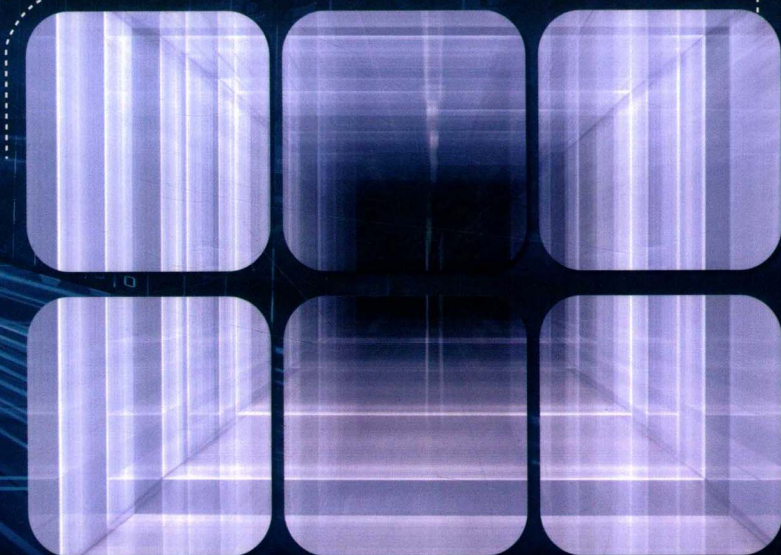


C++ 语言程序设计

姚娟 汪毅 主编



 科学出版社

C++语言程序设计

姚娟 汪毅 主编

科学出版社

北京

内 容 简 介

本书结合编者多年 C++ 语言程序设计的实践教学经验,按照“C++ 语言程序设计”课程教学要求编写,较为深入地介绍 C++ 语言程序设计基本理论、基本知识和基本方法,包括结构化程序设计、面向对象程序设计及程序设算法基础。

本书共 13 章,主要介绍 C++ 语言程序设计各个方面的知识,包括 C++ 语言概述、基本数据类型和表达式、控制结构、数组、函数、指针、结构与共用体、类和对象、运算符重载、继承和派生、多态性、输入/输出流、简单数据结构及算法。

本书内容丰富、概念清晰,并有配套的实验和习题指导书《C++ 语言程序设计习题与实验指导》(姚雅鹏、石礼娟主编,科学出版社出版),可作为高等院校 C++ 语言程序设计课程的教材,也可作为计算机爱好者的自学用书。

图书在版编目(CIP)数据

C++ 语言程序设计/姚娟,汪毅主编. —北京:科学出版社,2018.8

ISBN 978-7-03-058190-7

I. ①C… II. ①姚… ②汪… III. ①C 语言-程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2018)第 140774 号

责任编辑:戴薇 王国策 王会明 / 责任校对:陶丽荣

责任印制:吕春珉 / 封面设计:东方人华平面设计部

科学出版社出版

北京东黄城根北街 16 号

邮政编码:100717

<http://www.sciencep.com>

三河市骏杰印刷有限公司印刷

科学出版社发行

各地新华书店经销

*

2018 年 8 月第 一 版 开本:787×1092 1/16

2018 年 8 月第一次印刷 印张:21 1/2

字数:491 000

定价:60.00 元

(如有印装质量问题,我社负责调换(骏杰))

销售部电话 010-62136230

编辑部电话 010-62135397-2008

版权所有,侵权必究

举报电话:010-64030229; 010-64034315; 13501151303

计算机是人类智慧的结晶，是脑力劳动机械化、自动化的成功典范。在当今及未来，计算机都是科技进步、社会发展不可或缺的得力助手。

计算机是靠人来控制的。人们控制计算机使用的就是计算机程序设计语言。计算机程序设计语言，尤其是高级语言的出现和发展是计算机科学中富有智慧的成就之一。计算机程序设计语言是人为制订的一整套功能近乎完美的算法思想表达体系和计算机行为规范准则。学会一门计算机程序设计语言便掌握了一种掌控计算机的本领。

