预处理：宏 ---> 宏常量 + 宏函数

宏常量：

优点：

1.可以提高代码的可读性

2.在代码中涉及到的所有宏常量，只要修改宏定义的位置就会全部改掉

在C++中 const 修饰的就是一个常量

```
int main() {
    const int a = 10;
    int array[a];   // a 是常量
    int *pa = (int *)&a;
    *pa = 100;
    cout << a << endl;
    cout << *pa << endl;
    return 0;
}
```

宏函数：

优点：宏函数不是真正意义上的函数，而且在预处理期间会对宏函数进

行展开，既然展开了就没有函数调用参数压展开销，此时运行效

率就会提高。

缺点：a) 没有进行类型检测

b) 不能调试

c) 代码膨胀

d) 副作用

```
#define MAX(left, right) (((left) > (right)) ? (left) : (right))
int main() {
    int a = 20;
    int b = 10;
    cout << MAX(a, b) << endl;      // 20
    cout << MAX(++a, b) << endl;    // 22
    return 0;
}
```

内联函数

```
inline int Max(int left, int right){
    return (left) > (right) ? left : right;
}
int main() {
    int a = 20;
    int b = 10;
    cout << Max(a, b) << endl;      // 20
    cout << Max(++a, b) << endl;    // 21
    return 0;
}
```

验证：内联函数是否会展开

inline 建议性关键字--建议将该函数当成内联函数来处理‘

编译器检测函数是否满足将其当陈内联函数的条件，如果满足，则与处理的

时候就进行宏函数展开，不满足则直接将 inline 关键字忽略掉，不进行宏

函数展开

```
inline int Add(int left, int right){
    return left + right;
}
int main() {
    int a = 10;
    int b = 20;
    Add(a, b);      // 如果没有展开，这里就是一个函数调用，因此可
                        以再汇编代码中找找，看有没有call命令，有就说
                        明也有展开，没有则说明展开了
    return 0;
}
```

auto 关键字(C++11)