

PUTONG GAO DENG
YUANXIAO
YINGYONGXING
BENKE GUIHUA JIAOCAI

普通高等院校应用型本科规划教材

C 语言 程序设计基础 (第二版)

主 编 熊 壮

副主编 王 宁 霍敏霞

C YUYAN
CHENGXU SHEJI JICHU



重庆大学出版社

<http://www.cqup.com.cn>

内容提要

本书针对程序设计语言的初学者,以 C 语言为载体,以微软 Visual C++6.0 为环境,通过讨论 C 程序设计的一般过程和方法,重点介绍了结构化程序设计的基本思想和实现方法。本书坚持“理论够用,程序设计实际能力为重”的宗旨,通过数据组织、控制结构、文件处理等程序设计基础知识的讨论,向读者介绍使用 C 语言进行程序设计的基本方法。通过对指针与函数关系、指针与数组关系、指针数组、动态数组实现方法、构造数据类型使用方法等方面的讨论,向读者介绍 C 语言特有的一些重要知识,使读者能够循序渐进地掌握使用 C 语言开发各类常见应用程序的基本技能。

本书在附录中还提供了 ASCII 码表、C 程序设计中常用的标准库函数、使用 Visual C++6.0 集成环境开发 C 程序的基本方法等重要学习资料。

本书覆盖了 C 语言的应用基础,内容深入浅出,语言流畅,例题丰富,适合作为第一门程序设计语言课程教材,对于程序设计爱好者也是较好的入门教材或参考书。

图书在版编目(CIP)数据

C 语言程序设计基础/熊壮主编.—2 版.—重庆:

重庆大学出版社,2016.8

普通高等院校应用型本科计算机专业系列教材

ISBN 978-7-5689-0060-7

I. ①C… II. ①熊… III. ①C 语言—程序设计—高等学校—教材 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2016)第 196082 号

C 语言程序设计基础

(第二版)

主 编 熊 壮

责任编辑:章 可 版式设计:章 可

责任校对:秦巴达 责任印制:张 策

*

重庆大学出版社出版发行

出版人:易树平

社址:重庆市沙坪坝区大学城西路 21 号

邮编:401331

电话:(023) 88617190 88617185(中小学)

传真:(023) 88617186 88617166

网址:<http://www.cqup.com.cn>

邮箱:fxk@cqup.com.cn (营销中心)

全国新华书店经销

重庆升光电力印务有限公司印刷

*

开本:787 mm×1092 mm 1/16 印张:19.5 字数:451 千

2015 年 8 月第 1 版 2016 年 8 月第 2 版 2016 年 8 月第 4 次印刷

ISBN 978-7-5689-0060-7 定价:39.00 元

本书如有印刷、装订等质量问题,本社负责调换

版权所有,请勿擅自翻印和用本书

制作各类出版物及配套用书,违者必究

计算机程序设计能力是理工科大学生应具有的基本素养之一。培养学生的逻辑思维能力、抽象能力和基本的计算机程序设计能力,引导学生进入计算机程序设计的广阔空间是大学计算机程序设计课程的主要目标。

计算机程序设计的主要任务就是用合适的计算机程序设计语言对解决问题的方法进行编码处理,即编制程序。C 语言功能丰富、表达能力强、程序执行效率高、可移植性好;C 语言既有高级计算机程序设计语言的特点,同时又具有部分汇编语言的特点,因而 C 语言具有较强的系统处理能力;C 语言是一种结构化程序设计语言,支持自顶向下、逐步求精的程序设计技术,通过 C 语言函数结构可以方便地实现程序的模块化;在 C 语言的基础上发展起来的面向对象程序设计语言如 C++、Java、C#等与 C 语言有许多的共同特征,掌握 C 语言对学习进而应用这些面向对象的程序设计语言有极大的帮助。

本书编撰的宗旨是“理论够用,程序设计实际能力为重”,从结构化程序设计技术出发,以 C 程序设计语言为载体,介绍了 C 语言的基本语法、语义并对学习 C 语言过程中可能遇到的各种常见典型问题进行了分析,通过对程序设计技术基础范畴内各种典型问题的求解方法的描述以及相应 C 语言代码的描述,展现了在程序设计过程中如何对问题进行分析,如何组织数据和如何描述解决问题的方法,展现了在计算机应用过程中如何将方法和编码相联系的具体程序设计过程,进而向读者介绍了计算机结构化程序设计的基本概念、基本技术和方法。