C++语言既是目前广泛使用的一种程序设计语言，又是众多学习程序设计和从事软件开发人员的首选语言。它既支持面向过程程序设计，又支持面向对象程序设计；既适合作为教学及训练用计算机程序设计语言（适合作为高等院校相关专业第一门程序设计课程的语言进行学习），又能用于大型软件开发，特别是集体开发大型软件。

本书是面向没有程序设计基础的读者而编写的入门教材。通过本书的学习，读者应掌握使用C++语言设计应用程序的基本技能，了解面向对象和结构化程序设计的方法，能够编写、调试和运行实用、规范、可读性好的C++语言程序。本书在内容组织上循序渐进，不要求读者学过程序设计方面的先修课程。

全书共13章，分为3个部分：

第1部分为第1章~第7章，介绍C++语言编程的基本内容，包括控制结构、基本数据类型、表达式、函数、指针等。

第2部分为第8章~第12章，介绍类与对象、继承和多态性等面向对象程序设计的基础理论，以及C++语言的输入/输出流。

第3部分为第13章，介绍基本数据结构和简单算法。

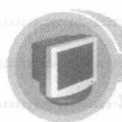
本书中的全部程序源代码均在CodeBlocks 13.12集成开发环境中调试通过。本书配有专门的习题及实验指导书。本书内容涵盖C++语言的基本语法、面向对象的概念和程序设计方法，鉴于课时所限，教师在教学组织上可结合学生特点适当取舍。

本书的编写人员全部是多年从事一线教学的教师，具有丰富的教学经验。本书由姚娟、汪毅担任主编。第1章、第13章由胡滨编写；第2章、第12章由郑芳编写；第3章、第4章由彭明霞编写；第5章、第6章由杨莉萍编写；第7章由邓君丽编写；第8章、第9章由汪毅编写；第10章、第11章由姚娟编写。姚娟负责全书的总体策划与统稿、定稿工作。编者在编写本书的过程中得到了科学出版社的大力支持和帮助，许多长期致力于C++语言教学的教师也对本书提出了许多宝贵的意见和建议，在此表示衷心的感谢。同时，对编写过程中参考的大量文献资料的作者一并致谢。

由于编者水平有限且时间仓促，书中难免有不足之处，敬请专家、读者批评指正，以便使本书不断完善。

编者

2018年5月



第 1 部分 结构化程序设计

| | |
|------------------|----|
| 第 1 章 C++语言概述 | 3 |
| 1.1 C++的产生 | 3 |
| 1.2 计算机上运行程序的方法 | 4 |
| 1.3 程序设计的基本概念 | 5 |
| 1.4 C++程序的基本结构 | 7 |
| 1.5 C++程序的基本要素 | 8 |
| 1.5.1 字符集 | 8 |
| 1.5.2 标识符 | 9 |
| 1.5.3 关键字 | 9 |
| 1.5.4 注释 | 9 |
| 1.5.5 简单的输入与输出 | 10 |
| 1.6 计算机的工作原理 | 12 |
| 第 2 章 基本数据类型和表达式 | 14 |
| 2.1 基本数据类型 | 14 |
| 2.1.1 整型 | 14 |
| 2.1.2 实型 | 18 |
| 2.1.3 字符型 | 19 |
| 2.1.4 布尔型 | 20 |
| 2.2 常量和变量 | 20 |
| 2.3 变量的定义和初始化 | 21 |
| 2.3.1 变量的定义 | 21 |
| 2.3.2 变量的初始化 | 22 |
| 2.4 运算符与表达式 | 23 |
| 2.4.1 算术运算符及其表达式 | 23 |
| 2.4.2 赋值运算符及其表达式 | 24 |
| 2.4.3 关系运算符及其表达式 | 25 |
| 2.4.4 逻辑运算符及其表达式 | 26 |

