elements in each array are directly related by their position in the arrays
    - They are rarely used

Introduction

- A structure’ s **form** consists of the symbolic names, data types, and arrangement of individual data fields in the record
- The structure’ s **contents** consist of the actual data stored in the symbolic names



```
Name: Golgar
Type: Monster
Location in Dungeon: G7
Strength Factor: 78
Intelligence Factor: 15
Type of Armor: Chain Mail
```

Figure 12.2 The form and contents of a structure

Single Structures

- Structure definition in C:

```
struct
{
   int month;
   int day;
   int year;
} birth;（补充：是一个结构变量）
```

  - Reserves storage for the individual data items listed in the structure
  - The three data items are the **members of the structure**
- Assigning actual data values to the data items of a structure is called **populating（填充） the structure**

Single Structures

- Multiple variables can be defined in one statement

```
struct {int month; int day; int year;} birth, current;
```

- Common to list the form of the structure with no following variable names
  - The list of structure members must be preceded by a user-selected **structure type name** （补充：也叫结构标记，常用风格）

```
struct Date
{
  int month;
  int day;
  int year;
};//分号很关键
typedef struct Date DATE;
DATE为新的类型名称，等价于struct Date
也可以合并写成
typedef struct {int…} DATE;
//注意链表里不可以，结构成员有指向同类型结构的指针
```

Single Structures

- Initialization of structures follows the same rules as for the initialization of arrays:
  - `struct Date birth = {12, 28, 1987};`
- Structure members can be of any data type

```
struct PayRecord
{
  char name[20];
  int idNum;
  double regRate;
  double otRate;
};
struct PayRecord employee = {"H. Price", 12387,
  15.89, 25.50};
```

Single Structures

- Individual members can also be arrays and structures

```
struct
{
  char name[20];
  struct Date birth;
} person;
```

- Access structure members
  - .
  - Example: `person.name[4]`

补充：结构变量

- 结构的成员在内存中按声明的顺序存储
  - 结构所占的空间是否是所有成员所占的空间之和？
  - 不一定，可能存在内存“对齐”（偏移量是特定字节数的倍数）
    - #pragma pack(show) 可通过warning形式揭示编译器的对齐倍数
  - 结构成员的地址之间或结构末尾可能产生“空洞”
  - 结构开始处没有“空洞”，以保证指向第一个成员的指针就是指向结构的指针
  - DEMO: smem.c
- 每个结构代表一种新的作用域
  - 任何声明在此作用域中内的变量名字都不会和程序中的其他名字冲突
  - 为其成员设置了独立的名字空间(name space)