本书的主要内容分为相辅相成的两个部分,第一部分包括第 1 章—第 5 章,主要介绍计算机程序设计高级语言共性的基础知识,包含的主要内容有:基本数据类型的使用,运算符和表达式计算基础,标准库函数的使用方法和顺序程序设计,结构化程序设计,函数调用中的数值参数和地址值参数传递,变量的作用域和生存期,数组使用和字符串处理基础,函数调用中的数组参数传递,文件数据处理基础。对于需要了解和掌握结构化程序设计基本思想的读者,通过这部分的学习可较为全面地掌握结构化程序设计的基本思想。第二部分包括第 6 章—第 9 章,主要介绍 C 语言特有的一些重要知识,包含的主要内容有:返回指针值的函数,指向函数的指针以及指向函数指针变量作函数的形式参数,数组与指针的关系,指针数组,命令行参数,用指针实现动态数组的方法,结构体数据类型,联合体数据类型,用 typedef 关键字描述复杂数据类型,编译预处理基础,位运算与枚举类型。通过第二部分内容的学习,读者可以较为全面地掌握 C 语言的基础知识。

本书选用 Microsoft Visual C++ 6.0 作为教学环境,书中的所有教学示例都在 Microsoft Visual C++ 6.0 集成开发环境中通过。附录中还提供了 ASCII 码表、C 程序设计中常用的标准库函数、使用 Visual C++ 6.0 IDE 开发 C 程序的基本方法等重要学习资料。

本书适于高等院校本、专科各专业作为程序设计语言类课程教材,同时可作为计算机专业本、专科学生、计算机应用开发人员、程序设计爱好者、计算机等级考试应试者在学习程序设计语言和程序设计技术时的参考资料。

参加本书编撰的作者都是长期从事程序设计课程教学的一线教师,在教材内容选取、章节教学顺序安排上都尽可能考虑了 C 语言基础内容与初学程序设计难度上的平衡,通过丰富的例题、流畅的语言对教学内容进行了深入浅出的描述。参加本书编撰的教师有:邓秋菊、霍敏霞、王丽艳、王宁、王伟锋、熊壮、张欣、周桥、钟怡。各章节编写分工如下:邓秋菊(第 4 章),霍敏霞(第 7 章、第 8 章),王丽艳(第 1 章),王宁(第 5 章、第 9 章),王伟锋(第 6 章),周桥(第 3 章),钟怡(第 2 章),张欣参与了第 5 章和第 8 章的编撰工作。全书由熊壮进行内容调整、修改,统一定稿。

限于编者水平,书中的错误和不妥之处在所难免,恳请读者不吝指教。

联系地址:重庆邮电大学移通学院计算机科学系。

E-Mail:xiongzc@cqu.edu.cn

编 者
2016 年 6 月

第 1 章 C 程序设计初步	1
1.1 C 程序结构和处理过程	1
1.1.1 C 程序的基本结构	1
1.1.2 C 程序的处理过程	4
1.2 C 语言的基本数据类型	5
1.2.1 C 程序中数据的表示	5
1.2.2 C 语言基本数据类型	6
1.2.3 C 程序中数据的输入输出	12
1.3 C 语言基本运算符和表达式的运算	19
1.3.1 C 运算符和表达式的概念	19
1.3.2 赋值运算符	21
1.3.3 算术运算符	22
1.3.4 自增自减运算符	23
1.3.5 复合赋值运算符	24
1.3.6 逗号运算符	24
1.3.7 sizeof 运算符	25
1.3.8 数据类型转换	26
1.4 C 语言标准库	27
1.4.1 C 标准库的使用方法	27
1.4.2 常用数学标准库函数介绍	28
习题 1	34
 第 2 章 C 程序的基本控制结构	 40
2.1 C 语言关系运算和逻辑运算	40
2.1.1 关系运算符	40
2.1.2 逻辑运算符	40
2.2 分支结构程序设计	42
2.2.1 单分支程序设计	42