| | | |
|------------|---------------------|-----------|
| 2.4.5 | 自增和自减运算符及其表达式 | 27 |
| 2.4.6 | 位运算 | 29 |
| 2.4.7 | 其他运算符 | 31 |
| 2.5 | 表达式中运算符的运算顺序 | 32 |
| 2.6 | 类型转换 | 33 |
| 2.7 | 语句 | 35 |
| 第3章 | 控制结构 | 36 |
| 3.1 | 顺序结构 | 36 |
| 3.2 | 选择结构 | 37 |
| 3.2.1 | if 语句 | 37 |
| 3.2.2 | switch 语句 | 44 |
| 3.3 | 循环结构 | 48 |
| 3.3.1 | while 语句 | 49 |
| 3.3.2 | do-while 语句 | 50 |
| 3.3.3 | for 语句 | 52 |
| 3.3.4 | break 和 continue 语句 | 56 |
| 3.3.5 | 双重循环 | 59 |
| 3.3.6 | 循环和选择结构的嵌套 | 62 |
| 第4章 | 数组 | 69 |
| 4.1 | 数组的基本概念 | 69 |
| 4.2 | 一维数组 | 70 |
| 4.2.1 | 一维数组的声明和初始化 | 70 |
| 4.2.2 | 一维数组的使用 | 72 |
| 4.2.3 | 一维数组的程序实例 | 74 |
| 4.3 | 二维数组 | 76 |
| 4.3.1 | 二维数组的声明和初始化 | 77 |
| 4.3.2 | 二维数组的使用 | 78 |
| 4.3.3 | 二维数组的程序实例 | 78 |
| 4.4 | 字符数组 | 82 |
| 4.4.1 | 字符数组的声明和初始化 | 82 |
| 4.4.2 | 字符串的输入与输出 | 83 |
| 4.4.3 | 字符串处理函数 | 85 |
| 第5章 | 函数 | 94 |
| 5.1 | 函数的定义和使用 | 94 |

| | | |
|-------|--------------|-----|
| 5.1.1 | 函数的定义 | 94 |
| 5.1.2 | 函数的返回值 | 95 |
| 5.1.3 | 函数原型 | 97 |
| 5.1.4 | 函数的调用 | 98 |
| 5.2 | 函数的参数传递 | 100 |
| 5.2.1 | 传递参数值 | 100 |
| 5.2.2 | 引用作为函数参数 | 101 |
| 5.2.3 | 传递地址值 | 104 |
| 5.3 | 函数的嵌套调用和递归调用 | 104 |
| 5.3.1 | 函数的嵌套调用 | 104 |
| 5.3.2 | 函数的递归调用 | 107 |
| 5.4 | 数组与函数 | 111 |
| 5.5 | 函数的其他用法 | 114 |
| 5.5.1 | 内联函数 | 114 |
| 5.5.2 | 带默认参数值的函数 | 115 |
| 5.5.3 | 函数重载 | 116 |
| 5.5.4 | 函数模板 | 119 |
| 5.6 | 变量的作用域和存储类型 | 123 |
| 5.6.1 | 全局变量和局部变量 | 123 |
| 5.6.2 | 静态变量和动态变量 | 127 |
| 第 6 章 | 指针 | 131 |
| 6.1 | 指针的基本概念 | 131 |
| 6.2 | 指针的声明与初始化 | 131 |
| 6.2.1 | 指针的声明 | 131 |
| 6.2.2 | 指针的初始化 | 132 |
| 6.3 | 指针运算 | 133 |
| 6.3.1 | 取地址运算与取值运算 | 133 |
| 6.3.2 | 指针的算术运算 | 134 |
| 6.3.3 | 指针的关系运算 | 135 |
| 6.3.4 | 指针的相减和赋值运算 | 136 |
| 6.4 | 指针与数组 | 137 |
| 6.4.1 | 指针与一维数组 | 137 |
| 6.4.2 | 指针和二维数组 | 143 |
| 6.4.3 | 指针数组 | 145 |
| 6.5 | 动态内存分配 | 147 |
| 6.5.1 | new 运算符 | 147 |