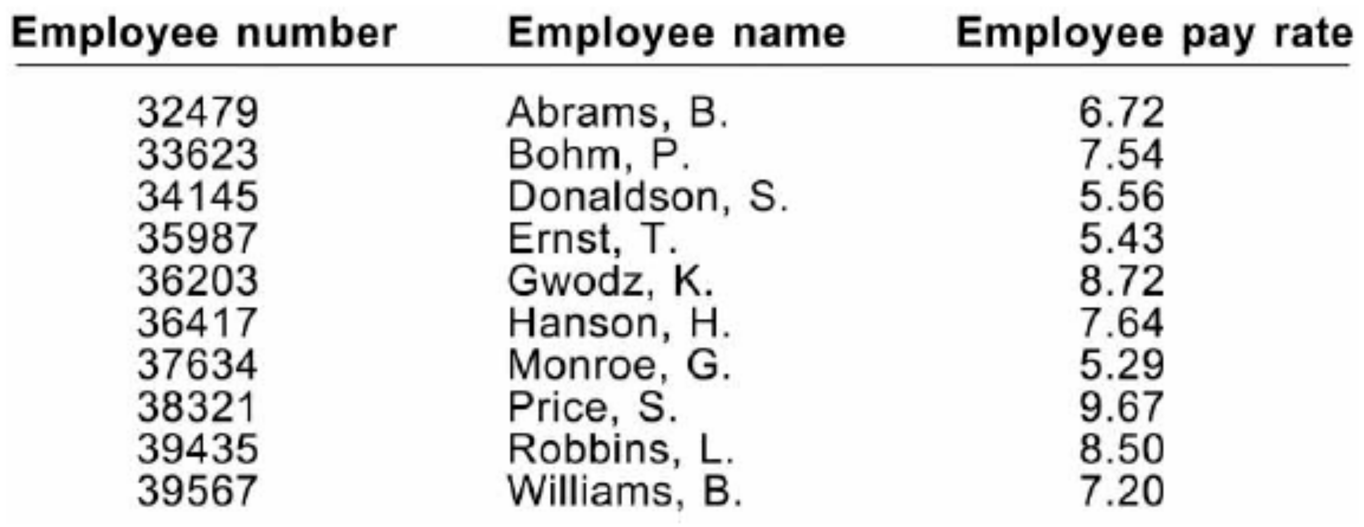
补充：结构的初始化

- 用于结构初始化的表达式必须是常量（C99中不再限制）
  - 不能用变量来初始化结构成员
  - 初始化式中的成员数可以少于它所初始化的结构，剩余成员用0作初值
    - 也可以多，编译器会产生warning
- 指定初始化（c99)
  - {.number=0,.name=“test”}
  - .+成员名称为指示符
  - 顺序不必一致
  - 未涉及成员均初始化为0
  - **DEMO: si.c**
- 复合字面量
  - `(struct student){10,”test”,10.0}`

补充：结构的复制

- 结构可以复制
  - `struct Student s1, s2;`
  - `s1={…};`
  - `s2=s1;`
  - 依次复制所有成员
- 产生的“奇怪”应用
  - 实现数组复制
  - `struct{int a[10];} a1, a2;`
  - `a1=a2;`
- 不支持==,!=,>,<,>=,<=等运算符
- **DEMO:sc.c**

