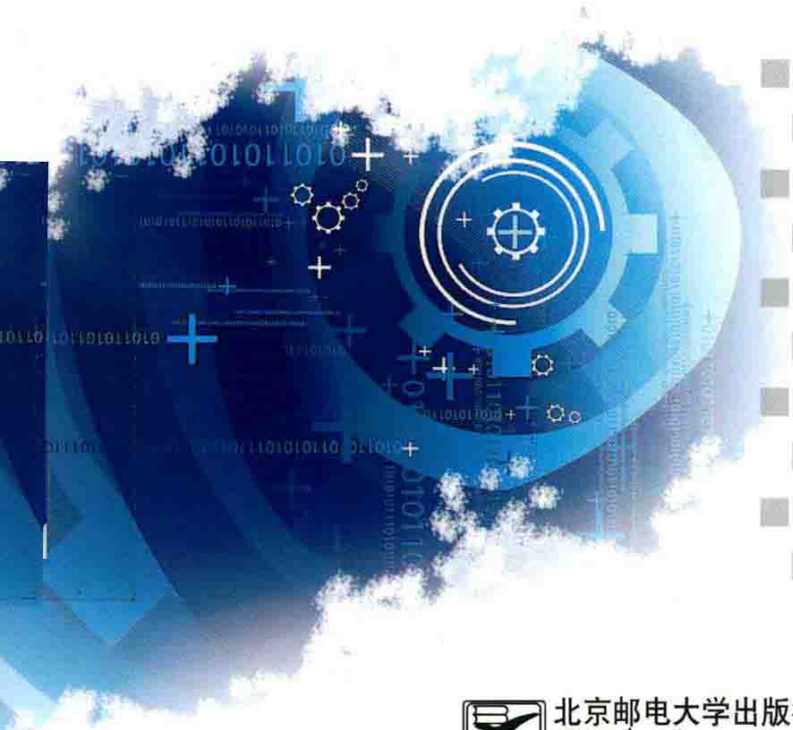
21世纪高等学校计算机科学与技术规划教材

C语言程序设计

C YUYAN CHENGXU SHEJI

主编　钱卫国　张玉生



北京邮电大学出版社
www.buptpress.com

21世纪高等学校计算机科学与技术规划教材

C语言程序设计

主　编　钱卫国　张玉生
副主编　孙　霞　刘　琰

北京邮电大学出版社
·北京·

内 容 简 介

C 语言是一种用途广泛、功能强大、使用灵活的高级语言，使用 C 语言既可以开发应用软件，又可以开发系统软件。目前，许多高等学校将 C 语言作为学生学习计算机程序设计的入门语言。

在编写过程中，作者结合多年程序设计课程的教学经验，在教学内容的安排上，突出了程序设计的方法与技巧，强调怎样用 C 语言解决一些实际问题，使读者对 C 语言的各种规定、语法在解决实际问题中逐步熟悉和掌握。对比较难掌握的内容（如函数、指针等内容）均分两章进行讨论，把难点进行分解，逐步深入，这种设计思路符合人类的认知规律。同时，作者力求做到体系结构合理、概念叙述准确、文字通俗易懂、例题选材合理。为方便教学和学生训练，作者同时编写了与本书配套的实验教材《C 语言程序设计习题与实验指导》一书。

本书适合作为高等学校非计算机专业程序设计课程的教材，也适合作为参加计算机等级考试的读者的学习用书。

图书在版编目(CIP)数据

C 语言程序设计/钱卫国，张玉生主编. -- 北京：北京邮电大学出版社，2012.12
ISBN 978-7-5635-3282-7

Ⅰ.①C… Ⅱ.①钱… ②张… Ⅲ.①C 语言—程序设计 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2012)第 269671 号

书　　名	C 语言程序设计
主　　编	钱卫国　张玉生
责任编辑	向　蕾
出版发行	北京邮电大学出版社
社　　址	北京市海淀区西土城路 10 号(100876)
电话传真	010-82333010　62282185(发行部)　010-82333009　62283578(传真)
网　　址	www.buptpress3.com
电子信箱	ctrd@buptpress.com
经　　销	各地新华书店
印　　刷	北京联兴华印刷厂
开　　本	787 mm×1 092 mm　1/16
印　　张	22.5
字　　数	557 千字
版　　次	2012 年 12 月第 1 版　2012 年 12 月第 1 次印刷