| | | |
|------------------------|----------------|------------|
| 6.5.2 | delete 运算符 | 149 |
| 6.6 | 指针与函数 | 150 |
| 6.6.1 | 指针作为函数参数 | 151 |
| 6.6.2 | 函数返回指针 | 153 |
| 6.6.3 | 指向函数的指针 | 154 |
| 第 7 章 | 结构体与共用体 | 159 |
| 7.1 | 枚举类型 | 159 |
| 7.1.1 | 枚举类型的定义和声明 | 159 |
| 7.1.2 | 枚举类型变量的使用 | 161 |
| 7.2 | 结构体 | 163 |
| 7.2.1 | 结构体类型的定义 | 163 |
| 7.2.2 | 结构体类型变量的定义方法 | 164 |
| 7.2.3 | 结构体变量的使用 | 166 |
| 7.2.4 | 结构体变量的初始化 | 167 |
| 7.2.5 | 数组和结构体 | 167 |
| 7.2.6 | 函数和结构体 | 171 |
| 7.3 | 共用体 | 175 |
| 7.3.1 | 共用体的定义 | 176 |
| 7.3.2 | 共用体变量的定义 | 176 |
| 7.3.3 | 共用体变量的使用 | 177 |
| 第 2 部分 面向对象程序设计 | | |
| 第 8 章 | 类和对象 | 183 |
| 8.1 | 面向对象程序设计方法 | 183 |
| 8.1.1 | 面向对象的基本概念 | 183 |
| 8.1.2 | 面向对象的基本特征 | 184 |
| 8.2 | 类与对象的声明和定义 | 187 |
| 8.2.1 | 类的声明 | 187 |
| 8.2.2 | 对象的定义 | 187 |
| 8.3 | 类的成员 | 190 |
| 8.3.1 | 数据成员 | 190 |
| 8.3.2 | 函数成员 | 190 |
| 8.4 | 对象的存储和访问 | 191 |
| 8.4.1 | 对象的存储 | 191 |

| | | |
|---------------|-------------------|------------|
| 8.4.2 | 对象的访问 | 193 |
| 8.4.3 | 指向对象的指针 | 196 |
| 8.4.4 | this 指针 | 198 |
| 8.5 | 构造函数 | 199 |
| 8.5.1 | 构造函数的定义 | 199 |
| 8.5.2 | 默认构造函数 | 201 |
| 8.5.3 | 构造函数的初始化表 | 202 |
| 8.5.4 | 复制构造函数 | 202 |
| 8.6 | 析构函数 | 206 |
| 第 9 章 | 运算符重载 | 210 |
| 9.1 | 用函数实现运算 | 210 |
| 9.2 | 运算符重载函数 | 214 |
| 9.2.1 | 运算符重载为成员函数 | 214 |
| 9.2.2 | 运算符重载为非成员函数(友元函数) | 216 |
| 9.3 | 典型运算符的重载 | 219 |
| 9.3.1 | 重载求相反数运算符 | 219 |
| 9.3.2 | 重载“++”运算符 | 221 |
| 9.3.3 | 重载赋值运算符 | 224 |
| 9.3.4 | 重载下标运算符 | 227 |
| 第 10 章 | 继承和派生 | 230 |
| 10.1 | 继承的基本概念 | 230 |
| 10.2 | 派生类 | 233 |
| 10.2.1 | 派生类的声明 | 233 |
| 10.2.2 | 派生类的构成 | 234 |
| 10.2.3 | 派生类的继承方式和访问属性 | 234 |
| 10.3 | 派生类的构造函数和析构函数 | 241 |
| 10.3.1 | 派生类的构造函数 | 241 |
| 10.3.2 | 派生类的析构函数 | 242 |
| 10.4 | 多继承与虚基类 | 246 |
| 10.4.1 | 多继承 | 246 |
| 10.4.2 | 多继承中的二义性问题 | 250 |
| 10.4.3 | 虚基类 | 252 |
| 第 11 章 | 多态性 | 258 |
| 11.1 | 多态性的概念 | 258 |