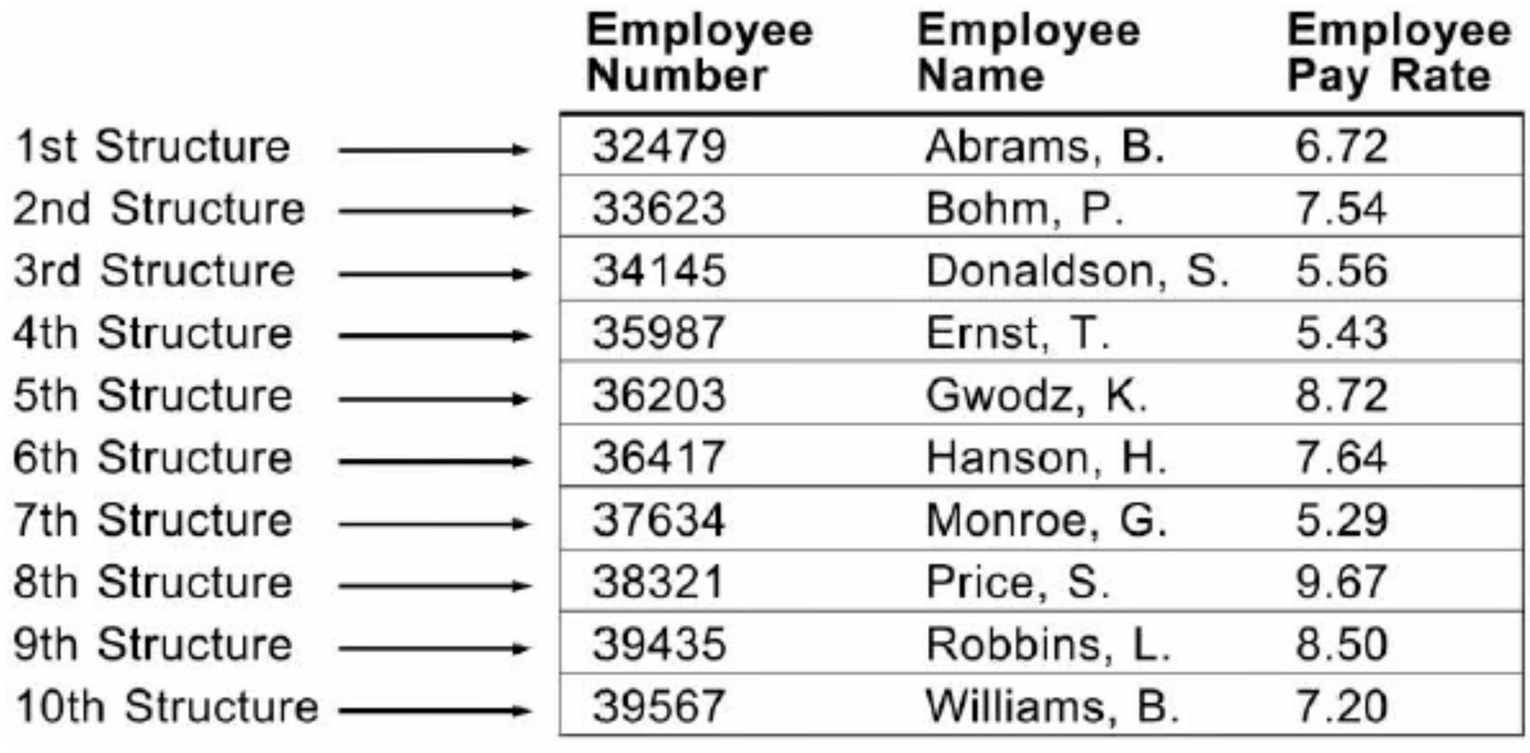
Arrays of Structures



Employee number	Employee name	Employee pay rate
32479	Abrams, B.	6.72
33623	Bohm, P.	7.54
34145	Donaldson, S.	5.56
35987	Ernst, T.	5.43
36203	Gwodz, K.	8.72
36417	Hanson, H.	7.64
37634	Monroe, G.	5.29
38321	Price, S.	9.67
39435	Robbins, L.	8.50
39567	Williams, B.	7.20

Figure 12.3 A list of employee data

Arrays of Structures



	Employee Number	Employee Name	Employee Pay Rate
1st Structure →	32479	Abrams, B.	6.72
2nd Structure →	33623	Bohm, P.	7.54
3rd Structure →	34145	Donaldson, S.	5.56
4th Structure →	35987	Ernst, T.	5.43
5th Structure →	36203	Gwodz, K.	8.72
6th Structure →	36417	Hanson, H.	7.64
7th Structure →	37634	Monroe, G.	5.29
8th Structure →	38321	Price, S.	9.67
9th Structure →	39435	Robbins, L.	8.50
10th Structure →	39567	Williams, B.	7.20

Figure 12.4 A list of records

Arrays of Structures (continued)

- Without explicit initializers, the numeric elements of both static and external arrays or structures are initialized to 0 (or nulls)
- `struct Employee`
- `{`
  - `int idNum;`
  - `double payrate;`
  - `double hours;`
- `};`
- `typedef struct Employee EMPLOYEE;`
- `EMPLOYEE elist[100];`

Passing and Returning Structures

- Individual structure members may be passed to a function **in the same manner as any scalar variable**
  - `display(emp.idNum)`
  - `calcPay(emp.payRate,emp.hours);`
- On most compilers, **complete copies of all members of a structure** can also be passed to a function by including the name of the structure as an argument to the called function
  - `calcNet(emp);`
  - 补充：可能造成大的系统开销，可以通过指针来避免