ISBN 978-7-5635-3282-7　　定价：39.00 元

前　言

C语言是一种用途广泛、功能强大、数据类型丰富、运算符多且使用灵活的结构化程序设计语言，它既具有高级语言程序设计的特点，又具有汇编语言的功能，所以使用C语言既可以开发应用软件，又可以开发系统软件。自20世纪90年代初C语言在我国推广应用以来，越来越多的人开始学习和使用C语言，目前绝大多数理工科院校都将C语言作为学生学习程序设计语言的入门语言。

参加本书编写的作者，均承担了多年高等学校非计算机专业的计算机程序设计课程的教学工作，积累了丰富的教学经验。在本书教学内容的安排上，针对非计算机专业的学生的特点，没有沿用传统书籍的安排方式，即先语法规定后例题验证的模式。首先映入读者眼帘的是怎样用C语言解决一些实际问题，始终围绕程序设计这一主线展开内容，着重于程序设计的基本方法与技巧，使读者对C语言的各种规定、语法在解决实际问题中逐步熟悉和掌握。对比较难掌握的内容（如函数、指针等内容）均分两章进行讨论，把难点进行分解，逐步深入，这种设计思路既便于教学，又符合多数人的认知规律，能够起到提高读者学习积极性和学习效率的作用。

全书内容分为13章。第1章为程序设计和C语言，主要对程序设计语言的发展、功能进行了介绍，同时对算法及C程序编辑、编译、连接、运行的方法进行了讨论。第2章为用C语言求解简单问题，介绍了使用if…else语句及for语句结合算术运算与关系运算求解简单分支与循环问题。第3章为分支结构程序设计，结合字符型数据及逻辑运算，介绍了多分支程序设计方法。第4章为循环结构程序设计，介绍了while循环、do…while循环及循环的嵌套，并对典型的算法（递推法、迭代法、穷举法）进行了讨论。第5章为函数初步，介绍了函数的定义与调用的方法，并对变量与函数的关系进行了讨论。第6章为数据类型和表达式，本章是对前几章所用到的数据类型和表达式的总结。第7章为数组，介绍了一维数组、二维数组、字符数组及字符串的使用方法，同时对数组作为函数的参数进行了讨论。第8章为指针初步，介绍了指针的基本概念、指针与函数、指针与一维数组、指针与字符串等内容，同时对用指针实现动态内存分配进行了讨论。第9章为结构与链表，介绍了结构的概念、结构变量的使用、结构数组、结构指针及链表。第10章为共用体与枚举类型，介绍了共用体与枚举类型的使用方法。第11章为再论函数，介绍了函数的组织方式、多模块程序设计、递归函数、宏定义、编译预处理。第12章为再论指针，介绍了指针与二维数组、指针数组与多重指针、指向函数的指针。第13章为文件，介绍了文件的分类、文件的基本操作等。

在编写过程中，作者力求做到体系结构合理、概念叙述准确、文字通俗易懂、例题选材合理。为使读者能很好地理解和掌握C语言程序设计的基本概念、编程方法与技巧，本书精选了大量例题，并给出分析与解答，每章末均配有适量的习题。为方便教学，作者同时编写了《C语言程序设计习题与实验指导》一书，作为实验教材与本书配套使用。作者为使用本书的

读者提供全书的程序代码、课件，可通过电子邮件联系索取。电子邮箱：zys6636@126.com。

本书由钱卫国、张玉生担任主编，由孙霞、刘炎担任副主编。参加本书编写的有张玉生（编写第 1、第 2、第 3 章）、钱卫国（编写第 4、第 5、第 7、第 11 章）、孙霞（编写第 6、第 9、第 10 章）、刘炎（编写第 8、第 12、第 13 章），此外贲黎明、施梅芳、刘春玉、周蕾、宗德才、何春霞、肖乐、盘丽娜、朱苗苗也参与了部分工作，在此表示感谢。全书由张玉生统稿。

