C++当中定义内联函数，可以让编译器将对内联函数的调用直接展开。

这就多少有点像宏定义了，而且没有宏定义的缺点（预处理替换，无法当成变量链接到符号表、调用有可能导致参数异常被改、等等）。

使用内联函数可以避免函数调用的开销（栈开辟、返回地址设定、栈展开），在一定的程度上可以提高程序的性能。但是这种提高是有代价的。
编译器将函数展开，会直接导致可执行程序变大。（导致运行缺页、cache 命中率降低 effective c++ page 135）

编译器在某些情况下会禁止 inline：

1：编译器禁止虚函数 inline

inline 展开是在编译时进行的（宏定义是在预处理时展开的），而虚函数是在运行时决定调用哪个函数的。因此编译器对虚函数的 inline 无能为力
(g++下试验过，编译器连警告都没有一个)

2：带有循环或者递归的调用

如何 inline 函数过于复杂，复杂到函数本身执行的成本，比函数调用（栈开销）成本还要高，编译器禁止 inline

3: 通过函数指针调用 inline 函数

虽然编译器会展开 inline 函数，但是还是会为内联函数生成函数本体。通过函数指针，使得其等于内联函数，通过函数指针调用，无法 inline

```
inline void f() {...}
void (*p)()=f;
p();   //此处无法内联
```