2.2.2	复合语句在程序中的使用	43
2.2.3	双分支程序设计	45
2.2.4	多分支程序设计	46
2.3	循环结构程序设计	51
2.3.1	while 循环控制结构	51
2.3.2	do...while 循环控制结构	52
2.3.3	for 循环控制结构	54
2.3.4	空语句及其在程序中的使用	55
2.3.5	循环的嵌套结构	56
2.3.6	break 语句和 continue 语句	57
2.4	基本控制结构简单应用	60
2.4.1	穷举方法程序设计	60
2.4.2	迭代方法程序设计	62
2.4.3	一元高阶方程的迭代程序解法(*)	65
习题 2		68
第 3 章 函数		75
3.1	函数的定义和调用	75
3.1.1	函数的定义和声明	75
3.1.2	函数调用中的数值参数传递	79
3.2	函数调用中的指针参数传递	82
3.2.1	指针变量的定义和引用	82
3.2.2	函数调用中的地址值参数传递	85
3.3	函数的嵌套调用和递归调用	89
3.3.1	函数的嵌套调用	89
3.3.2	函数的递归调用	91
3.4	变量的作用域和生存期	95
3.4.1	变量的作用域	96
3.4.2	变量的生存期	100
习题 3		105
第 4 章 数组和字符串		112
4.1	数组的定义及数组元素的引用	112
4.1.1	一维数组的定义和元素引用方法	112
4.1.2	二维数组和多维数组	117
4.2	字符数组和字符串	122

4.2.1	字符数组的定义和初始化	122
4.2.2	字符数组的输入输出	123
4.2.3	常用字符类数据处理标准库函数	127
4.3	函数调用中的数组参数传递	135
4.3.1	一维数组作函数的参数	136
4.3.2	二维数组作函数的参数	139
4.4	数组的简单应用	141
4.4.1	数组元素值的随机生成	141
4.4.2	基于数组的常用排序方法	143
4.4.3	基于数组的常用查找方法	145
习题 4		149
第 5 章	C 程序文件处理基础	155
5.1	顺序存取文件处理基础	155
5.1.1	C 语言的文件数据类型	155
5.1.2	文件的打开/创建和关闭	157
5.1.3	文件内部读写位置指针和文件尾的检测方法	159
5.2	文件处理中数据的读/写方法	159
5.2.1	单个字符数据的读写	160
5.2.2	字符串数据的读写	164
5.2.3	格式化数据的读写	165
5.2.4	数据块的读写	169
5.3	随机存取文件处理基础(*)	172
5.3.1	随机存取文件处理的基本概念	172
5.3.2	重置文件内部记录指针	172
5.3.3	设置文件内部读写位置指针	174
5.3.4	获取文件内部读写位置指针的当前位置	177
5.3.5	文件读写操作模式的使用方法	178
习题 5		181
第 6 章	指针	188
6.1	指针与函数	188
6.1.1	返回指针值的函数	188
6.1.2	指向函数的指针变量	191
6.2	指针与一维数组	195
6.2.1	指向一维数组元素的指针变量	195

6.2.2 指向一维数组的指针变量	196
6.3 指针与二维数组(*)	200
6.3.1 多级指针的定义和引用	200
6.3.2 指向二维数组元素的指针变量	201
6.3.3 指向二维数组的指针变量	203
6.4 指针数组与命令行参数	207
6.4.1 指针数组的定义和引用	207
6.4.2 命令行参数(*)	210
6.5 使用指针构建动态数组	212
6.5.1 动态数据的概念和存储分配标准库函数	212
6.5.2 一维动态数组的建立和使用	214
6.6 指针与字符串(*)	216
6.6.1 字符串的指针表示	216
6.6.2 字符串处理标准函数的指针参数	218
习题 6	227
第 7 章 编译预处理基础	234
7.1 宏定义预处理命令及其简单应用	234
7.1.1 不带参数的宏定义	234
7.1.2 带参数的宏定义	236
7.2 文件包含预处理命令及其简单应用	237
7.2.1 文件包含的书写形式及意义	237
7.2.2 用文件包含方式组织多源文件 C 程序	238
7.3 条件编译预处理命令及其简单应用	240
7.3.1 #if、#elif、#else、#endif	240
7.3.2 #ifdef 和 #ifndef	241
习题 7	243
第 8 章 结构体和联合体	248
8.1 结构体类型的定义和使用	248
8.1.1 结构体类型和结构体变量的定义	248
8.1.2 typedef 关键字的简单应用	251
8.1.3 结构体变量的使用方法	253
8.2 结构体数组	257
8.2.1 结构体数组的定义和数组元素引用	257
8.2.2 结构体数组作函数的参数	259

