高级函数特性

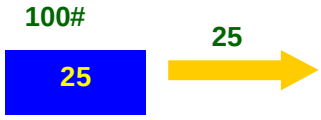
Andrew Huang<bluedrum@163.com>

目标

- 引用
- **默认参数**
- 内联函数
- **函数重载**

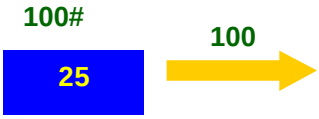
按值传递

- 函数调用中复制参数的值
- 函数只能访问自己创建的副本
- 对副本进行的更改不会影响原始变量



按引用传递

- 函数调用中传递参数的引用
- 主要优点
  - 函数可以访问主调程序中的实际变量
  - 提供一种将多个值从被调函数返回到主调程序的机制



向函数传递引用 2-1

- 引用提供对象的别名或可选名
- “&”告诉编译器将变量当作引用

```
void swap(int& i, int& j)
{
    int tmp = i;
    i = j;
    j = tmp;
}
void main()
{
    int x, y;
    swap(x,y);
}
```

向函数传递引用 2-2

- 引用就是对象本身
- 不要认为
  - 引用是指向对象的指针
  - 引用是该对象的副本

- 大的数据结构按引用传递，效率非常高

返回引用

- 返回引用不是返回变量的副本
- 函数头中包含一个“&”

```
int &fn(int &num)
{
   return(num);
}
void main()
{
   int n1, n2;
   n1 = fn(n2);
}
```

常量引用

- 用于不希望修改对象，以及要把大对象当作输入参数的情况
- 高效性和安全性

```
double distance(const point& p1, const point& p2);
```

- 将引用声明为常量，不能再绑定别的对象

```
int const &ri = num1;
```

函数

- 函数声明
  - 函数名
  - 函数返回值的类型
  - 函数的参数个数和类型
- 函数声明可以不包含参数名
- 调用函数时可以不指定全部参数

函数的默认参数

- 为可以不指定的参数提供默认值

```
void func(int = 1, int = 3, char = '*');
```

或

```
void func(int num1,int num2 = 3,char ch = '*');
```

参数的默认值 2-1

- 一旦给一个参数赋了默认值，后续所有参数也都必须有默认值

```
void errfunc(int num1=2,int num2,char ch='+');//错误
```

- 默认值的类型必须正确
- 默认值可以在原型或者函数定义中给出，但不能在两个位置同时给出
- 建议在原型声明中指定默认值

参数的默认值 2-2

- 调用上面声明的函数 func()

```
func(2,13,'+');
func(1);       //第二个和第三个参数采用默认值
func(2,25);    //第三个参数采用默认值
func();        //所有这三个参数都采用默认值
func(2,,‘+’);  //错误!
```

- 如果遗漏了中间的参数，编译器将报错

默认参数的优点

- 如果要使用的参数在函数中几乎总是采用相同的值，则默认参数非常方便
- 通过添加参数来增加函数的功能时，默认参数也非常有用

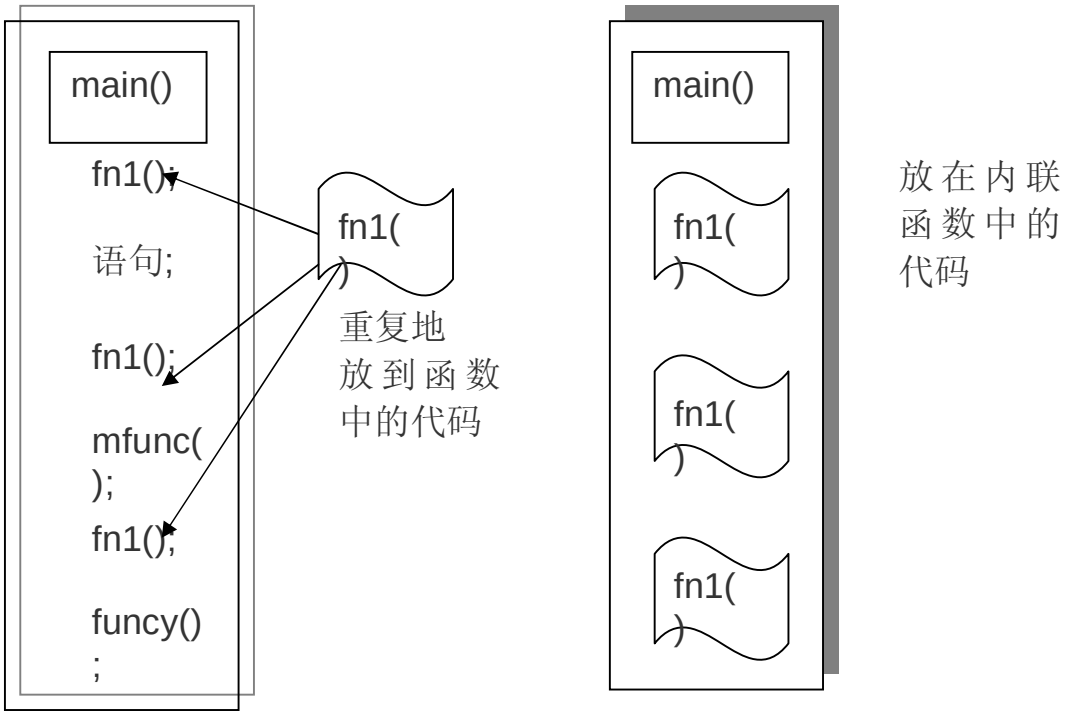
内联函数 2-1

- 通常的函数调用会节省内存空间，但是会花费一些额外的时间
- 内联函数节省短函数的执行时间

```
inline float converter(float dollars);
```

内联函数 2-2

- 非常短的函数适合于内联
- 函数体会插入到发生函数调用的地方



注意事项

- 编译器必须先看到函数定义，而不是声明
- 编译器有可能会忽略inline关键字
- 不允许为不同的源文件中的内联函数指定不同的实现

函数重载 2-1

- 具有相同的名称，执行基本相同的操作，但是使用不同的参数列表
- 函数多态性

```
void display();
void display(const char*);
void display(int one, int two);
void display(float number);
```

函数重载 2-2

- 编译器通过调用时参数的个数和类型确定调用重载函数的哪个定义
- 只有对不同的数据集完成基本相同任务的函数才应重载

函数重载的优点

- 不必使用不同的函数名
- 有助于理解和调试代码
- 易于维护代码

数据类型不同的重载

- 参数的类型不同，编译器就能够区分

```
int square(int);
float square(float);
double square(double);
```

- 同一函数名输出任何数据就是重载了输出函数

参数个数不同的重载

```
int square(int);       //函数声明
int square(int,int,int);
int asq = square(a)  //函数调用
int bsq = square(x,y,z)
```

- 编译器会调用参数匹配的函数
- 与函数的声明顺序无关
- 不会考虑返回类型

函数重载的作用域规则

- 重载机制只有在函数声明的作用域内才有效

```
class first{
  public:
    void display();
  };
  class second{
  public:
      void display();
  };
```

```
void main()
{
  first object1;
  second object2;
  //没有发生函数重载
  object1.display()
  object2.display();
}
```

总结

- 引用
- 默认参数

- 内联函数
- 函数重载

课堂练习