本书适合作为高等学校非计算机专业程序设计课程的教材，也适合作为参加计算机等级考试的读者的学习用书。

由于写作时间仓促、作者水平有限，不妥之处在所难免，敬请读者批评指正。

编　者

目　录

第 1 章　程序设计和 C 语言 …… 1

1.1　引例 …… 1

1.2　程序与程序设计语言 …… 2

1.2.1　程序设计语言的发展 …… 2

1.2.2　程序设计语言的功能 …… 4

1.2.3　程序的算法表示 …… 5

1.2.4　程序设计语言的语法 …… 8

1.3　C 语言的发展历史与特点 …… 11

1.3.1　C 语言的发展历史 …… 11

1.3.2　C 语言的特点 …… 12

1.4　C 程序的编辑、编译和运行 …… 12

1.4.1　程序设计的任务 …… 12

1.4.2　运行 C 程序的步骤与方法 …… 13

习题 1 …… 14

第 2 章　用 C 语言求解简单问题 …… 16

2.1　引例 …… 16

2.2　常量、变量和数据类型 …… 18

2.3　算术运算和赋值运算 …… 19

2.3.1　算术运算 …… 19

2.3.2　赋值运算 …… 19

2.4　数据输入与输出 …… 21

2.4.1　格式输出函数 printf() …… 21

2.4.2　格式输入函数 scanf() …… 21

2.5　求解简单分支问题 …… 22

2.5.1　引例 …… 23

2.5.2　关系运算 …… 23

2.5.3　if…else 语句 …… 24

2.5.4　数学库函数 …… 26

2.6　求解简单循环问题 …… 27

2.6.1　引例 …… 27

2.6.2　for 循环语句 …… 28

2.7 简单程序设计举例 …… 32
习题 2 …… 34
第 3 章 分支结构程序设计 …… 37
3.1 引例 …… 37
3.2 字符类型 …… 38
3.2.1 字符常量 …… 38
3.2.2 字符变量 …… 38
3.3 字符类型数据的输入与输出 …… 39
3.3.1 字符输入函数 getchar() …… 39
3.3.2 字符输出函数 putchar() …… 39
3.3.3 调用函数 scanf()和 printf()输入输出字符 …… 40
3.4 逻辑运算 …… 41
3.5 分支结构程序设计 …… 43
3.5.1 if 语句的嵌套 …… 43
3.5.2 多分支 if 语句 …… 47
3.5.3 多分支 switch 语句 …… 49
3.6 分支结构程序设计举例 …… 52
习题 3 …… 54
第 4 章 循环结构程序设计 …… 57
4.1 引例 …… 57
4.2 while 语句 …… 58
4.3 do…while 语句 …… 61
4.4 goto 语句 …… 63
4.5 break 语句和 continue 语句 …… 64
4.5.1 break 语句 …… 64
4.5.2 continue 语句 …… 66
4.6 循环嵌套 …… 67
4.7 典型算法举例 …… 70
4.7.1 递推法 …… 70
4.7.2 迭代法 …… 72
4.7.3 穷举法 …… 73
习题 4 …… 75
第 5 章 函数初步 …… 77
5.1 引例 …… 77
5.2 函数概述 …… 78
5.3 函数的定义 …… 79
5.3.1 无参函数的定义 …… 79
5.3.2 有参函数的定义 …… 80

5.3.3 空函数的定义 …… 82
5.4 函数的调用 …… 83
5.4.1 函数调用的形式和过程 …… 83
5.4.2 参数传递 …… 85
5.4.3 函数的返回值 …… 87
5.4.4 函数声明 …… 88
5.5 变量与函数 …… 89
5.5.1 局部变量和全局变量 …… 89
5.5.2 变量生命周期与存储类型 …… 93
5.6 函数程序设计举例 …… 96
习题5 …… 98
第6章 数据类型和表达式 …… 101
6.1 引例 …… 101
6.2 数据的存储和基本数据类型 …… 102
6.2.1 数据的存储 …… 102
6.2.2 基本数据类型 …… 103
6.3 常量和变量 …… 107
6.3.1 标识符 …… 107
6.3.2 常量 …… 108
6.3.3 变量 …… 112
6.4 数据的输入和输出 …… 114
6.4.1 输入输出函数 …… 114
6.4.2 整型数据的输入和输出 …… 115
6.4.3 实型数据的输入和输出 …… 119
6.4.4 字符型数据的输入和输出 …… 120
6.5 类型转换 …… 121
6.5.1 自动类型转换 …… 121
6.5.2 赋值类型转换 …… 122
6.5.3 强制类型转换 …… 124
6.6 运算符与表达式 …… 124
6.6.1 算术表达式 …… 124
6.6.2 赋值表达式 …… 128
6.6.3 关系表达式 …… 129
6.6.4 逻辑表达式 …… 130
6.6.5 条件表达式 …… 132
6.6.6 逗号表达式 …… 133
6.6.7 位运算 …… 133
6.6.8 其他运算符 …… 136

