怎样学好 C++

网络空间安全学院　张曙

QQ:949708317　e-mail:evianzhang1999@gmail.com

1. 我们学了什么？我们为什么要学这些？

C++ 这门课，对绝大多数同学来说，都是接触到的第一门编程课。这门课的目标主要有两个：

- 学习 C++ 的语法；
- 学习编程及算法的思想。

前者包括如何定义函数、如何声明类、如何写模板等，而后者主要包括迭代、递归的使用，排序、查找算法等。就未来的发展而言，这两者都是很有必要的。

就 C++ 的语法而言，我们现在不仅需要掌握基础的 C++ 语法，因为无论是嵌入式开发中用到的 C，还是 Android 开发中用到的 Java，都使用的是一套类似于 C++ 的语法。比如说一个计算阶乘的函数`factorial`：

C

```c
int factorial(int n)
{
    if (n == 0)
      return 1;
    else
      return n * factorial(n - 1);
}
```

C++

```cpp
int factorial(int n)
{
    if (n == 0)
      return 1;
    else
      return n * factorial(n - 1);
}
```

Java

```java
public static int factorial(int n) {
    if (n == 0) {
      return 1;
    } else {
      return n * factorial(n - 1);
    }
}
```

以上三种语言实现这一功能的代码，无论是语法还是逻辑上，都极其类似。

此外，作为一门面向对象的编程语言，C++ 中的类、继承、多态等重要思想在许多现代的面向对象编程语言中都有所体现，如 iOS 开发中的 swift，游戏编程中的 C# 等语言。因此，在学习 C++ 的过程中，打好 C++ 语法的基础，对将来快速入门新的编程语言是极其有帮助的。

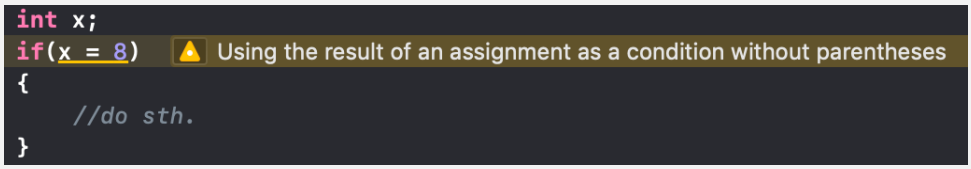
而对于我们在 C++ 课上学习到的编程及算法的思想，则会在将来我们编程的每时每刻都有所体现。如何用简洁的逻辑编写出一个高效的程序，也是我们将来最需要考虑的问题之一。而我们目前打好算法的基础，也可以为将来学习数据结构等专业课程铺平道路。

2. IDE

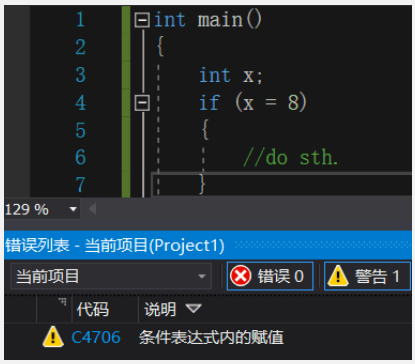
想要学好 C++，将来高效率地使用 C++，就需要一个好的 IDE（集成开发环境）。我推荐的 C++ 的 IDE 包括：最新版的 Visual Studio(Windows 系统), Xcode(macOS 系统), CLion, Code::Blocks, Dev-C++. 相比于命令行中未装插件的 vim 以及被微软淘汰的 VC++, 这些 IDE

具有更好的语法高亮、实时编译功能，也能最大可能地避免我们编程中的语法错误。例如，我们如果在条件语句的判断中错误地将"==" 打成了"=", 那么好的 IDE 就会给出 warning：

