函数重载

函数重载的条件

函数重载：C++允许函数重名
函数定义要素：返回值、函数名、参数列表、调用约定、函数体
参数列表：参数的类型、参数的个数、参数的顺序

条件

1. 函数名相同
2. 参数的类型不同**或**参数的个数不同**或**参数的顺序不同
3. 不考虑返回值与调用约定
4. 同作用域的函数

```
void foo(...)
{
    ...
}

namespace my_foo {
    void foo(...)       // 将外部foo屏蔽掉
    {
        ...
    }

    void foo(...)       // 重载命名空间中的foo
    {
        ...
    }
}
```

函数重载的编译器实现

使用名称粉碎，使用`undname`查看名称粉碎前的名字

```
Microsoft (R) C++ Name Undecorator
Copyright (C) Microsoft Corporation. All rights reserved.

Undecoration of :- "?foo@@YAHHH@Z"
is :- "int __cdecl foo(int,int)"
```

C++与C

C++调用C函数，使用关键字`extern "C"`告诉C++编译器，后面的名字使用C的名称粉碎规则
多个函数使用时：

```
extern "C" {
void foo1();
void foo2();
...
}
```

使用`extern "C"`将无法进行函数重载

函数匹配

1. 查找候选函数，查找所有指定名称的函数作为候选
    1. 若候选个数为0，则报错

        > error: 找不到标识符

    2. 若候选个数大于0，则查找匹配函数
2. 查找匹配函数：完全匹配（参数列表一致）和可以转换的函数（隐式类型转换）
    1. 若匹配的函数个数为0，则报错

        > error: 每一个一个可以转换所有参数类型

    2. 匹配函数的个数大于0，查找最佳匹配
3. 查找最佳匹配
    1. 若最佳匹配个数为1，匹配成功，调用该函数
    2. 若最佳匹配个数大于1，报错

        > error: 对重载函数的调用不明确

默认参数对函数重载的影响

```
void foo(int a, int b)
{
    ...
}

void foo(int a, int b, double c=3.14)
{
    ...
}

foo(1, 2);      // 报错，不知道调用哪个函数
```

assert断言

`assert`：调试或者测试期间查错，如果表达式为否，则会显示包含文件路径、代码行数等信息，然后调用`abort`退出程序
`static_assert(constexpr, "string")`：C++支持，编译期检查，如果表达式为假，则会输出编译错误信息为后面跟的字符串