Passing and Returning Structures

- A structure can be passed by address
  - `calcNet(&emp);`
  - `double calcNet(struct Employee *pt)`
  - `(*pt).idNum or pt->idNum`
  - **DEMO:stra.c**

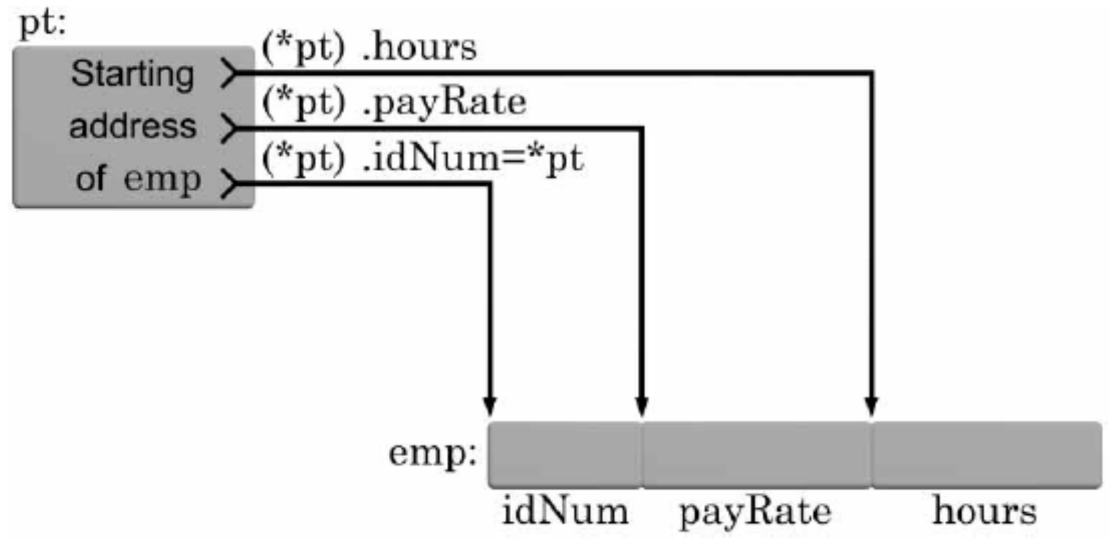


Figure 12.5 A pointer can be used to access structure members

Passing and Returning Structures

- ++ and -- can be applied to structures
  - 补充：->的优先级高于自增运算符
  - ++pt->hours：将hours加1
  - (pt++)->hours：获取hours，结构体指针指向下一个结构体
  - (++pt)->hours：指向下一个结构体，获取hours
  - **DEMO:strarray.c**

Unions

- A **union** is a data type that **reserves the same area in memory for two or more variables**

```
union
{
  char key;
  int num;
  double price;
} val;
```

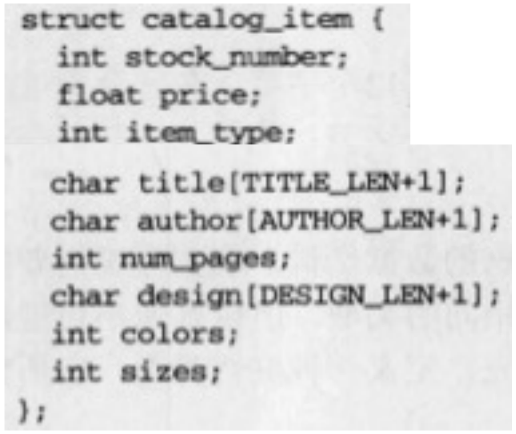
  - Each of these types, but only one at a time, can actually be assigned to the union variable
  - A union reserves sufficient memory locations to accommodate its largest member’s data type