8.3 结构体数据类型与指针的关系	261
8.3.1 结构体类型变量与指针的关系	261
8.3.2 结构体类型数组与指针的关系	263
8.3.3 结构体类型的简单应用——单链表基本操作(*)	265
8.4 联合体数据类型	271
8.4.1 联合体数据类型的定义及联合体变量的引用	272
8.4.2 联合体类型与结构体类型的区别	276
习题 8	278
第 9 章 枚举类型和位运算	285
9.1 枚举类型及其简单应用	285
9.1.1 枚举类型的定义和枚举变量的引用	285
9.1.2 枚举数据类型的简单应用	287
9.2 位运算及其应用	290
9.2.1 位运算符	290
9.2.2 位运算的简单应用	294
习题 9	298

第 1 章 C 程序设计初步

1.1 C 程序结构和处理过程

C 语言是 20 世纪 70 年代初期在贝尔实验室开发出来的一种广为使用的编程语言。C 语言早期主要用于 UNIX 系统,由于其具有强大的功能,很快被移植到其他操作系统平台下,在各类大、中、小和微型计算机上得到了广泛的使用。

1988 年,美国国家标准化协会(American National Standards Institute,ANSI)在综合各种 C 语言版本的基础上制定了 C 语言文本标准,称为 ANSI C 标准。1990 年,国际标准化组织(the International Organization for Standardization,ISO)公布了以 ANSI C 为基础制定的 C 语言国际标准 ISO C,即人们通称的标准 C。1988 年以后推出的各种 C 语言版本与标准 C 都是兼容的。1999 年,国际标准化组织又对 C 语言标准进行了修订,在原 C 语言特征的基础上,增加了一些面向对象的特征,命名为 ISO/IEC 9899:1999,简称 C99。目前流行的版本多是以 ANSI C 标准为基础开发的,但不同版本之间的语言功能和语法规则略有差别。

C 语言的特点如下:

- 语言简洁、紧凑。C 语言共有 30 多个关键字,9 种控制语句,程序书写形式自由。
- 功能强大。C 语言共有 34 种运算符,并把括号、赋值、强制类型转换都作为运算符处理,从而使表达式多样化,并可以实现比较复杂的运算。同时 C 语言包含整型、实型、数组类型、枚举类型等数据类型,丰富的数据类型使得 C 语言可以表达各种复杂的数据结构,具有很强的数据处理能力。
- 模块化和结构化的语言。C 程序由 3 种基本结构组成,分别是顺序结构、选择结构和循环结构,这三种结构组合可以完成任何复杂的任务。同时,C 语言将函数作为程序设计的基本单位,使得代码可以重用,程序便于维护和调试。
- 语法灵活,规则简洁。C 语言的语法限制不太严格,程序设计和书写形式自由度大。
- 具备低级语言和高级语言的双重功能。C 语言可以直接访问内存的物理地址,能进行位(bit)操作,实现对硬件的编程,也可以用于开发操作系统等项目。
- 程序效率高。C 语言程序生成目标代码质量高,程序执行速度快。
- 可移植性好。在 IBM PC 的 Windows 操作系统上编写的 C 语言程序,不用修改或稍作修改,就可其他计算机系统和操作系统运行。

1.1.1 C 程序的基本结构

下面通过两个示例来了解 C 语言源程序结构的特点,细节部分在后面的章节中逐步介绍。

【例 1.1】 在屏幕上输出信息：“欢迎大家学习 C 语言!”。

```
/* Name: ex0101.c */           //注释语句
#include <stdio.h>              //预处理命令
int main( )                    //主函数
{
    printf("欢迎大家学习 C 语言! \n"); //输出字符串
    return 0;
}
```

程序说明:

