C++ 有哪些缺点？

有相当一部分人对 C++ 是又爱又恨。那么应该有一些 C++ 自身的原因导致他的使用者（这里指的是熟练的使用者）对它有所不满。

提问的目的是试图通过获得一些观点，尝试设计一种比 C++ 更少陷阱的语言。因此，类似 C++ 对 C 的兼容这种历史原因的做法，不被视作缺点。

49 个回答

默认排序

陈硕

C++、编程、算法 话题的优秀回答者

209 人赞同了该回答

C++ 的函数重载决议规则是所有语言中最复杂的，因为他允许用户以两种方式自定义隐式类型转换。
比如有下面两个函数：

```
Employee* findEmployee(const std::string& surname, const std::string& givenName, bool retired = false);

Employee* findEmployee(const std::string& fullName, bool retired = false);
```

那么

```
Employee* e = findEmployee("Chen", "Shuo");
```

对应哪个？

如果原来代码里只有第一个函数，现在有人新增了第二个重载，会造成什么后果？

编辑于 2014-09-17

蓝色

C++、编程、编程语言 话题的优秀回答者

219 人赞同了该回答

这样的问题最适合随便乱侃大山了，甚至可能都答不到问题的点上，因为侃着侃着就会乱说了。

C++很复杂，这门语言有太多的诱惑，程序员需要极度的自律。而 C++设计成这样也是因为 Bjarne 说过他不希望把自己任何的喜恶都加在语言上，他希望程序员自己去判定，完全信任程序员。然而，信息时代发展到现在了，程序员也已经不像 80，90 时代一样，似乎是少数人才可以做得事情，现在可以写冒泡排序都可以找到一个编程开发工作，不需要理解计算机组织，不需要理解操作系统等等。而由于计算机基础的缺失，这样的程序员却往往是需要语言来帮助规范的，需要语言帮他选择，而非他来选择语言的特性。而这样的发展也是必然，如互联网时代，很多时候就是需要可以快速构建抢占市场，所以需要很快的搭建出来可运行的业务，而这时候很多情况下会选择动态语言。当然，这样很多时候也会有代价，当达到很大规模时，会回过头借助 Native 的语言，如 C++，而 Herb 也发表过 Why C++的 Presentation: youtube.com/watch?。不过很多企业还没有达到这样的规模就消失在尘埃中了……

也正是这样，C++具有着很大的复杂性，融合 OOP，GP 等各种编程范式于一身，而这也是对新手不利的地方。虽然现在的 C++已经进化了，C++11/14 对于新手来说已经

算友善了，少了很多历史残留的坑，但是 C++发展的路途增加新特性也是对新手的更大学习负担，如增加的右值引用，std::move，std::forward 到底又是什么东西？

C++的复杂性也体现在类型系统上，不知道多少人都被 C++的隐式类型转换给弄的头昏脑胀，如为什么会有 bool 转换这样的东西。同时，加入了泛型编程和右值引用后，那么类型推导也变得复杂。如：

```
template<typename T>void f(T&& param);

Object o;

f(o); // param type is lvalue reference

f(std::move(o)) // param type is rvalue reference
```

为什么明明是 T&&，却会让我的类型推导出不同的类型，这个时候对于初次接触的人都是不解。以前有句话叫做没有读过 Effective C++的 C++程序员不应该让他开发，我觉得这句话现在也可以引申为没有读过 Effective Modern C++的 C++程序员不应该让他开发，因为这本书写的真的很好很透彻，而这本书就详细解释了这个例子的原因。

下面我想说说上面几个用户提到的问题，如模块化系统，编译慢，ABI，异常等问题。

的确，C++在不断的进化，在不断的改善，也同时增加了新手的学习。如上面有人谈到C++没有模块系统，编译慢的问题，而C++17有一篇提案讲述的正是这个，叫做Module System，简单的语法：

```
Module M; // Module Declaration

// Export Interface to clientexport{

    int foo(){return 47;}};

void bar() {}
```

```
import std.vector; // #include <vector>import M;

using namespace std;

int main(){

    vector<int> v;

    foo();
```

```
bar(); // oops!}
```