| | | |
|----------------------|------------------|------------|
| 11.2 | 派生类对象替代基类对象 | 260 |
| 11.3 | 虚函数 | 262 |
| 11.3.1 | 虚函数的定义 | 262 |
| 11.3.2 | 虚函数的使用限制 | 265 |
| 11.3.3 | 虚析构函数 | 270 |
| 11.4 | 抽象类 | 272 |
| 第 12 章 输入/输出流 | | 275 |
| 12.1 | 输入/输出流的基本概念 | 275 |
| 12.1.1 | 流类库的头文件 | 275 |
| 12.1.2 | 输入/输出流类库的体系 | 276 |
| 12.1.3 | 输入/输出流实例 | 277 |
| 12.1.4 | 重载“<<”和“>>”运算符 | 283 |
| 12.2 | 格式化输入/输出 | 285 |
| 12.2.1 | 用 ios 类成员函数进行格式化 | 285 |
| 12.2.2 | 用操作符进行格式化控制 | 287 |
| 12.3 | 文件流 | 288 |
| 12.3.1 | 文件流类及文件流对象 | 290 |
| 12.3.2 | 文件的打开和关闭 | 291 |
| 12.3.3 | 文本文件的读与写 | 293 |
| 12.3.4 | 二进制文件的读与写 | 296 |
| 12.3.5 | 文件的随机访问 | 297 |

第 3 部分 程序设计算法基础

| | | |
|-------------------------|---------|------------|
| 第 13 章 简单数据结构及算法 | | 303 |
| 13.1 | 查找 | 303 |
| 13.1.1 | 查找的基本概念 | 303 |
| 13.1.2 | 顺序查找 | 304 |
| 13.1.3 | 折半查找 | 304 |
| 13.2 | 排序 | 306 |
| 13.2.1 | 冒泡排序 | 306 |
| 13.2.2 | 选择排序 | 309 |
| 13.2.3 | 直接插入排序 | 310 |
| 13.3 | 线性表概述 | 312 |

| | |
|---------------------|-----|
| 13.4 链表的实现..... | 314 |
| 13.4.1 结点类的实现..... | 314 |
| 13.4.2 单链表类的实现..... | 316 |
| 13.4.3 链表的应用实例..... | 327 |
| 参考文献 | 331 |

第 3 部分

结构化程序设计

在1970年代后期和1980年代初期的支持面向对象程序设计的语言。本章主要介绍C语言的基本概念和编程方法进行介绍。

第1部分

结构化程序设计

1964年，IBM公司为计算机软件开发人员开发了记述语言BCPL(Basic)。1970年，Ken Thompson在继承BCPL优点的基础上，开发了实用的B语言。1972年在B语言的基础上进一步扩充和完善，设计出了C语言。当时，设计C语言是为了编写UNIX操作系统。之后C语言经过多次改进并开始流行。C++语言是在C语言基础上发展和完善的，而C语言发展成了C#语言的优势逐步成为实用性很强的语言。

C语言的主要特点如下。

(1) C语言是一种结构化的程序设计语言，语言简洁且使用灵活方便。它既适用于设计编写大的系统程序，又适用于编写小的控制程序，还可用于科学计算。

(2) C语言既有高级语言的特点，又有汇编语言的特点。C语言运算符丰富，除了提供常用的算术逻辑运算符外，还提供了二进制的位运算符，同时提供了灵活的数据结构。加上C语言编写的程序表述灵活方便，功能强大。用C语言开发的程序结构性好，执行效率高。

(3) C语言的可移植性好。用C语言在某一型号的计算机上开发的程序，基本上可以移植到另一种型号的计算机上运行。

(4) C语言的程序结构不够严密，程序设计的自由度大。对于比较精通C语言的程序员来说，可以说游刃有余的、通用的程序。但对于初学者来说，要比较熟练运用C语言来编写程序，并不是一件容易的事情。与其他高级语言相比，C语言调试程序比较困难，往往是编好程序输入、编译后，编译并容易通过，但在执行时仍会出错。但只要能掌握C语言的语法规则，编写程序及调试程序还是比较容易掌握的。