①第 1 行/*...*/括起来的为注释信息,用于说明程序的功能或程序的名字,对于程序的编译和运行不起作用。

②第 2 行中#include 是一个文件包含命令,作用是把头文件 stdio.h 包含到本程序中,成为程序的一部分。C 语言提供的头文件中包含各种标准库函数的函数原型,在程序中调用某个标准库函数时,必须将该函数原型所在的头文件包含进来。本程序中的头文件 stdio.h 中的函数主要处理数据流的标准输入/输出。

③第 3 行中的 main 是函数名,表示主函数。每个 C 程序必须有一个 main 函数,也只能有一个主函数。

④第 5 行是由 printf 函数构成的语句。printf 函数是 C 语言提供的标准输入输出库函数,作用是有关信息输出到显示器屏幕上。

⑤第 6 行是函数返回语句,return 0 表示 main() 正常结束。

【例 1.2】 从键盘输入两个整数,求两个数的和并将结果输出到屏幕上。

```
/* Name: ex0102.c */
#include <stdio.h>
int main( )                //主函数
{
    int numAdd(int a,int b); //函数声明
    int x,y,z;              //变量说明
    printf("please input x,y:\n");
    scanf("%d,%d",&x,&y);    //输入 x,y 值,注意书写格式
    z=numAdd(x,y);
    printf("%d+%d=%d\n",x,y,z ); //输出结果
    return 0;
}
int numAdd( int a,int b)    //定义 numAdd 函数
{
    int c;
    c=a+b;
    return c;
}
```

程序说明:

①本程序中包括两个函数,主函数 main 和被调用的函数 numAdd。numAdd 函数是一个用户自定义的函数,其功能是求两个数的和。

②C 程序中,每个函数都是一个相对独立的代码块,它们在程序中书写的顺序是任意的。被调用函数的定义如果出现在调用点之后,要对被调用函数进行声明。例如在本例中,主函数 main 调用了 numAdd 函数,而 numAdd 函数的定义在主函数之后,所以在主函数中对 numAdd 函数进行声明。关于函数有关知识,将在后面章节进行详细介绍。

上面两个简单 C 程序示例包含了 C 语言程序的基本组成部分,这些基本成分有:

①预处理命令。预处理是 C 语言编译程序提供的一项重要功能,它是对源文件进行编译之前,编译系统自动进行的一些准备工作。C 语言程序中,凡是以“#”开头的均为预处理命令,通常放在程序的开头,每条预处理命令单独占一行。预处理命令不是 C 语句,所以不需要用“;”结束。每个完整的 C 程序都会涉及数据的输入或输出,都需要使用预处理命令:#include <stdio.h>或者#include "stdio.h"将 stdio.h 文件包含到程序中来。

②C 程序由一个或多个函数组成,一个 C 程序中有且仅有一个主函数 main。主函数可以调用其他函数,但不能被其他函数所调用。一个 C 语言程序中,主函数可以出现在程序结构中的任意位置,程序总是从主函数开始执行,并在主函数中结束运行。

C 程序中的函数体用一对花括号“{}”括起来,函数体内是与函数功能相关的数据描述和 C 语句。

C 程序最常用的两种主函数框架如下所示,本书采用整型主函数形式。

#include<stdio.h>	#include<stdio.h>
int main()	viod main()
{	{
:	:
return 0;	}
}	

③注释。C 程序中的注释用来对程序工作过程或某些特殊功能进行注解作用,在程序中加入注释,可以提高程序的可读性和可维护性。所有的注释字符都被 C 编译器忽略,即程序不会执行其中的注释部分。C 程序中使用两种注释形式:

- 单行注释形式。以双斜杠“//”为引导,跟在后面的字符序列,一直到该行结束都是注释信息。单行注释可以书写在单独的一行上,也可以书写在可执行语句的后面。注释内容较少时,使用这种方法较为简便。