Xcode



Visual Studio



此外，老旧的 IDE 所使用的编译器版本也相对较低，也有可能造成意想不到的运行错误。

说到这里，就不得不说**遇到错误该怎么办**的问题。既然用了先进的 IDE，就应该充分利用编译器给出的报错信息。当我们费尽千辛万苦终于完成了代码的编写的时候，按下编译按钮，突然一片红色出现在眼前，控制台弹出无数 Error(s), Warning(s). 这时候该怎么办？这时候，第一件事，千万不是盯着报错的那一行，和代码大眼瞪小眼，一边揉搓着自己的头发，一边嘟囔："这怎么会错呢?" 这时候，应该眼盯控制台，看报错信息！看看究竟是括号少了一半，还是没有声明变量。这里，有一个最近才发生的例子，可以体现出查看报错信息的重要性：

npm 中有一个用来开发私密比特币钱包的库`bitpay/copay`, 某些不怀好意的人在其中加入了一些恶意代码，用于盗取比特币。由于并没有多少人在意库的源码，导致这一恶意代码广泛传播。但是，在一次 npm 的更新后，当人们使用该库的时候，得到了一个 warning 的信息：

```
[DEP0106] DeprecationWarning: crypto.createDecipher is deprecated.
```

这一信息引起了某些人的注意，在检查源码后，终于找到了恶意病毒。

由此可见，不关心报错信息，不仅会让自己陷入纠错-揪头发的怪圈，还有可能导致忽略巨大的隐患！

3. It is English that counts!

不光是学好 C++，学好任何一门编程语言，都需要很好的英文素养。我们来看一段代码：

错误示范

```
double calculateJiage(double danjia, int shuliang)
{
    return danjia * shuliang;
}
```

我们要知道，我们写代码不是光给自己看的，将来也是要和一个团队一起合作的。别人看着这个“双语”代码，心里会作何感想？不仅如此，由于汉语的一音多字性，如果写一个交易网站，这样命名的话，一个月后再看自己代码的时候，又怎么能知道`maiShangPin` 是要买还是卖呢？

此外，由于绝大多数的编程语言都是由英语母语者发展的，因此，一门语言最好的指导书、说明文章基本上都是用英语写的。尽管许多书都有汉化版，但是，不仅有许多书没有中文版，而且有的中文版的翻译，也会使人摸不着头脑。“鲁棒性”、“句柄”、“套接字”这些翻译，远不如英文原文 robustness, handler, socket 让人容易理解。因此，培养好自己的英文素养，能够阅读英文书籍是最好的选择。

有些人可能会想，我自己英文阅读本身就不好，还怎么能看专业性这么高的英文书籍呢？事实上，与英文散文、小说不同，除了专业性词汇以外，这些英文书籍里的用词其实并不高端，都是我们早已学过的单词和语法。我们只要在开始的时候辛苦一下，多记一些专业性的单词，之后就可以非常通畅地阅读下去了。比如说我们来看如下一段话：

> Consider the normal life cycle of a local variable in a function. The variable comes into existence during the execution of the function. At that time, memory is allocated on the **stack** (and possibly on the **heap**) to provide **storage** space for the value. The variable is used inside the function and then the function ends.
>
> (*Professional JavaScript for Web Developers*)

这段话是我随手从书中摘出来的。我们可以发现，除了我加粗的 stack, heap, storage 大家可能不认识以外，其他单词我们都认识，语法也没有什么复杂的从句、独立主格、悬垂分词之类的，都是我们日常看得懂的。而我们看不懂的单词，正是一些计算机专业的术语。因此，我们只需要在一开始的时候，多记一些术语，之后就能很轻松地阅读相应的英语书籍了。

4. 代码要有代码的样子

正如我们小时候练字的时候被灌输的那样，“字如其人”。事实上，代码的风格也如其人。我们来看下面两段代码：

错误示范

```
int func(int a,int b)
{
int c=a*b;
for(int i=1;i<c;i++)
{
if(i%2==1)
cout<<a<<endl;
else if(i%2==0)
cout<<b<<endl;
}
return c;
}
```