补充：union

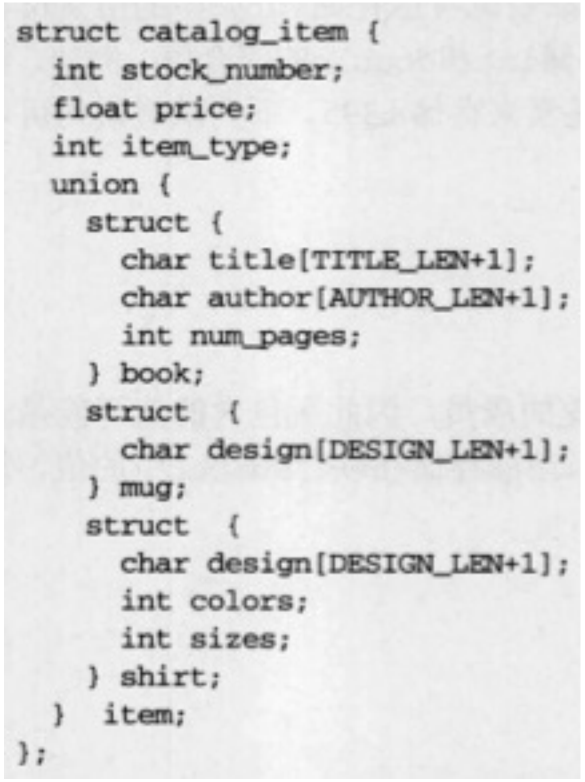
- union中可以定义多个成员，union的大小由最大成员的大小决定
- union成员共享同一块大小的内存，一次只能使用其中的一个成员。
- 对某一个成员赋值，会覆盖其他成员的值
  - 因为他们共享一块内存
  - 但前提是成员所占字节数相同
  - 当成员所占字节数不同时只会覆盖相应字节上的值

补充：使用联合的场景举例

- 三种商品：书籍，杯子和衬衫
  - 每种商品都包含库存量、价格及其他与商品相关的信息
    - 书籍：书名、作者、页数
    - 杯子：设计
    - 衬衫：设计、可选颜色、可选尺寸
  - 有没有更好的方法
    - 面向对象



```
struct catalog_item {
   int stock_number;
   float price;
   int item_type;
  char title[TITLE_LEN+1];
  char author[AUTHOR_LEN+1];
  int num_pages;
  char design[DESIGN_LEN+1];
  int colors;
  int sizes;
};
```



```
struct catalog_item {
  int stock_number;
  float 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;
};
```

Unions

- Individual union members are accessed using the same notation as structure members
- Typically, **a second variable keeps track of the current data type stored in the union**

```
switch(uType)
{
  case ‘c’: printf("%c", val.key);
            break;
  case ‘i’: printf("%d", val.num);
            break;
  case ‘d’: printf("%f", val.price);
            break;
  default : printf("Invalid type : %c", uType);
}
```

补充：将标记与联合封装

- typedef struct
- {
  - int kind;//标记，当前联合内的存储类型
  - union
  - {
    - int i;
    - double d;
  - }u;
- }Number;
- Number n;
- …
- if(n.kind==0)
  - printf(“%d”,n.u.i);

Unions

- A type name can be associated with a union to create templates

```
union DateTime
{
   long days;
   double time;
};
union DateTime first, second, *pt;
```

- Pointers to unions use the same notation as pointers to structures
- **Unions may be members of structures and arrays; structures, arrays, and pointers may be members of unions**

Unions

```
struct
{
  char uType;
  union
  {
    char *text;
    double rate;
  } uTax;
} flag;
```

- `rate` is referenced as `flag.uTax.rate`
- The first character of the string whose address is stored in the pointer `text` is accessed as `*flag.uTax.text`

补充：联合的初始化

- `union {`
- `int i;`
- `double d;`
- `} u={0};`
- 初始化表达式必须是常量
- 只有联合的第一个成员可以获得初始值
- 联合也可以复制

Homework

- 1. P452， 第2题
- 2. P456， 第4题
- 3. P462， 第2题
- 4. P462， 第4题
- 5. 《现代方法（2）》P293， 第5题。