- 多行注释形式。以“/*”开始,以“*/”结束,中间括起的全部字符都是注释信息。当注释内容较多,用一个语句行书写不下时,通常使用这种方法。

在调试程序时,可以将暂时不使用的语句用注释符括起来,在编译时跳过这部分程序,待到调试结束后根据实际需要再决定去掉注释符恢复语句或者删除相应的语句。

④在 C 程序中,每一条语句都必须以分号“;”结尾。但预处理命令,函数头和花括号“{}”之后不需要加分号。

⑤标识符、关键字以及每一个相对独立的语言成分之间,必须至少使用一个空格来间隔。

⑥C 程序书写格式比较自由,一行可以写一个语句,也可以写多个语句,也可以将一个

语句写在多行上。书写程序时,建议每条语句占用单独的一行,采用适当的缩进形式,使程序层次分明。

1.1.2 C 程序的处理过程

用符合 C 语言规范的方式书写并保存的 C 程序称为源程序文件,源程序文件不能直接执行,需要将它翻译成计算机能够识别并执行的机器语言程序。一个 C 语言程序的完整处理过程一般分为 4 个步骤:

第一步:编辑。

编辑是指 C 语言源程序的输入和修改,程序保存时,文件名由用户自己选定,扩展名一般为“.c”,也可以使用开发环境默认的扩展名。

第二步:编译。

编译是把 C 源程序翻译成计算机可以理解并执行的机器语言组成的程序。C 语言的编译过程分为两个阶段:第一阶段是编译预处理,处理所有的预处理命令。第二阶段是编译程序。编译程序对源程序进行句法和语法检查,当发现错误时,错误的类型和在程序中的位置将被显示出来,以帮助用户进行修改,若没有发现句法和语法错误,则生成目标文件,目标文件默认和源文件同名,其扩展名为“.obj”。目标文件还是不能被执行,它们只是些在内存中可重新定位的目标程序模块。

第三步:连接。

连接也称为链接,是用连接程序将与当前程序有关的、已经存在的几个目标模块链接在一起,形成一个完整的程序代码文件。经过正确连接所生成的文件才是可执行文件。可执行文件默认与源文件同名,扩展名为.exe。

第四步:执行。

若程序执行后获取了正确结果,则程序处理过程完成;若程序执行后没有获得期望的结果,表示该程序有逻辑错误,这时需要调试程序。图 1.1 表明了上述过程。

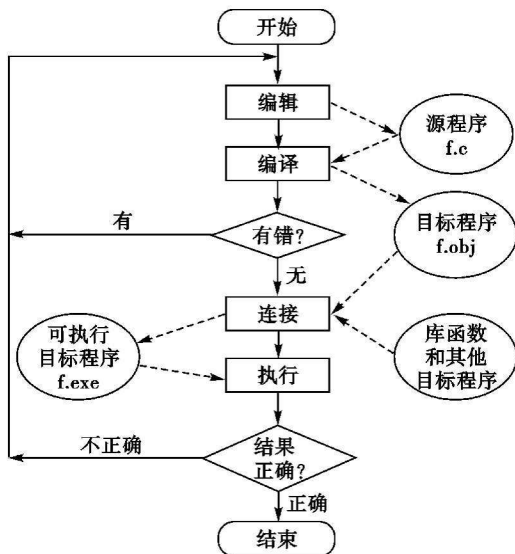


图 1.1 C 语言程序的执行过程

本教材选用 Microsoft Visual C++ 6.0 IDE 作为教学环境,使用该环境处理 C 程序的方法请参考附录 C。

1.2 C 语言的基本数据类型

在 C 语言中,每一个数据都具有特定的数据类型,C 语言的数据类型主要有整型、实型、字符型、数组、指针等。

1.2.1 C 程序中数据的表示

C 语言程序的基本构成要素包括:字符集、标识符、保留字、常量、变量、运算符等。

1.C 语言的字符集

字符是组成语言的最基本的元素。每种程序设计语言都规定了书写源程序时允许使用的特定的字符集。C 语言字符集由以下字符组成:

- 字母:小写字母 a~z,大写字母 A~Z。
- 数字:0~9 共 10 个。
- 空白符:空格符、制表符、换行符统称为空白符,空白符只在字符常量和字符串常量中起作用。在其他地方出现时,只起间隔作用,编译程序对它们忽略不计。
- 特殊字符:+、=、-、_、()、*、&、%、\$、!、|、<、>、\、,、;、\:、“ ”、‘ ’、/、?、{}、[]、^。