正确示范

```
int func(int multiplierA, int multiplierB)
{
    int product = multiplierA * multiplierB;
    for (int i = 1; i < product; i++)
    {
        if (i % 2 == 1)
          cout << multiplierA << endl;
        else
          cout << multiplierB << endl;
    }
    return product;
}
```

左边的代码让人一看就让人心里十分悲伤，欲哭无泪，不忍卒读。而右边的代码，一眼看上去十分清爽，肯定是一个好看的小哥哥/小姐姐写的。

5. 找资料与阅读

要想学好一门计算机语言，不仅是 C++，阅读是十分重要的。我们要阅读什么呢？我们又要怎样阅读呢？

第一，我们要学会从适当的地方找资料

当我们在编程的时候，难免会遇到各种各样的问题。怎样向函数里传递二维数组？inline 函数的本质是什么？虚函数表和多态有什么关系？有了这些问题，我们难免需要上网去搜索。除了百度以外，我们其实还有很多地方可以去查找：

- GitHub(https://github.com)
  这个网站被调侃为全球最大同性交友网站。来自世界各地、使用不同程序语言的程序员都会在这个网站上管理自己的代码、总结自己的编程经验。
- stack overflow(https://stackoverflow.com)
  该网站汇聚了无数的关于编程有关的问题，来自世界各地的人都会帮你解答。同时，由于是个国际性网站，所以你编程到半夜，在上面问问题也会有人帮你解答哦。
- CSDN(https://www.csdn.net)
  该网站汇聚了中国许多程序员，基本上我们目前遇到的所有问题都可以在该网站上得到解答，同时，也有许多资料都可以在该网站上下载。
- 简书 (https://www.jianshu.com)
  和 CSDN 一样，也有许多程序员把自己的经验总结在简书之中。

除了在这些网站中搜索问题，我们也可以在这些网站中看代码。作为编程刚刚入门的人，我们更需要的是多看代码。而那些久经考验的，高效率的代码，往往都总结在这些网站之中。所以，多看代码也是十分必要的。

资源在这里

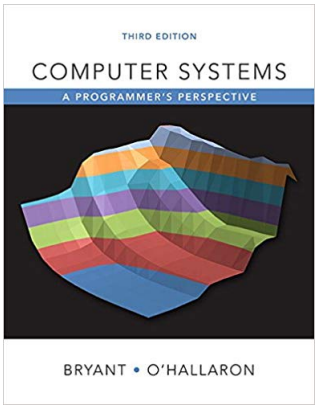
我推荐的书籍，都可以在GitHub上的网站（点我）中找到。此外，这里面还有许多有用的资源。

除了问问题、看代码以外，我们也需要不断地汲取新的有关编程、计算机的知识。这里，就需要我们选择合适的教材。但是，绝大多数教材又厚又贵，有的在国内还很难买到。因此，这就需要我们养成阅读电子书的习惯。有许多人说，自己不适应盯着电脑屏幕看书，学习效率不高。但是，这却是我们在大学阶段学习的最有效方法。进入大学以后，相关专业的教材一下子丰富起来，我们不可能每本教材都买或者借它的实体版来看，而且许多入门级别的教材我们也只是看过一遍之后就能了然于胸，之后编程遇到问题时，也一般就不再在这些入门书籍中寻找解答了。因此，绝大多数书籍，我们还是通过电子版来阅读，只有极少数需要反复翻阅的经典教材，才需要我们看实体版。

下面，我为大家推荐一些书籍。同时，正如我之前所讲的，优秀的教材一般最初都是用英文写的。因此，这些书籍，也大部分是英文的书籍。

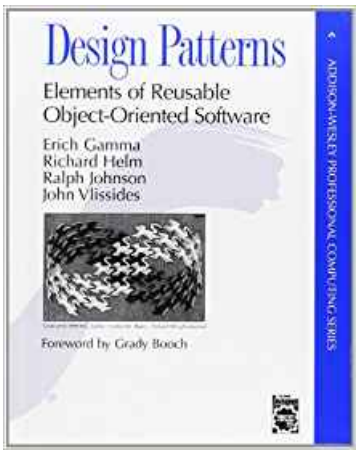
- 计算机基础类

  如果要想对计算机底层是怎样实现的进行大致的了解，一定要推荐的就是 *Computer Systems: A Programmer's Perspective*. 不需要多介绍，看了都说好，隔壁的程序员都馋哭了！