6.7 运算符的优先级与结合性 …… 137
6.8 表达式程序设计举例 …… 138
习题 6 …… 140
第 7 章 数组 …… 144
7.1 引例 …… 144
7.2 一维数组 …… 145
7.2.1 一维数组的定义和引用 …… 145
7.2.2 一维数组的初始化 …… 147
7.2.3 一维数组程序设计举例 …… 149
7.3 二维数组 …… 154
7.3.1 二维数组的定义和引用 …… 155
7.3.2 二维数组的初始化 …… 157
7.3.3 二维数组程序设计举例 …… 159
7.4 字符数组 …… 162
7.4.1 字符数组的定义和引用 …… 162
7.4.2 字符数组的初始化 …… 163
7.4.3 字符数组程序设计举例 …… 164
7.5 字符串 …… 165
7.5.1 字符串的存储 …… 165
7.5.2 字符串的输入与输出 …… 166
7.5.3 字符串的操作 …… 170
7.5.4 字符串程序设计举例 …… 171
7.6 数组作为函数的参数 …… 173
习题 7 …… 177
第 8 章 指针初步 …… 180
8.1 引例 …… 180
8.2 指针与指针变量 …… 181
8.2.1 地址和指针 …… 181
8.2.2 指针变量的定义与初始化 …… 182
8.2.3 指针变量的引用 …… 185
8.3 指针与函数 …… 187
8.3.1 指针作为函数的参数 …… 187
8.3.2 返回指针值的函数 …… 191
8.3.3 指针作为函数参数的程序设计 …… 191
8.4 指针与一维数组 …… 192
8.4.1 指针与一维数组的关系 …… 192
8.4.2 指针的运算 …… 195
8.4.3 指针替代数组的程序设计 …… 197

8.5　指针与字符串 …… 200
8.5.1　字符串与字符指针 …… 200
8.5.2　使用字符数组和字符指针处理字符串的区别 …… 202
8.5.3　常用的字符串处理函数 …… 203
8.6　用指针实现动态内存分配 …… 208
8.6.1　动态内存分配函数 malloc()和 calloc() …… 208
8.6.2　动态内存释放函数 free() …… 210
8.6.3　动态内存分配调整函数 realloc() …… 210
8.6.4　动态内存分配程序设计 …… 211
习题 8 …… 212
第 9 章　结构与链表 …… 215
9.1　引例 …… 215
9.2　结构的概念与定义 …… 217
9.3　结构变量 …… 219
9.3.1　结构变量的定义与初始化 …… 219
9.3.2　结构变量的使用 …… 222
9.4　结构数组 …… 225
9.4.1　结构数组的定义与初始化 …… 225
9.4.2　结构数组的使用 …… 228
9.5　结构指针 …… 231
9.5.1　结构指针的概念 …… 231
9.5.2　指向结构数组的指针 …… 233
9.5.3　结构指针作为函数参数 …… 235
9.6　链表 …… 237
9.6.1　链表的概念 …… 238
9.6.2　链表的基本操作 …… 239
习题 9 …… 247
第 10 章　共用体与枚举类型 …… 251
10.1　共用体 …… 251
10.1.1　引例 …… 251
10.1.2　共用体类型与共用体变量 …… 253
10.1.3　共用体变量的初始化 …… 256
10.1.4　共用体变量的使用 …… 256
10.2　枚举类型 …… 261
10.2.1　引例 …… 261
10.2.2　枚举类型的定义 …… 262
10.2.3　枚举类型的使用 …… 263
10.3　类型定义 …… 265