2.标识符和保留字

标识符是用来区分不同实体的符号。在 C 程序中,标识符可以用作变量名、符号名、函数名、文件名等。标识符分为两大类:保留字和用户标识符。

保留字(又称为关键字)是 C 语言中有特定意义和用途、不得作为他用的字符序列,其中 C89 标准规定的保留字有 32 个,C99 标准增加了 5 个,详见表 1.1,其中标有星号上标的保留字是在 C99 标准中增加的。

表 1.1 C 语言保留字

流程控制	break	case	continue	default	do	else
	for	goto	if	return	switch	while
存储属性	auto	extern	static	register		
数据类型	_bool*	_complex*	_imaginary*	char		
	const	double	enum	float	int	long
	restrict*	short	signed	struct	union	unsigned
	void	volatile				
其他	inline*	sizeof	typedef			

注意:所有的 C 保留字必须小写。

用户标识符是在程序中给用户自定义数据对象(如变量、常量、函数、数据类型等)命名的符号。在不混淆的情况下,把用户标识符称为标识符。

C 语言要求标识符遵循下列规则:

- ①标识符是由字母、数字和下划线构成的序列,不能以数字开头。
- ②标识符中,大小写是有区别的。ab 和 AB 是不相同的标识符。
- ③不能用保留字或关键字做标识符。

以下标识符是合法的:

```
g    a4    file_name    _ping
```

以下标识符是非法的:

```
63c                //以数字开头
char                //与保留字同名
first-name          //出现了非法字符“-”
```

关于标识符有几点建议:

①为了增加程序的可读性和清晰性,标识符最好能够表达其所代表数据对象的物理含义,即尽量做到“见名知义”。

②尽量避免使用容易混淆的字符。例如:

0(数字)、O(大写字母)、o(小写字母);

1(数字)、l(大写字母)、i(小写字母);

2(数字)、Z(大写字母)、z(小写字母);

③名字不要太短,可以采用词组形式。采用词组形式时,可采用如下格式:

- 骆驼命名法,名字中的每一个逻辑断点都用一个大写字母标记。例如,myFirstName。
- 下划线命名法,即使用下划线作为一个标识符中的逻辑点,分隔所组成标识符的词汇,例如,my_first_name 等。

④名字不可太长,虽然标准 C 不限制标识符的长度,但它受各种版本的编译系统和具体机器的限制,同时,太长的名字在书写时容易出错。

3. 常量和变量

在程序执行过程中,其值不能被改变的量,称为常量;其值有可能发生变化的量,称为变量。在程序中,常量可以不经说明而直接引用(即直接书写在程序代码中),而变量则必须先定义后使用。定义变量时,不但要用合适的标识符命名,而且还必须说明变量的数据类型,以便编译器确定该变量的存储和处理方法。变量定义的一般形式为:

```
数据类型名 变量名 1,变量名 2,...;
```