而这篇提案也提到了有用户答的私有成员问题，如在 Module M 声明的 foo 是可见的，而 bar 则是完全不可见，而远非私有成员的不可访问，但可见。

而其实大家痛苦的很多地方，C++标准委员会的人也不是不知道，如 ABI 的问题，也有提案说想要做一个标准，这篇提案是微软的 Herb 提出的： isocpp.org/files/papers。而前一篇的 Module System 深得 Bjarne 的喜欢，而通过的几率其实也是蛮大的了，而 Herb 这篇我不知道，说不定也已经被毙了，只是我不知道。

对于异常，大家都在讨论用不用，而我前段时间也去研究过异常，把整个异常的实现都走通了一遍，我感叹着编译器实现异常的有趣，如 try throw 是如何一步一步的被 Runtime 捕获，异常的类型信息是如何插入到表，landing pad 的选取等 Exception Handling in LLVM。可是，对于用户来说，是否真的需要异常呢？我曾在一个回答中说 C++的异常是很鸡肋的东西，或者说的更直白点，要用好 C++的异常不容易，不是无脑的 try throw 就好，C++也不会像 Java 一样会强制要求你，C++的设计就是程序员自己做主。

而上面同样说到没有 Metadata，没有反射。而 C++17 也在考虑加入反射。是的，正如 C++标准委员会之前定的计划一样，C++11 是大改动，C++14 小修改，C++17 又是大改动。

总体来说，C++真的很复杂，或许现阶段还有 ABI，还有编译链接模型，还有类型系统，模块系统等的问题，其实 C++也许都可以解决，但是解决完后又变复杂了，因为要保持与之前的兼容，还要保证高性能这一根本立足点，这真是对智力的一大考验啊，所以我一直在说 C++标准委员会的人都蛮碉的，比如 C++11 提出的 Memory Model，真是把能抽象剥离的都抽象剥离出来了，很精细。

我是很喜爱 C++的，更是 Bjarne 的脑残粉，也靠着 C++在吃饭。我很欣喜的看着 C++不断的变好，但是也必须承认 C++还有很多不足，如很多知友和我上面提到的很多问题都的确是客观存在的。

发布于 2015-06-25

21921 条评论

分享

收藏感谢收起

丧心病狂大薯条

此账户弃用，暂使用另一个同名账户。

15 人赞同了该回答

连引用都有三种的语言。。。

发布于 2015-06-22

1510 条评论

分享

收藏感谢

cosagon i

11 人赞同了该回答

c++的第一个缺点 1 就是因为兼容 c，带来风格上的不一致。

缺点 2，c++没有反射和内省，导致有时候用起来麻烦，比如操作数据库，会有大量的手工代码。

另外很多人语病的，没有 gc，我不认为是什么缺点，我自己写的 c++程序，很少直接使用 new 和 delete，一般交给 stl 管理了，或者使用 make_shared 处理了。

另外一个 stl 带来的内存碎片，从语言层面来说，不是什么缺点，因为所有语言都会有内存碎片的问题，只是有些带 gc 的语言内置实现了内存池，比如 python，对于 c++ 这种要求适应各种环境的开发语言来说，内置内存池是不现实的(显然大多数客户端软件并不需要内存池)，内存碎片应该从设计层面解决，而不是语言层面解决，且 stl 本身也提供了解决方法。

还有一个所谓的二进制兼容问题，主要是开发组件的商业公司需要，windows 上有

com/winrt 组件帮你解决，linux 上基本都是开源的，并且 gcc/clang 编译出来的大多数库不存在二进制兼容问题(以后难说)，二进制兼容问题并不迫切。

总结来说，c++的最大缺点是，太灵活，导致复杂(这里复杂是指设计上的多样性)，但也是最大优点，从不会给你在软件设计上套上任何枷锁。

编辑于 2013-10-11

114 条评论

分享

收藏感谢

靳超

十八般代码耍的有莫有样

5 人赞同了该回答

1. C++在多线程情况下，如果不用 Actor 模型，对象的生命周期管理简直是噩梦

2. 用异常来处理错误？你见过整个服务器程序每个函数都用 try/catch 包一下的写法吗，对于没有正确理解 exception 的人来说，c++的异常最好的用法就是只在 main 函数中用

3. 对象之间的交互可用的模型太多，listener, callback, message, bind function, 导致不同开发者之间思维习惯大大不同

发布于 2015-06-22

54 条评论

分享

收藏感谢

裴草莓

写什么代码，回家种田

8 人赞同了该回答

其实现在感觉用起来感到十分明显的缺点并不来自 c++本身，正如蓝色大大所言，语言上的缺点其实都不是问题，天塌下来，有语言标准委员会顶着.真正到达应用起来的时候，需要注意的地方已经少了许多.而且我秉承的一贯开发模式就是设计的复杂，用的开心.stl 也是如此.