习题 10 …… 270
第 11 章 再论函数 …… 273
11.1 函数的组织 …… 273
11.1.1 引例 …… 273
11.1.2 函数的嵌套调用 …… 275
11.1.3 全局变量的存储类型 …… 277
11.1.4 内部函数与外部函数 …… 279
11.2 多模块程序设计 …… 281
11.2.1 使用工程文件 …… 281
11.2.2 使用文件包含 …… 283
11.3 递归函数 …… 285
11.3.1 引例 …… 285
11.3.2 递归函数的概念 …… 286
11.3.3 递归函数程序设计 …… 288
11.4 宏定义 …… 289
11.4.1 无参宏定义 …… 289
11.4.2 带参宏定义 …… 292
11.5 编译预处理 …… 295
习题 11 …… 297
第 12 章 再论指针 …… 302
12.1 指针与二维数组 …… 302
12.1.1 指针与二维数组的关系 …… 302
12.1.2 指向二维数组元素的指针 …… 305
12.1.3 行指针变量(数组指针) …… 306
12.2 指针数组与多重指针 …… 307
12.2.1 指针数组的概念 …… 307
*12.2.2 指针数组与命令行参数 …… 309
*12.2.3 指向指针的指针 …… 310
12.2.4 使用指针数组处理字符串 …… 312
*12.3 指向函数的指针 …… 313
12.4 指针程序设计举例 …… 315
习题 12 …… 317
第 13 章 文件 …… 319
13.1 引例 …… 319
13.2 数据文件概述 …… 320
13.2.1 ASCII 文件与二进制文件 …… 320
13.2.2 缓冲文件系统 …… 321
13.2.3 文件结构与文件类型指针 …… 322

13.2.4　标准输入输出设备文件 …… 322
13.3　文件的打开与关闭 …… 323
13.3.1　文件的打开 …… 323
13.3.2　文件的关闭 …… 324
13.4　文件的操作 …… 325
13.4.1　字符方式文件读写函数 fputc()和 fgetc() …… 325
13.4.2　字符串方式文件读写函数 fputs()和 fgets() …… 327
13.4.3　格式化文件读写函数 fprintf()和 fscanf() …… 328
13.4.4　数据块读写函数 fwrite()和 fread() …… 330
13.5　其他文件操作函数 …… 332
13.6　文件程序设计举例 …… 335
习题 13 …… 337
附录 A　ASCII 码字符集 …… 339
附录 B　运算符的优先级和结合性 …… 340
附录 C　C 语言常用标准库函数 …… 342
参考文献 …… 348

第1章　程序设计和C语言

C语言是一种用途广泛、功能强大、使用灵活的高级语言，使用C语言既可以开发应用软件，又可以开发系统软件。自20世纪90年代初C语言在我国推广应用以来，越来越多的人开始学习和使用C语言，目前几乎绝大多数理工科院校都开设有C语言程序设计课程。

本章首先由一个例子出发，使读者对C语言程序有一个感性认识，然后简要介绍程序与程序设计语言、C语言的发展历史与特点，以及C程序的编辑、编译和运行的步骤。

1.1　引　　例

【例1.1】 编写程序，输出两个整型数中的较大值。

程序代码：

```
#include <stdio.h>
int main()
{ int a,b,c;                                    /* 定义3个整型变量 */
  int max(int,int);                             /* 函数声明 */
  printf("Input an integer:");                  /* 显示提示信息 */
  scanf("%d",&a);                               /* 从键盘获得一个整数,赋给a */
  printf("Input an other integer:");            /* 显示提示信息 */
  scanf("%d",&b);                               /* 从键盘获得一个整数,赋给b */
  c=max(a,b);                                   /* 调用函数,求a与b的最大值,赋给c */
  printf("The max of %d and %d is:%d\n",a,b,c); /* 输出最大值 */
  return 0;
}
int max(int x,int y)                            /* 求最大值的函数 */
{ int z;                                        /* 定义临时变量z */
  if(x>y)
    z=x;                                        /* x>y时,z的值等于x */
  else
    z=y;                                        /* x<y时,z的值等于y */
  return z;                                     /* 结束函数,返回z */
}
```

