西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

设计函数 - 接口与实现分离

1、对象初始化

2、声明与定义

4.7 对象初始化

- 在前面的描述中多次使用“变量赋初值”一说，实际上这样的说法不完整，特别是“赋初值”容易让人联想起“赋值”的概念。从现在开始，使用“初始化”这个词语来替代“赋初值”。

4.7 对象初始化

- 1．初始化概念
- 对象定义时指定了变量的类型和标识符，也可以为对象提供初始值，定义时指定了初始值的对象被称为是“已初始化的（initialized）”，而创建对象并给它初始值就称为初始化（initialization）。

- 初始化不是赋值，赋值的含义是擦除对象的当前值并用新值代替，而初始化除了给初值外，之前还创建了对象。

4.7 对象初始化

- 例如：

```
int a=10,b=5,c;
```

- 变量a和b初始化了。
- C语言的初值只能是常量值。

4.7 对象初始化

- 由于使用“=”来初始化变量，使人容易把初始化当成是赋值的一种形式。但是在C语言中初始化和赋值的确是两种不同的操作。

4.7 对象初始化

```
1 #include <stdio.h>
2 int m=100; //全局变量已初始化
3 int n; //全局变量未初始化
4 int main()
5 {
6   int a;
7   a=10+m+n; //a=10+100+0
8   return 0;
9 }
```

4.7 对象初始化

```
1 #include <stdio.h>
2 int m=100; //全局变量已初始化
3 int n; //全局变量未初始化
4 int main()
5 {
6   int a;
7   a=10+m+n; //a=10+100+0
8   return 0;
9 }
```

4.7 对象初始化

```
1 #include <stdio.h>
2 int m=100; //全局变量已初始化
3 int n; //全局变量未初始化
4 int main()
5 {
6   int a;
7   a=10+m+n; //a=10+100+0
8   return 0;
9 }
```

4.7 对象初始化

【例4.11】

静态局部变量和动态局部变量的比较。

4.7 对象初始化

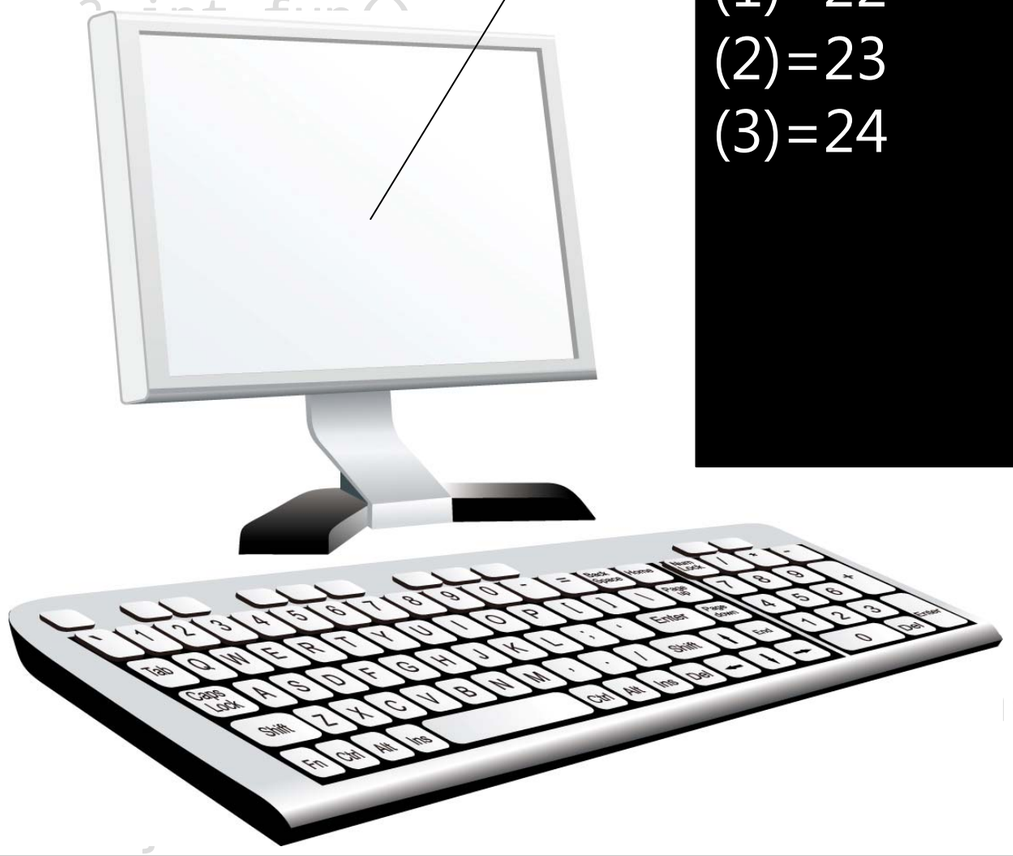
例4.11

```
1 #include <stdio.h>
2 int fun()
3 {
4   static int m=10; //进入函数前已初始化
5   int n=10;
6   m++, n++;
7   return m+n;
8 }
9 int main()
10 {
11   int i;
12   for (i=1;i<=3;i++)
13     printf("(%d)=%d\n",i,fun());
14   return 0;
15 }
```

4.7 对象初始化

例4.11

```
1 #include <stdio.h>
2 int fun()
```



程序运行屏幕

```
(1)=22
(2)=23
(3)=24
```

```
();
```

4.7 对象初始化

例4.11

```
1 #include <stdio.h>
2 int fun()
3 {
4    static int m=10; //进入函数前已初始化
5    int n=10;
6    m++, n++;
7    return m+n;
8 }
```