我觉得更多的缺点是在于学习曲线的陡峭造成的从业者能力参差不齐，比如我经常被同事吐槽写出看不懂的代码（事实证明我只是为了安全和易用在折磨自己），再比如有一些不明真相的小白盲目的崇拜类似 Google style 这样的“标准”，踏马的，里面一句 singleton 产生互相依赖就让它内存泄露，让我对这份标准黑到死，一个朋友说，至少

那份标准我有一半在反对，我可能比他水平差点，我反对 25%，c++就是这样，能产生出各种性格迥异的程序员，不服就干.还有一些自认为是 c++程序员的人，根本就不读书，还觉得自己特牛逼，如果 c++能屏蔽这部分人，不让这帮丫挺的出来祸害人，那就是完美.

发布于 2016-01-01

8 添加评论

分享

收藏感谢

于洋

PaddleDev github.com/PaddlePaddle

7 人赞同了该回答

补充一个资料，感觉还不错

C++ Frequently Questioned Answers

虽然那个 FQA 有点老了，很多也有不错的解决方法了，不过总的来说肯定比我写得好。

下面是原答案

1、编译慢。当年写 C++的时候，小项目编译的时候可以吃一个冰棍了。。当然可以用 ccache 和 distcc

2、人员混杂，会的不会的都说精通 C++，很难在开始筛选人。

3、没有 abi，所有的库基本上一个编译器就要有一个二进制。。。看看 Qt 就知道了，装一个 windows 版本的 Qt，所有的编译器都覆盖到能有几个 G 了

4、exception 基本上不可用。

话说，居然没人说编译慢这个问题。。我觉得这才是我不喜欢 Cpp 的根本问题。。。

编辑于 2015-07-07

715 条评论

分享

收藏感谢

Xi Yang

被 VS 和 MinGW 轮流喂屎

10 人赞同了该回答

看了一下 C++ FQA，感觉里面虽然有些偏激，但还是有不少非常深刻的内容。

1：C++没有很好地做到隐藏私有成员。私有成员必须要被写在头文件里，因为需要它们来计算类型的尺寸。
2：C++没有包、模块的概念，只有头文件。可以认为 1 也是这个问题的部分体现。

3：C++提供的运行时特性非常贫乏，只有那个简陋的 RTTI。实际上，复杂到需要 C++ 的系统，通常都已经复杂到了需要那些运行时的动态特性，比如反射。C++那种试图把一切东西在编译时定死的思想，我是不认同的。
至于那些坑太多一类的问题，就更不用提了。。。

我还是觉得可以把 C++砍成两个语言，一个是 C with template，另一个是加上 RTTI、反射、所有对象引用计数的“高级”语言。

再补充几个有时可以部分替代 C++的语言：D，Go，Rust，Vala，C#

编辑于 2014-09-17

1023 条评论

分享

收藏感谢

金山

搬砖民工

特性太多，太难掌握。

发布于 2015-06-25

0 添加评论

分享

收藏感谢

张云聪

分布式、流式计算

这几天整天因为这个编译不过：

```
boost::python::object tuple(boost::python::handle<>(_kv_tuple));
```

C++把这句话当成了个函数声明= =，没办法多加了个括号...

```
boost::python::object tuple((boost::python::handle<>(_kv_tuple)));
```

gcc 的鬼错误提示，完全看不出来哪里错了。

发布于 2015-06-27

04 条评论

分享

收藏感谢

奚衡

知乎海盗一枚

17 人赞同了该回答

这个问题属于老生长谈的问题

C++的生存空间正在被大大的挤压，现在有一种很普遍的认识就是 C++在做底层不如 C，在做企业应用又做不过 java，确实现在的现状令 C++的前途堪忧，C++的设计的初衷在很大程度是为了弥补 C 在面向对象的不足，但是设计使得其语法过于繁琐，相比与 java，.net，python，ruby 这样的语言，它实在是太重了