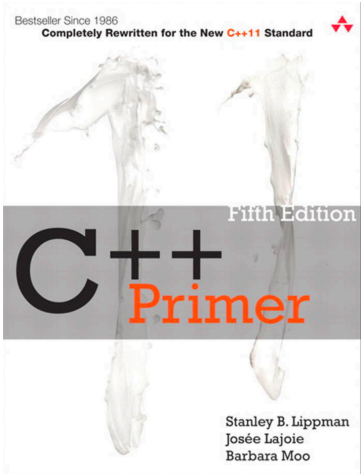
- 编程基础类

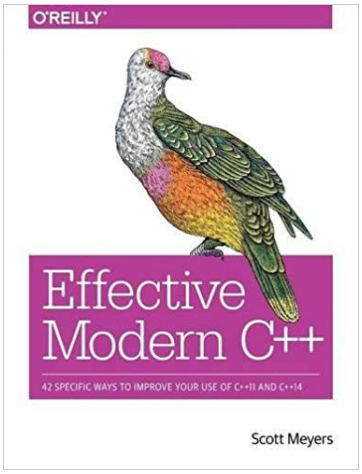
  我们要编程，要有明确的逻辑思维，同时，无论是什么语言，编程都会有一些通用的技巧。因此，设计模式十分重要。因此，关于设计模式，我推荐的是 *Design Patterns* 与 *Head First Design Patterns*. 前者业界经典，后者通俗易懂。



- C++

  首推 *C++ Primer*. 这本书涵盖了 C++ 许多基础的语法（所以很厚 _(:з」∠)_ ）。同时，还有 *Effective Modern C++*, 这也是本十分经典的书。





6. 提升自己

除了学习代码，程序员还有可以提升自己的地方。这里主要说两个：熟练运用 git 托管自己的代码，以及熟练运用 markdown 进行写作。这里只提供这两个方向，具体的细节还希望大家自己去寻找！

7. 关于 C++

根据我考试的经验来看，我们大一的程序设计及语言这门课程的考试，主要以 C++ 语法为主。因此，如何在考试中取得好成绩，主要还是看对 C++ 这门语言的掌握程度。而我在这里，不打算枯燥地一一列举语法，说每个语法的重点，而是想把我认为 C++ 里比较关键的想法和大家分享一下。

7.1 关于变量

首先，关于变量。C++ 中的变量，包含名字、值、类型、地址这四个方面。请看以下的程序：

代码

```
int main()
{
    int a = 2;
    int b = 3;
    float c = a;
    int &d = a;
    int *e = &a;
    cout << "a:" << a << endl;
    cout << "&a:" << &a << endl;
    cout << "b:" << b << endl;
    cout << "&b:" << &b << endl;
    cout << "c:" << c << endl;
    cout << "&c:" << &c << endl;
    cout << "d:" << d << endl;
    cout << "&d:" << &d << endl;
    cout << "e:" << e << endl;
    cout << "&e:" << &e << endl;
    return 0;
}
```

输出

```
a:2
&a:0x7ffeefbff668
b:3
&b:0x7ffeefbff664
c:2
&c:0x7ffeefbff660
d:2
&d:0x7ffeefbff668
e:0x7ffeefbff668
&e:0x7ffeefbff650
```