随着C语言应用的推广，C语言存在的一些缺陷也开始暴露出来，并受到人们的关注。例如，C语言的数组不能动态增长、缺少支持代码重用的结构。随着软件工程规模的扩大，难以用于开发特大型程序。

为了克服C语言本身固有的缺点，并提供C语言编程、设计一种与汇编语言接近的语言。1980年，贝尔实验室的Bjarne Stroustrup及其同事对C语言进行了改进和扩充，并引入面向对象类的概念形成C++语言。

第1章 C++语言概述

C++语言是在C语言的基础上发展起来的支持面向对象程序设计的语言。本章主要对C++语言的基本概念和编程方法进行介绍。

1.1 C++的产生

20世纪60年代, Martin Richards 为计算机软件开发人员开发了记述语言 BCPL(basic combined programming language)。1970年, Ken Thompson 在继承 BCPL 优点的基础上发明了实用的 B 语言。1972年, 贝尔实验室的 Dennis Ritchie 和 Brian Kernighan 在 B 语言的基础上进一步充实和完善, 设计出了 C 语言。当时, 设计 C 语言是为了编写 UNIX 操作系统。之后 C 语言经过多次改进并开始流行。C++语言是在 C 语言基础上发展和完善的, 而 C 语言吸收了其他语言的优点逐步成为实用性很强的语言。

C 语言的主要特点如下。

1) C 语言是一种结构化的程序设计语言, 语言简洁且使用灵活方便。它既适用于设计和编写大的系统程序, 又适用于编写小的控制程序, 还可用于科学计算。

2) 它既有高级语言的特点, 又具有汇编语言的特点。C 语言运算符丰富, 除了提供对数据的算术逻辑运算符外, 还提供了二进制的位运算符, 同时提供了灵活的数据结构。用 C 语言编写的程序表述灵活方便, 功能强大。用 C 语言开发的程序结构性好, 目标程序质量高, 程序执行效率高。

3) 程序的可移植性好。用 C 语言在某一种型号的计算机上开发的程序, 基本上可以不做修改, 而直接移植到其他不同型号和不同档次的计算机上运行。

4) 程序的语法结构不够严密, 程序设计的自由度大。对于比较精通 C 语言的程序设计者来说, 可以设计出高质量的、通用的程序。但对于初学者来说, 要比较熟练运用 C 语言来编写程序, 并不是一件容易的事情。与其他高级语言相比, C 语言调试程序比较困难, 往往是编好程序输入计算机后, 编译时容易通过, 但在执行时仍会出错。但只要真正领会 C 语言的语法规则, 编写程序及调试程序还是比较容易掌握的。

随着 C 语言应用的推广, C 语言存在的一些缺陷也开始显露出来, 并受到人们的关注。例如, C 语言的数据类型检查机制比较弱、缺少支持代码重用的结构, 随着软件规模的扩大, 难以用于开发特大型程序等。

为了克服 C 语言本身存在的缺点, 并保持 C 语言简洁、高效, 与汇编语言接近的特点, 1980年, 贝尔实验室的 Bjarne Stroustrup 博士及其同事对 C 语言进行了改进和扩充, 并把 Simula 67 语言中类的概念引入 C 语言。1983年, 由 Rick Maseitti 提议将改进

的 C 语言正式命名为 C++ (C Plus Plus)。研制 C++ 语言的目标是使其既要继承 C 语言的所有优点, 又要根除 C 语言中存在的问题。后来, 人们又把运算符的重载、引用、虚函数等功能引入 C++ 语言中, 使 C++ 语言的功能日趋完善。

C++ 语言是 C 语言的一个超集, 对 C 语言是兼容的。C++ 语言对 C 语言的兼容也使 C++ 语言受到了一定的限制。因为 C 语言是面向过程的语言, 为了保持这种兼容性, C++ 语言必须支持 C 语言的面向过程特性, 这样, 两种不同风格的程序设计技术融于同一种语言中, 使 C++ 语言不是一个纯正的面向对象语言, 所以 C++ 语言实际上是混合型的面向对象语言。