我个人在使用 C++的使用有非常多的困惑，我归结大致有这几点：

1. OOP 方面的困惑，比如虚函数，继承等问题

2. 指针和引用的困惑，到底什么时候用指针还是引用，这点也是相当的困惑，你会发现有些函数的接口提供的参数提供的指针，有的提供的是引用，这些地方是大用可深究之处的

3. 类型转换，这点也是另很多初学者困惑的地方，一不小心就会让你的程序踩到陷阱

4. 天书般的的模板和元编程，会让才接触的初学者相当困惑，早年我曾经阅读过俄罗斯人写的一个关于 svg 的 GDI+库，里面几乎全是类模板，看了让你泪奔

5. 内存释放，虽然可控性强，但是带来的问题很多，程序写大了，这个问题就更是非常难避免，即使有智能指针但是也不能解决所有的问题

关于 C++的诟病，Linus Torvalds 以前也是喷了不少，特别是他比较了 C 和 C++

不过以 C ++闻名于世的 Andrei Alexandrescu，我记得在 08 年来过中国，接受 csdn 的采访，当被问到 C++的前途，他引用了 scott meyer 的一段表述，你看现在什么语言写的软件是最赚钱的，呵呵，答案是微软的 office，office 就是 C++写的，不过现在这个家伙正在和其他几位 C++大佬在推 D 语言，当然有一个很明确的目的是改进 C++

总而言之，C++学习的成本很大，难度也最大，要学好真的不简单，C++的门徒都是在一堆的困惑中前进的

发布于 2012-04-26

收藏感谢

知乎用户

2 人赞同了该回答

简单来说，C++的根本问题就是，当你试图去解决一些 C++本身解决不了的问题的时候，所引入的解决方案往往会带来更多的问题。

实际上比 C++好用的语言多的是，不能理解尝试设计这样一门语言的意义何在？如果只是作为练习，那么是否比 C++好用根本没有必要作为考虑因素。

所有这些比 C++更好用的语言，相比 C++有两个主要缺陷：

1.性能；

2.路径依赖；

缺陷 1 在某种程度上已经不能算作是缺陷，性能有时不仅仅依赖于语言的好坏，还依赖于编译器和解释器的能力。现在已经出现一些编译器能编译出性能上媲美 C++的目标代码，比如 V8。但对于某些高可靠性的领域，C++还是有无可替代的优势。在这些领域，人的成本不是最高的。

缺陷 2 才是阻碍其它语言发展的真正因素。

编辑于 2015-06-25

28 条评论

分享

收藏感谢

知乎用户

3 人赞同了该回答

不能说更“好用”，只能说在合适的层面使用更合适的语言更佳。在较高封装层次使用 py/lua 等作为 oop 开发比较省力，我会喜欢多个语言结合使用，充分发挥各自的优点来完成一个作品，单纯用一个语言就想搞好各种东西是不靠谱的。至于说 C++的缺点，在较高层次下使用真的很不方便，远不如 py

编辑于 2012-04-26

34 条评论

分享

收藏感谢

沈文

删你评论只因你太 low

1 人赞同了该回答

1. 太大了，入门书就 1000 好几百页。

2. 太能干了，根据目测其实任何工作只需要一个小小的子集，不过小白很难专门去学那个子集....

发布于 2015-06-25

11 条评论

分享

收藏感谢

dccmx

搞技术的

37 人赞同了该回答

努力在更高层面做设计开发，却又被低层细节所累（典型的如内存管理，字符串处理）。

一旦接收了 C++中的 OOP，就上了船了，要实现一个能融入 C++类型系统的“完美”的 class 是何其繁琐。

语法很脏（一部分拜兼容 C 所赐，一部分源于支持多范型，还有一部分源于设计理念：不替开发者做选择，而追求无限的灵活性）。

比 C++更好用的语言？我找到一个：Go 语言。