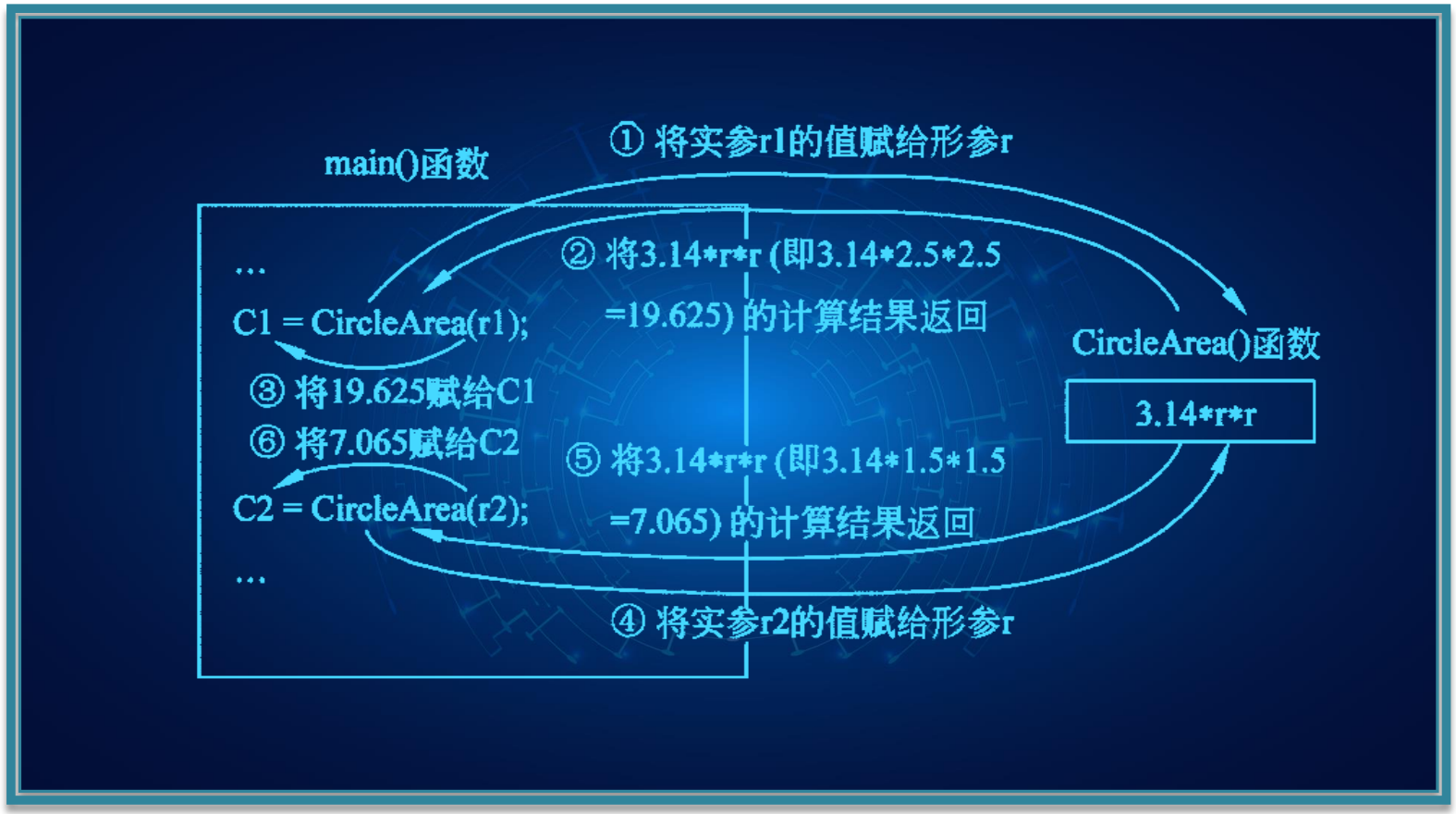
C++与C的主要差异 – 内联函数

内联函数

- 当调用一个函数时，系统会将当前函数的运行状态保存起来，然后再去执行被调用的函数；当被调用的函数执行完毕后，系统会将刚才保存的运行状态恢复，继续执行函数调用后面的运算。例如，对下图所示的函数调用，当调用CircleArea()函数时，系统会先保存main()函数的运行环境，再跳转到CircleArea()函数执行相应的操作；当CircleArea()函数执行结束后，系统会恢复刚才保存的main()函数的运行环境，继续执行CircleArea()函数调用后的运算（即将CircleArea()函数的返回值赋给C1或C2）。


main()函数
① 将实参r1的值赋给形参r
…
C1 = CircleArea(r1);
② 将3.14*r*r (即3.14*2.5*2.5
=19.625) 的计算结果返回
CircleArea()函数
3.14*r*r
③ 将19.625赋给C1
⑥ 将7.065赋给C2
⑤ 将3.14*r*r (即3.14*1.5*1.5
=7.065) 的计算结果返回
C2 = CircleArea(r2);
…
④ 将实参r2的值赋给形参r

- 运行环境的保存和恢复、函数跳转都要消耗一定的时间。如果被调用函数实现的功能比较复杂，其计算时间会远远大于函数调用所额外消耗的时间，此时函数调用所带来的额外时间开销就可以忽略不计。但是如果被调用函数实现的功能非常简单并且将被非常频繁调用，在编写程序时就必须要考虑因函数调用所造成的额外时间开销。
- 为了解决上述问题，一种比较好的方案就是使用内联函数。在编译程序时，系统会直接将调用内联函数的地方用内联函数中的语句体做等价替换。这样，在程序运行时就不需要函数调用，从而避免运行环境保存和恢复以及函数跳转所引起的额外时间开销，提高程序的执行效率。

内联函数定义的一般格式如下:

inline <函数类型> <函数名>([<形参表>])

{

函数体

}

在函数定义的<函数类型>前加上inline关键字，即为内联函数的定义。例如，对用于求圆面积的函数的功能非常简单，为了避免函数调用所引起的额外时间开销，可以将它定义成内联函数:

```
inline double CircleArea(double r)
{ return 3.14*r*r; }
```

- 提示:
- （1）内联函数只适用于功能简单的小函数。对于函数体中包含循环、switch等复杂结构控制语句的函数及语句比较多的函数，即便该函数被定义为内联函数，编译器也往往会放弃内联方式,将其作为普通函数处理。
- （2）必须在内联函数定义处给出inline关键字。如果仅在函数声明时给出inline关键字,而在函数定义时未给出inline关键字，则该函数会被编译器视为普通函数。