右边为该程序在某次运行后的输出。从输出我们可以看到，变量b 和变量a 的值、地址都不同，类型相同。变量c 和变量a 的地址不同，类型不同，值的大小相同（实际上在内存中的存储也不同）；变量d 和变量a 的名字不同，但是值、地址相同，类型从某种意义上也是相同的；变量e 的值为变量a 的地址，地址为自身独特的地址，类型为整型指针。

仔细研究上面的代码，相信大家就可以对变量有一个比较深刻的了解了。同时，这也可以加深大家对类的概念的理解：类，实际上是一种类型，而类的实例化对象，才是具有名字、值、类型、大小的一个实实在在的对象。

此外，通过对变量的分析，我们也可以更加清楚地了解函数实参向形参传递的过程。对于一个类型T1 和类型T2，如果有函数`void func(T1 a)` 和类型为T2 的一个变量b，那么，在语句`func(b);` 的过程中，实际发生了什么呢？实际发生的是`T1 a = b;` 这一语句的使用。我们知道，在 C++ 中，等号赋值是将右边的变量的值赋给左边的变量的值。因此，a 和b 实际上具有不同的地址。因此，在之后函数内对a 进行操作时，并没有改变b 的地址，从而也就没有对那个地址内存有的值进行操作。因此，无论在函数内对a 做了什么，都不会对b 产生影响。

最后，请大家思考一下，如果我想在函数内部改变一个传入的整型指针的指向，函数的形参类型应该是什么？如果这个问题能够想明白，相信就一定会对指针和引用有较好的了解了。（答案是`int *&`）

7.2 数组与指针

我们书上讲的，是数组名可以看作指向数组第一个元素地址的指针。这里为什么说“看作”呢？因为数组名其实不是指针。在我们定义`int array[50];` 的时候，出现了一个名字叫`array` 的“变量”。之所以打上引号，是因为它实际上不是一个变量。它的类型是`int[50]`，它的值是`array[0]` 的地址。但是，它没有地址。当对它使用取地址符`&array` 时，实际上

编译器对a 进行隐式类型转换，使其类型变为指针，地址为`array[0]` 的地址。同样地，当对它使用解引用符`*array` 时，也是将其隐式转化为指针。

但是，`sizeof(array)` 并不和别的指针一样占用一个机器字长的内存，而是返回整个数组的大小。此外，可以对指针p 使用`p++;` 这样的方法，但不能有`array++;`.

7.3 内存分配

正如我们之前讲到的，一个变量拥有名字、值、地址、类型四个属性。那么，这里的地址从何而来？自然是从计算机的内存中分配而来。在一个程序运行的过程中，计算机的内存分为两个部分：给这个程序用的，和不给这个程序用的。而给这个程序用的内存，又分为已经被这个程序用的，和还未被这个程序用的。

对于这些分配的细节，这里不再多说，之后自然会学到。这里要提到的是，一个变量，得到内存时的细节。当我们声明一个变量的时候，系统就会给这个变量从“给这个程序用的，但还没被这个程序用的”内存中找出一块来给这个变量。找多大的一块呢？对于变量a，分配给a 的内存大小为`sizeof(a)`. 也就是说，对于数组来说，在声明一个数组的时候，就一次性把整个数组的内存都分配给它了。

讲到内存分配，就不得不提new 运算符。关于动态内存分配的种种，这里也不再细说。但是，new 有一个特性：当需要被动态分配内存的变量的类型是一个类的时候，new 运算符会自动调用该类的构造函数，从而为这个类的成员变量分配空间并初始化。

7.4 多态与动态绑定

这里是大家经常绕不清的一个地方，特别是当类函数里调用别的类函数的时候。这里，提供一个小技巧：当一个类函数`funcA()` 的内部调用类函数`funcB()` 的时候，实际上是调用的`self.funcB()`. 而`self` 则是指向实例化对象的一个**指针**。牢记这一点，相信大家遇到多态的题目都可以迎刃而解了。

8. 最后

想学好 C++，学好编程，最重要的是多练习。要牢记在心，“我亦无他，唯手熟尔”。