运行结果：

```
Input an integer:12
Input an other integer:23
The max of 12 and 23 is:23
```

说明：

①本程序并不要求读者能完全理解，相关内容会在后续章节详细介绍，这里只要有个初步印象即可。

②由上面程序可以看出，C 程序是由函数组成的。本程序涉及 4 个函数：main()，max()，scanf()和 printf()。其中，main()函数是程序的主函数；max()是程序中定义的函数，其作用是求两个数中的较大数；scanf()和 printf()是系统事先设计好的函数，分别用于输入和输出。

③所有的 C 程序有且只有一个 main()函数。C 程序总是从 main()函数的第 1 条语句开始运行的，当 main()函数结束时，程序也就结束了。

④本程序执行时，首先从调用 printf()函数开始，输出第 1 个提示，然后调用 scanf()函数从键盘获得一个值，并赋给变量 a；接着调用第 2 个 printf()函数输出第 2 个提示，再调用第 2 个 scanf()函数从键盘获得一个值，并赋给变量 b；求两个数的较大值是由 max()函数完成的。

⑤程序中包含了数据表达与数据处理(流程控制)两部分。在 main()函数中数据表达使用语句“int a,b,c;”，该语句定义了 3 个整型变量，变量 a 与 b 分别存放输入的整数，变量 c 用于存放函数调用的结果，即 a 与 b 的较大值；在 max()函数中数据表达使用语句“int z;”，z 是一个临时工作变量，它只能在 max()函数中使用，其作用是存放形式参数 x 与 y 中的较大值，函数结束时返回给 main()函数的调用处。在 max()函数中数据处理使用 if 语句来实现。

1.2 程序与程序设计语言

程序(program)是为实现特定目标或解决特定问题而用程序设计语言(计算机语言)编写的一系列语句和指令，计算机能严格按照这些指令去做。程序的执行过程实际上是对程序所表达的数据进行处理的过程。一方面，程序设计语言提供了一种数据表达与数据处理的功能；另一方面，编程人员必须按照程序设计语言的语法要求进行编程。

1.2.1 程序设计语言的发展

自 1946 年世界上第一台电子计算机问世以来，计算机应用已经渗透到人们生活的方方面面，极大地推动了社会的进步与发展。特别是因特网(Internet)的发展，从根本上改变了人们的生活方式，人们已经难以摆脱对计算机的依赖。几十年来计算机硬件技术在不断地飞速发展着，同时软件技术也没有停止前进的步伐。用来开发软件的程序设计语言经过多年的发展，其技术和方法日臻成熟。程序设计语言的发展经历了以下几个阶段：

1. 机器语言

机器语言属于第一代程序设计语言。按照冯·诺依曼原理，计算机内部运算采用的是二进制，也就是说计算机只能识别和接收由 0 和 1 组成的指令代码，这种计算机能直接识别和接收的二进制代码称为机器指令(machine instruction)。机器指令的集合(即指令系统)就是该计算机的机器语言。用机器语言编写的程序称为目标程序(object program)，目标程序可以被计算机直接执行，且运行效率是最高的。但由于不同类型的计算机的指令系统存在差异，因而在一种类型的计算机上编写的机器语言程序，在另一种不同类型的计算机上有可能不能运行。

显然，机器语言与人们习惯用的语言差别太大，由于其难学、难写、难记、难修改，采用机器语言编程只是极少数人能够完成的工作。

2. 汇编语言

为了减轻使用机器语言编程的困难，人们采用助记符来代替机器指令的二进制串，如用 ADD 表示加法，SUB 表示减法，MOV 表示传送数据等。这样就能使运算指令使用符号而不再使用二进制表示。采用这种方法编写的程序，容易被人读懂，程序的修改与维护也很方便，这种程序设计语言就是汇编语言，也称为第二代程序设计语言。然而计算机并不认识这些助记符，这就需要一个专门的程序，负责将这些助记符翻译成二进制的机器指令，这种翻译程序被称为汇编程序。