在一个纯粹的面向对象语言中, 强调开发快速原型的能力, 而混合型的面向对象语言是在传统过程化语言中加入了各种面向对象的语言机制, 它所强调的是开发效率和运行效率。

当前用得较为广泛的 C++ 开发环境有 Visual C++、CodeBlocks、Borland C++ 等。下面简要介绍 CodeBlocks。

对于大多数编程人士来说, CodeBlocks 是一个很强大的编程工具; 对于编程初学者来说, 它是一个易学易懂的编程工具。CodeBlocks 是一个开放源码的、全功能的跨平台 C/C++ 集成开发环境。CodeBlocks 由 C++ 语言开发完成, 使用了著名的图形界面库 wxWidgets。CodeBlocks 自发布开始就成为跨越平台的 C/C++ 集成开发环境, 支持 Windows 和 GNU/Linux。由于它开放源码的特点, 因此 Windows 用户可以不依赖于 Visual Studio.NET, 编写跨平台的 C++ 程序。CodeBlocks 提供了许多工程模板, 支持语法醒目显示, 支持代码完成, 且支持工程管理, 以及项目构建、调试, 对于初学 C 语言和 C++ 语言的人来说, 是一个不错的运行工具。

1.2 计算机上运行程序的方法

下面以一个简单的程序来进行说明。当输入以下程序, 编译运行时, 计算机从屏幕输出 “Hello World”。

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     cout<<"Hello World"<<endl;
6     return 0;
7 }
```

1. 信息在计算机中的表示

信息在计算机中都是用 0 或 1 表示的。计算机通过这些位信息及上下文来解读这些

0 或 1。输入的 Hello World 程序就是由 0、1 组成的序列，将这些位信息每 8 位组织成 1 字节，每个字节用来表示一个文本字符。ASCII 码表给出了一种字符与数字的对应关系。Hello World 程序以字节方式存放于文件中，其每个字符对应一个数字，具体可参考 ASCII 码表。

2. 将程序翻译成机器可读的格式

因为输入的 Hello World 程序是人可读的，机器并不能直接识别它们，所以需要把这些文字翻译成机器可执行的二进制文件。这个工作是由编译系统完成的。编译系统由预处理器、编译器、汇编器、连接器 4 部分组成。以 Hello World 程序为例，编译系统的各部分共同将源文件编译成二进制可执行文件。各个部分完成的具体工作如下。

1) 预处理器：根据以“#”开头的命令，修改源程序。例如，根据 `#include<iostream>` 行，预处理器读取系统头文件 `iostream` 的内容，代替此行内容。源程序经过预处理后，得到另一个 C++ 程序。

2) 编译器：将预处理后的文件转换成汇编程序。编译器将不同的高级语言转换成严格一致的汇编语言格式进行输出。汇编语言以标准的文本格式确切描述每条机器语言指令。

3) 汇编器：将汇编程序翻译成机器语言指令，并将这些指令打包成一种可定位的目标程序格式。汇编后得到的文件即为二进制文件。

4) 连接器：Hello World 程序中调用 `cout` 对象，而这个对象必须以适当的方式并入程序中，这个工作由连接器完成。将外部所需的文件并入后，得到一个完整的 Hello World 可执行文件。可执行文件加载到存储器后，由系统负责执行。

1.3 程序设计的基本概念

程序设计中的基本概念介绍如下。

1) 程序：人们将需要计算机做的工作写成一定形式的指令，并把它们存储在计算机的内部存储器中，当人们给出命令后，计算机就按指令操作顺序自动进行。这种可以连续执行的一条条指令的集合就是人与机器进行对话的语言。

2) 高级语言：用接近人们习惯的自然语言和数学语言作为语言的表达形式，如 C、BASIC、C++、Java、Pascal 等。

3) 机器语言：由 0 和 1 构成的二进制指令或数据，贴近硬件。

4) 源程序：由高级语言编写的程序。

5) 目标程序：由二进制代码表示的程序。

6) 编译程序：能够把用户按照规定写出的语句一一翻译成二进制的机器指令，即具有翻译功能的程序。