第1章·C++快速入门

由于本章的定位是“快速入门”，所以我们注定会忽略很多细枝末节。但是这并不意味着它不重要。这些细枝末节往往对一个人的编码水平有着决定性的影响。所以我会在后面尽量提到“究竟忽略了哪些细节”，请读者自行查阅学习。相对于前面章节的“讲故事”，本章的内容可能会显得枯燥一些。但是作为后面章节的基础，希望大家能够倾尽全力去学习这些内容。

1. 关于C++

凡是语言，都一定有着自己的规则。而不同于英语、汉语这种自然语言，作为计算机所能识别的语言，它是需要在严格约束下避免二义性的，所以说我们将学到的C++语言，是一门语法规则非常严格，但又“灵活多变”的语言。

所谓非常严格，是指C++中不允许出现歧义，这种没有歧义的文法是经过严格的文法规则的限制、和一些约定来约束的。所以，如果在你的代码中出现了自己都会觉得有歧义的地方，往往就是程序中容易出错的地方。

而所谓灵活多变，就是指在掌握C++严格的语法之后，可以写出无穷无尽充满创意的程序。从黑框(控制台程序)，到带有图形界面(GUI)的程序，再到游戏，都可以利用C++的语法加上你脑中的算法搞定！而所有的这些灵活多变的内容，都是在严格的文法约束下实现的。所以说C++语言，是一门语法规则非常严格，但又灵活多变的语言！

C++是一门高级语言，所谓高级语言正如你们在前面章节中看到的，是不同于机器语言、汇编语言；能更加贴近人类思维的一门语言。C++是在C语言的基础上，增加了例如面向对象编程等诸多特性的程序设计语言。在程序设计竞赛中被广泛采用。

非常不严格地讲：我们常常所说的“C++”，包含了这门语言和一系列的工具。语言顾名思义就不多阐述。而工具实际上包含了很多内容：

我们将C++语言所编写的代码叫做“源代码”。而源代码是不能直接被计算机执行的（回想我们在上一章中讲述的内容）。而是应该利用编译程序，将源代码编译成计算机能够识别的机器代码然后再执行。而执行编译工作的工具就叫做“编译器”。

除此之外，仅仅有C++语言，我们连基本的功能（例如屏幕输出）都实现不了。即便是高级语言，想要与硬件设备交互，一般都需要直接编写一定的机器语言或者汇编语言，所以才能让我们的程序操作硬件：屏幕打印、屏幕输入、文件操作等等。而我们想要通过一己之力实现这些最底层的操作是很困难的。好在C++提供了一组可以被调用的工具。这些工具其实就是别人写好的代码，通过C++语法能够接受的形式所调用，最终辅助我们完成底层调用的功能，让我们的精力能够集中在更重要的算法设计上而不是千篇一律的与底层的通讯上。我们称这种“别人的代码”叫做“库”。就如同仓库一样，我们可以调用仓库中的内容来实现我们的意图。

那么我们的程序是怎么与别人的程序产生关联的呢？这有很多种情况，我只讲述最常见的一种：静态链接。我们的程序在调用库的时候，编译器可以仅仅在相应的位置上做个标记，表明这里的程序是调用了库而不是我自己的。这么做的好处就是可以充分地提高编译效率。因为库中的代码往往经过严格的编写、测试，基本能保证没有问题，所以我们只需要保存该库编译完毕的状态，没有必要在编译自己的程序时还要将别人的库再编译一遍。我们只需要编译自己的程序就可以了。所以我们的程序与库之间是并列关系，而不是包含关系。但是想要让自己的程序能够变成可执行程序，仅仅是这种“做标记”的方法是不可接受的。因为你所需要的库文件在你的电脑上会有，但并不意味着别人的电脑上也会有。再加上可执行程序的复杂原理，我们必须将这些静态库与我们的程序绑定在一起。而将编译完的“目标代码(Object Code)”与库文件链接形成“可执行文件”的工具叫做“链接器”。

也就是说，C++程序设计流程是：1，编写代码。2，编译成目标代码。3，链接成可执行文件。

一般来说，想要实现上面的流程，需要利用操作系统提供的黑框(Linux下的终端、Windows下的控制台)，再加上一大堆非常复杂的命令才能实现。编译命令甚至有可能比源代码还要复杂。如果需要对自己的代码进行调试，那可是更加麻烦。所以聪明人们就为懒人发明了又一件工具：集成开发环境（Integrated Development Environment，IDE）。所谓开发环境，可以将其看作是写代码的一个工具。我们可以在这个开发环境中编写代码、代码着色、自动排查错误等。所谓“集成”就是指将“编译器”、“连接器”都集中在IDE中。程序员可以简简单单地点击按钮，就可以实现整套编译链接运行调试等功能。

在ACM/ICPC的比赛中，通常使用的编译器是GCC(GNU Complier Collection)。GCC是一个支持很多种语言的开源的编译器的集合。它可以安装在Windows（不推荐）和Linux等多种操作系统上。而在比赛中普遍使用的是安装了Linux操作系统和GCC编译器的开发环境，并且配置了某些IDE来让学生进行开发。

关于IDE：对于Windows来说，我门推荐使用Dev-C++。对于Linux，我们推荐使用Code-Blocks。

1. 小试牛刀

说了这么多，是时候看一看C++的代码长得是什么样子了：

代码2-1,传说中的Helloword!

00 #include <iostream>

01 using namespace std;

02

03 int main()

04 {

05 cout<<”Hello IMUDGES!”<<endl;

06 return 0;

07 }

上面就是传说中的C++代码。实现的功能也很简单，输出“Hello IMUDGES”。关于程序的前两行，请参考课后习题2，大家可以查阅资料自行了解。我们主要来看3-6行。

第3行中我们看到了int main()这种写法。在这里你只需要知道，这是一个名字叫做main，返回值是int的函数。具体什么是函数先不用管。第4和第6行之间，用大括号括起来了一段代码。这表明的这部分代码是属于main函数中的。

程序想要执行，必须要找到执行的入口，也就是第一行执行的操作。在C++中，这个入口就是main()函数。程序将会在main函数中第一行数据开始依次执行，而执行完毕，这个main()函数也会返回一个值给操作系统。

也就是说，你所写的程序中，必须有一个叫做main()的函数。而你应该将你要进行的操作放在这个函数中。

如果大家对于上面的这部分内容难以理解也没关系，因为在后面你的教程中我将会为大家仔细解释个中缘由！

1. 高级语言程序的基本结构

正像说话和写文章需要分节分段一样，高级语言的程序也具有层次结构。尽管细节有着些许差别，但是绝大部分高级语言之间对于层次结构都惊人地相似。在很多高级语言中，程序一般由若干个文件构成，文件由类型定义、函数等组成，而函数则是由多条语句按照一定的编程风格组织在一起的。

1. 小结、习题etc

习题1.查阅资料解释什么是目标文件？目标文件是什么工具产生的？内部应该有哪些内容？

习题2.什么是#include <iostream>？“std::”这种形式表示什么？

1. s