由于机器语言与汇编语言均很“接近”计算机，人们常常称它们为“低级语言”。

3. 高级语言

为了克服低级语言的缺陷，人们在实践中逐渐认识到，应该设计一种这样的语言：接近于数学语言或人类的自然语言（英语），同时又不依赖于计算机硬件，编制的程序能在所有的机器上通用。经过不懈努力，1954 年，第一个完全脱离机器硬件的高级语言——FORTRAN（FORmula TRANslator，公式翻译器）语言问世了。

这种语言功能很强，且不依赖于具体机器，用它编写的程序几乎可以在任何型号的机器上运行，人们把这种语言称为“高级语言”。高级语言的特点是易学、易用、易维护，人们可以更有效、更方便地用它来编制各种用途的计算机程序。

当然，高级语言编写的程序也不能被计算机直接运行，同样需要经过“翻译”。这里我们将高级语言编写的程序称为源程序（source program），将源程序翻译成目标程序的程序称为编译程序。

自从第一个高级语言问世，几十年来先后出现的高级语言有上千种，比较有影响的有 FORTRAN，ALGOL，COBOL，BASIC，QBASIC，Pascal，LISP，C，C＋＋，Delphi，Visual Basic，JAVA，C＃等。

按照语言的特性，高级语言又经历了不同的发展阶段。

（1）非结构化的语言

人们在使用早期的高级语言编程时，编程风格比较随意，没有编程规范可以遵循，程序中的流程可以随意跳转，程序员往往只追求程序的执行效率，而不顾及程序的结构，使程序变得难以阅读和维护。早期的 FORTRAN、ALGOL 和 BASIC 都属于非结构化的语言。

非结构化编程的后果是严重的，到了 20 世纪 60 年代中后期，开发的软件越来越多，规模越来越大，但软件的正确性却越来越难以保证，直接导致许多耗费巨资开发的软件由于含有错误而不能使用，计算机界出现了软件危机。

（2）结构化的语言

为了解决非结构化的语言所带来的问题，1969 年提出了结构化程序设计方法，1970 年，第一个结构化程序设计语言——Pascal 语言出现，标志着结构化程序设计时期的开始。结构化程序设计方法规定：程序必须由具有良好特性的基本结构（顺序结构、分支结构、循环结构）构成，程序中的流程不允许随意跳转，程序总是由上而下顺序执行各个基本结构。采用结构化的语言所编制的程序结构清晰，易于阅读和维护。QBASIC、Pascal 和 C 都属于结构化的语言。

(3)面向对象的语言

自 20 世纪 80 年代开始,提出了面向对象(object oriented)的程序设计思想。相比而言,之前的高级语言可以称之为面向过程的程序设计语言,程序的执行是流水式的,即在一个模块被执行完成前,不能去执行另一个模块,程序员不能随意地去改变程序的执行流程。除此之外,程序中不仅需要实现每一个过程的细节,而且程序不易重复使用。这些都与人们日常处理事务的方式不一致,人们所希望的是在对象(object)的每一个事件发生时都能得到及时的处理。C++,C#,Visual Basic 和 JAVA 等语言是支持面向对象程序设计方法的语言。

1.2.2 程序设计语言的功能

程序设计语言是人与计算机进行交流的桥梁,人要让计算机按照自己的意愿处理数据,就必须用程序设计语言表达所要处理的数据及控制数据处理的流程。因此,程序设计语言必须具有数据表达与数据处理的能力。

1. 数据表达

数据是计算机处理的对象,在解决实际问题时,通常包含着各种类型的数据,数据类型(data type)就是对某些具有相同性质的数据集的总称。例如,整数、实数均是最基本的数据类型。数据类型有两个含义:该数据类型能表示些什么样的数据(即取值范围)? 能对这些数据进行何种操作(即运算类型)? 例如,在 Turbo C 中整数类型的取值范围为−32 768～32 767,而+、−、*、/、%是作用在整数类型上的运算。

通常每种程序设计语言中,都事先定义了几种基本的数据类型,用来定义程序中用到的数据,如整型、实型、字符型等。这些基本的数据类型可以定义的数据对象表现为两种形式:常量(又称常数)和变量。常量的值在程序中是不改变的。例如,在 C 语言中,2 是一个整型常量,2.0 是一个实型常量,'2'是一个字符型常量。变量的值则可以随时改变。例如,在 C 语言中,使用 int a 定义了一个整型变量 a,则 a=10 表示给变量 a 赋值 10。