4.7 对象初始化

- 2．对象初始化规则
- 可以看出对象什么时候初始化，会有什么样的初值是由它的存储方式来决定。

4.7 对象初始化

表4-3 对象初始化规则

对象	初值	初始化时机
全局对象且初始化	设定值	程序运行前已完成
全局对象未初始化	0填充	程序运行前已完成
静态局部对象且初始化	设定值	程序运行前已完成
静态局部对象未初始化	0填充	程序运行前已完成
局部对象且初始化	设定值	每次进入函数时
局部对象未初始化	随机值	

“0填充”是指将对象内存单元所有二进制位设为0，对于数值型来说就是值为0。

4.8 声明与定义

- 在前面的描述中经常性地使用了两个词语：声明（declarations）和定义（definitions），例如：函数定义和函数声明，变量定义和extern声明等，后面出现得更频繁，因此有必要在这里讨论两个词语在描述C语言语法时的确切含义。

4.8 声明与定义

- 1．基本概念
- 定义用于创建对象并为对象分配实际的存储空间，而且还将一个名字与此对应；在一个程序中，对象有且仅有一个定义。如果定义多次，编译器会提示重复定义错误。

- 声明用于向程序表明对象的类型和名字，它有两重含义：一是表明名字已在别的地方有定义，二是表明名字已用，别的地方不能再使用这个名字。

4.8 声明与定义

- 作个拟人化的比喻：
- “定义”告诉编译器想要在存储空间有个位子，于是编译器为它安排了一个位子，且用名字与它对应。如果再次定义，那么编译器会想这个名字不是已经分配了位子吗？再要就是错误的。

4.8 声明与定义

- “声明”告诉编译器信息：一个名字已经有了一个位子了，是什么什么样的；当编译器遇到这个名字时，它会依据得到的信息去检查使用是否得当，如果不对就会指出用错了。
- “声明”可以重复地告诉编译器信息，只要确保不是不同的信息而让编译器不知如何是好即可。如果编译器从未得到信息，那么它会毫不犹豫地指出使用是错的。“定义”在想要位子且已经得到的时候，实际上也捎带向编译器告诉了信息，换言之，定义同时也会产生声明。

- 2．函数声明和函数定义的区别
- 函数定义包含函数体，即包含实现函数功能的代码，显然这样的代码不能出现两次，否则编译器会出现二义性：究竟用哪一段代码。所以函数重复定义就会编译错误。

4.8 声明与定义

- 因为定义也包含声明，所以函数定义后可以在往下的作用域调用这个函数，那么在函数定义前呢？由于编译器是从文件开始往下编译的，让它没有函数信息就调用函数时，它就会给出“未定义”错误，所以需要在调用前面给出函数声明，方法是使用函数原型。但必须确保函数原型与函数定义是一致的，否则同样是二义性错误。

4.8 声明与定义

- 由于编译器对每个文件编译是独立的，所以在一个文件得到的声明信息不会传到另一个文件中，换言之，需要在另一个文件重新给出声明信息。如果是在与函数定义不同的源文件出现了函数调用，那么就要在那个文件中再次出现函数原型。

- 3．对象声明和对象定义的区别
- 对象定义时就创建了与对象类型对应的存储空间，还可以初始化对象，初始化只能出现在创建对象存储空间的时候。
- 可以根据前面的作用域规则来使用对象，值得注意的是，对象是不能用在定义时所处作用域的上一级区域的。如函数中定义的变量不能在全局作用域使用，复合语句中定义的变量不能在函数中使用等。

4.8 声明与定义

- 对象定义后，由于定义包含声明，故可以在其后的作用域内使用。如果想要在定义的前面使用对象，就应该在前面给出对象声明。方法是使用extern，例如：

```
1 extern int a;
2    ⋮
3 int a=10;
```

4.8 声明与定义

- 如果把第1行写成：

```
1 extern int a=10;
```

- 则是错误的，因为extern是声明，它仅向编译器提供“有一个整型变量a”这样的信息，它没有实际产生a的存储单元，因而也不能初始化。

4.8 声明与定义

- 当extern声明出现在全局作用域上，extern声明允许出现初始化，例如：

```
1 extern int a=100; //出现在全局作用域是允许的
2     ┆
3 int a; //错误，重复定义
4     ┆
5 extern int a=100; //错误，重复定义
```

- 但此时不能再定义对象，例如第3行，随后给出的extern声明不能再初始化，例如第5行。

4.8 声明与定义

- 综上所述，声明与定义是有区别的。
- 定义创建对象且可以初始化，但定义在程序中只能有且只有一次；
- 声明向编译器提供对象类型和名字的信息，不能初始化，声明可以又多次重复多次，只要信息是统一即可；

- 在全局作用域上，可以让声明初始化，则此种写法的声明被当作定义，因而只能出现一次。

4.8 声明与定义

- 有了正确的声明与定义，就可以让对象的声明和定义分离，这种做法在单个文件的程序中没有太大作用；但是当程序有多个文件时，声明和定义的分离就是必须的。
- 处理这种情况的方法是：一个文件含有对象的定义，使用该对象的其他文件则包含该对象的声明。

- 将声明与定义分离（成两个文件），对于函数来说，实际上就是将（函数的）接口和（函数的）实现（或函数的定义）分离。从而为编写规模化程序奠定基础。


程序设计