g++ -o test main.cpp rational.cpp

每个源文件都被称为一个翻译单元(translation unit)，在某一个翻译单元里面定义的东西在另一个单元里使用正是链接发挥作用的地方。

1. 存储类(storage class)
   1. 决定了程序将变量的值存储在计算机上的什么地方、如何存储、以及变量应该有着怎样的作用域
   2. auto默认存储类，是默认的，看不到，储存在栈里
   3. static储存类，没有初始的变量存储在.rss，被声明过的static就存储在.data,全局变量也是在这
   4. extern存储类 相当于告诉编译器“请相信我，我发誓我知道这个变量在其他翻译单元里面肯定存在，它只是没有在这个文件里声明而已”
   5. register存储类 要求编译器把一个变量存储在CPU的寄存器里，但有着与自动变量相同的作用域
2. 链接
   1. 编译器构建程序3步骤
      1. 执行预处理指令；
         1. 替换相应头文件代码，搞得就像从一开始就在.cpp文件里面似的
      2. 把.cpp文件编译成.o文件(翻译单元)
         1. 变量分配内存并执行各种错误检查
      3. 把.o文件链接成一个可执行文件
         1. 多个源文件：编译好每一个组件之后，编译器需要把它们链接在一起才能生成最终的可执行文件
   2. 3种链接

凡是有名字的东西（函数、类、常量、变量、模板、命名空间。。。）必然属于其中之一

* + 1. 外链接(external):每个翻译单元都可以访问这个东西，前提是只要它知道有这么一个东西存在

普通函数、变量、模板和命名空间都有外链接，就像main.cpp可以使用rational.cpp文件里定义的类和函数一样，其实我们一直都在使用，只是今天我们来次总结。

//this.cpp

int a=1;

//that.cpp

int b=a;

//错误

//this.cpp

int a=1;

//that.cpp

extern int a;

int b=a;

//正确

//这样不行

* + 1. 内连接(interal):某个翻译单元里面定义的东西，只能在翻译单元里使用，在任何函数以外定义的静态变量都有内连接：

//this.cpp

static int d = 8;

//that.cpp

static int d = 9;

//这两个文件各有一个同名的变量，但是它们是毫不相干的，因为static 说明是一个内链接，只在内部使用，外部看不到。

* + 1. 无链接(none)：在函数里定义的变量只存在于该函数的内部，根本没有任何链接