在许多程序设计语言中,基本的数据类型可以用于构造复杂的数据类型,以表达实际问题中存在的复杂的数据结构。例如,可以用基本的数据类型构造出数组(array)、结构(structure)、指针(pointer)等复杂类型。在 C 语言中,int a[10]是用整型构造出一个包含 10 个元素的数组,a 为数组名,数组中的 10 个元素均为一个整型变量。

2. 数据处理

程序设计语言除了具备良好的数据表达能力之外,还必须提供对不同类型的数据进行处理的手段。对数据的处理是通过语言的一系列流程控制语句实现的。

在 1.2.1 小节中已经介绍,按照结构化程序设计的观点,任何程序的基本结构都可以通过 3 种基本的控制结构(顺序结构、分支结构和循环结构)进行组合来实现。

● 顺序结构:一条语句执行完后,按自然顺序执行下一条语句。例如,C 语言中的赋值语句、输入、输出语句等都构成了顺序结构。

● 分支结构:又称选择结构。计算机在执行程序时,常常需要根据不同的条件选择执行不同的语句。例如,C 语言中的 if 与 switch 语句都可以构成分支结构。

● 循环结构:许多时候,计算机需要重复执行相同的语句。重复执行一般是有条件的,在条件满足时,重复执行;在条件不满足时,则不会重复执行。例如,C 语言中的 for,while 与

do…while 语句都可以构成循环结构。

这 3 种基本结构的共同特点是：

①只有单一的入口和单一的出口。

②结构中的每个部分都有被执行的可能。

③结构内不应出现永不终止的死循环。

当所要求解的问题复杂时，所编写的应用程序经常由上万条语句组成，需要由多人来完成。这时，常常要将一个大任务分解为若干个子任务，每个子任务又分成很多个小子任务，每个小子任务只完成一个简单的功能。在程序设计时，用一个个小模块来实现这些小子任务。而每一个小模块基本上均由上述的 3 种基本结构组合而成。这种程序设计方法就是结构化程序设计(structured programming)。C 语言就是典型的结构化程序设计语言。

1.2.3　程序的算法表示

人们常说，软件的主体是程序，程序的核心是算法。这是因为要使计算机解决某个问题，首先必须针对该问题设计一个解题步骤，然后再据此编写出程序并交给计算机执行。这里所说的解题步骤就是“算法”，采用程序设计语言对问题的对象和解题步骤进行的描述就是程序。瑞士计算机科学家尼·沃思(N. Wirth)有一句名言：“计算机科学就是研究算法的学问”。由此可以看出算法在程序设计中的重要性。

通俗地讲，算法就是解决问题的方法与步骤。例如，要交换两个变量 a 和 b 的值，按照下面的步骤就可以完成交换：

①输入变量 a 和 b 的值。

②将变量 a 的值赋给变量 t。

③将变量 b 的值赋给变量 a。

④将变量 t 的值赋给变量 b。

⑤输出变量 a 与 b 的值。

尽管针对不同问题所设计的算法千变万化，简繁各异，但作为算法，都应具备如下几个特征：

● 确定性：算法的每条指令必须有明确的含义，不能有二义性。对于相同的输入必须得出相同的执行结果。

● 有穷性：一个算法应包含有限个操作步骤。也就是说，在执行若干个操作步骤之后，算法将结束，而且每一步都在合理的时间内完成。

● 可行性：算法中指定的操作都可以通过已经实现的基本运算执行有限次后实现。

● 有零个或多个输入：算法是用来处理数据对象的，在大多数情况下，这些数据对象需要通过输入来得到。

● 有一个或多个输出：算法的目的是为了求“解”，“解”只有通过输出才能得到。

算法的表示可以有多种形式，如文字表示、流程图表示、伪代码和程序设计语言表示等。下面对算法的表示方法进行简单叙述。

1. 用文字描述算法

下面是用文字描述的几个算法：

【算法 1】交换两个变量 a 和 b 的值。算法步骤的文字描述见本节开始。