4. 运算符

C 语言提供了丰富的运算符,运算符与变量、函数一起组成表达式,实现各种运算功能。运算符由一个或多个字符组成。

1.2.2 C 语言基本数据类型

数据类型是数据的基本属性,描述数据的存储格式和运算规则。不同类型的数据在内存中所占存储空间的大小是不同的,能够支持的运算和相应的运算规则也不同。

在 C 语言中,数据类型可分为 4 类,它们是本数据类型、构造数据类型、指针类型、空

类型,如图 1.2 所示。

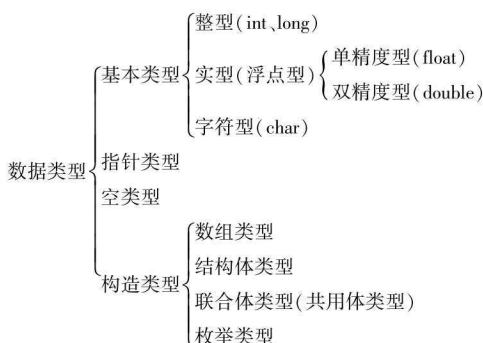


图 1.2 C 语言的数据类型

不同数据类型的存储空间大小和取值范围都不同,与使用的编译器版本也有一定的关系,见表 1.2。

表 1.2 C 语言的标准数据类型

类 型	关键字	Turbo C++编译器		Visual C++编译器	
		字节数	数据表示范围	字节数	数据表示范围
有符号整型	int	2	$-2^{15} \sim 2^{15}-1$	4	$-2^{31} \sim 2^{31}-1$
有符号短整型	short	2	$-2^{15} \sim 2^{15}-1$	2	$-2^{15} \sim 2^{15}-1$
有符号长整型	long	4	$-2^{31} \sim 2^{31}-1$	4	$-2^{31} \sim 2^{31}-1$
无符号整型	unsigned int	2	$0 \sim 2^{16}-1$	4	$0 \sim 2^{32}-1$
无符号短整型	unsigned short	2	$0 \sim 2^{16}-1$	2	$0 \sim 2^{16}-1$
无符号长整型	unsigned long	4	$0 \sim 2^{32}-1$	4	$0 \sim 2^{32}-1$
单精度浮点型	float	4	$3.4e-38 \sim 3.4e+38$	4	$3.4e-38 \sim 3.4e+38$
双精度浮点型	double	8	$1.7e-308 \sim 1.7e+308$	8	$1.7e-308 \sim 1.7e+308$
字符型	char	1	$-2^7 \sim 2^7-1$	1	$-2^7 \sim 2^7-1$
无符号字符型	unsigned char	1	$0 \sim 2^8-1$	1	$0 \sim 2^8-1$

1. 整型数据

整型数据分为两大类:有符号型和无符号型。有符号的整数既可以是正数,也可以是负数。不带符号位(只包含 0 和正数)的整数为无符号整数。

在 C 语言中整型数据用机器的一个字长来存储,所以整型数据的表示范围与计算机系统的软硬件环境有关。在字长为 16 位的计算机系统中,整型数据占用 2 个字节,在字长为 32 位的计算机系统中,整型数据占用 4 个字节。同样一个整型数据,在取值范围能满足要求的情况下,应尽量使用较短的数据类型。

在 C 语言中,整型常量可以用以下形式表示:

- 十进制整数:没有前缀,其数码为 0~9,如 76、-33 等。
- 八进制整数:以 0 作为前缀,其数码为 0~7,如 016、-052 等。

• 十六进制整数:以 0x 或 0X 为前缀,其数码为 0~9,A~F 或 a~f,如 0x5d、0X4af 等。

若要表示一个长整型常量,则应在数字后面加上后缀“l”或“L”表示,如 136L。而无符号数则应在数字末尾加后缀“u”或“U”表示。例如:0xE5LU 表示十六进制无符号长整数 E5。

在 C 程序中,用于存放整型数据的变量称为整型变量。整型变量有短整型、基本整型、长整型和无符号整型四种,其数据类型名分别由关键字 short int(或 short)、int、long int(或 long)与 unsigned int(或 unsigned)表示。例如:

```
int x,y;           //定义 x,y 为整型变量
long a,b;          //定义 a,b 为长整型变量
unsigned t,q;       //定义 t,q 为无符号整型变量
```

变量定义时可以进行初始化,即在定义变量的同时通过赋值号(=)给变量赋值,例如:

int a = 10, b = 24; 等。

【例 1.3】 整型数据在程序中的使用示例。

```
/* Name: ex0103.c */
#include <stdio.h>
int main()
{
    int a,b,c;
    long x,y,z;
    a=45;           //常量值直接赋给变量
    printf("请输入整型变量 b 的值:"); //提示用户输入数据
    scanf("%d",&b); //输入数据
    c=a+b+83;       //常量值直接写在表达式中
    printf("a=%d,b=%d,c=%d\n",a,b,c); //输出变量 a,b,c 的值
    printf("请输入长整型变量 x 和 y 的值:");
    scanf("%ld,%ld",&x,&y); //输入数据,用逗号分隔
    z=x-y;
    printf("%ld-%ld=%ld\n",x,y,z); //输出变量 x,y,z 的值
    return 0;
}
```

上面程序中的格式输入函数 scanf 和格式化输出函数 printf 的使用将在稍后章节中讨论,目前仅需知道可以通过这种方式输入输出数据即可。程序一次运行的情况如下所示:

请输入整型变量 b 的值:68

a = 45, b = 68, c = 196

请输入长整型变量 x 和 y 的值:7895678,2346485

7895678-2346485 = 5549193

2. 实型数据

C 语言中,实型常量也称为实型常数、实数或者浮点数。在计算机内存中,实数一律以