高等学校程序设计课程系列教材
程序设计学习健康跑系列教材

C语言程序设计

王立柱 编著

高等教育出版社

高等学校程序设计课程系列教材
程序设计学习健康跑系列教材

C语言程序设计

王立柱　编著

高等教育出版社·北京

内容提要

本书共9章内容：C语言程序基本结构、函数、指针和数组、顺序表、结构、字符串、文件、链表、二维数组和指针。

本书从属于一个系列——“程序设计学习健康跑”，包括《C语言程序设计》《C++语言程序设计》《数据结构与算法》三本书。

本书由一个程序序列贯穿，每一个程序都是在前一个程序的基础上改进而成，拾级而上；每一个概念就在这个过程中应需而生。

本书配有MOOC版多媒体课件：既可以助教，又可以助学；而且可以通过录播，直接生成MOOC。

本书既可作为高等院校本科和专科C语言程序设计课程的教学用书，又可作为编程爱好者的自学教材。

图书在版编目（CIP）数据

C语言程序设计 / 王立柱编著. -- 北京：高等教育出版社，2019.2

ISBN 978-7-04-051132-1

Ⅰ. ①C… Ⅱ. ①王… 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（2019）第005883号

策划编辑	韩　飞	责任编辑	韩　飞	封面设计	王　鹏	版式设计	童　丹
插图绘制	于　博	责任校对	吕红颖	责任印制	田　甜		

出版发行	高等教育出版社	网　址	http://www.hep.edu.cn
社　址	北京市西城区德外大街4号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印　刷	人卫印务（北京）有限公司		http://www.hepmall.com
开　本	787mm×1092mm　1/16		http://www.hepmall.cn
印　张	15.75		
字　数	360千字	版　次	2019年2月第1版
购书热线	010-58581118	印　次	2019年2月第1次印刷
咨询电话	400-810-0598	定　价	32.80元

物 料 号　51132-00

王立柱，天津师范大学教授，湖北工业大学特聘教授，教育部－微软精品课主持人，主编多本国家级规划教材。先后获得天津市教学成果二等奖，湖北省教学成果一等奖。

C语言程序设计

王立柱

1 计算机访问http://abook.hep.com.cn/187902，或手机扫描二维码、下载并安装Abook应用。

2 注册并登录，进入“我的课程”。

3 输入封底数字课程账号（20位密码，刮开涂层可见），或通过Abook应用扫描封底数字课程账号二维码，完成课程绑定。

4 单击“进入课程”按钮，开始本数字课程的学习。

课程绑定后一年为数字课程使用有效期。受硬件限制，部分内容无法在手机端显示，请按提示通过计算机访问学习。

如有使用问题，请发邮件至abook@hep.com.cn。

http://abook.hep.com.cn/187902

前　言

这里是罗陀斯，就在这里跳跃吧！

这里有玫瑰花，就在这里起舞吧！

在提倡全民编程的时代，作为教师，经常被问及三个问题：一是学习编程必须学习哪些内容；二是学习编程有什么更有效的方法；三是学习编程有什么更深刻的意义。

1

什么是程序？专业上有一个关于程序本质的经典概括：算法 + 数据结构 = 程序。算法是问题求解的有限步骤，数据结构是算法实现的工具，因此学习编程必须学习数据结构：工欲善其事，必先利其器。

编程需要编程语言，C++ 语言不仅是应用广泛的系统编程语言，而且包含着常用数据结构的标准化即标准模板库，现在的很多编程语言，例如 Java 和 Python，都是在这个基础上开发出来的。因此，学习编程必须学习 C++ 语言。

C++ 语言是在 C 语言的基础上直接发展而来的。用李未院士的“三个语言环境”理论来讲，C++ 是对象语言，C 是解释 C++ 的元语言，C 对 C++ 的解释性关系是模型语言，数据结构是这种解释性关系的一种理想模本，三者密不可分。因此，学习编程必须学习 C 语言。

如上所述，学习编程，必须把 C、C++ 和数据结构作为一个有机的整体来学习。于是，第二个问题可以落实为：学习 C、C++ 和数据结构有什么更有效的方法？

2

练习书法先要临摹字帖，学习写作先要阅读经典。C++ 创始人本贾尼说，学习编程，如同学习写作，先要模仿经典程序。

什么是经典程序？对 C、C++ 和数据结构这个整体来说，C++ 标准模板库就是经典程序，因为它是常用数据结构的标准化。

但是在很多人看来，C++ 标准模板库高不可攀。于是，本书把第二个问题又进一步落实到如何搭建阶梯，使每一个学习编程的人都可以轻松地拾级而上，模仿标准模板库的常用代码。这个阶梯是一个程序序列，它从一个简单程序开始，每一个程序都是在前一个程序的基础上稍加改进而成。它将学习模式从传统的概念引领改为程序引领，概念不再是机械的灌输而是实践的概括。

这个程序序列在 C 语言部分是如何搭建的呢？这要从类型讲起。C 语言和 C++ 语言一样，都是基于类型的编程语言。类型有两类，一类是简单类型，另一类是复合类型。简单类型是独立的类型，主要有整型、实型和字符型。这些类型的差别主要是输入、输出格式符的差别，这种差别使得对一种类型的学习，很容易转换到对另一种类型的学习。因此，本书以整型为主，从一个整数的按位逆置输出程序开始，通过功能的不断扩展，形成一种程序序列，贯穿第 1 章 C 语言程序基本结构和第 2 章函数。

复合类型是依赖其他类型而存在的类型。指针和数组是核心复合类型，它们相互依赖，相互作用，解决了一系列综合性程序设计问题，由此构成若干主题，每一个主题都是一个程序序列，称之为主题程序序列。一个主题程序序列构成一章："指针和数组密切相关"，这是第3章指针和数组的主题；"顺序表是带有基本函数的数组"，这是第4章顺序表的主题；"字符串既是特殊的数组，也是特殊的顺序表"，这是第6章字符串的主题；"文件是隐式的顺序表"，这是第7章文件的主题；"链表是拓扑意义上的顺序表"，这是第8章链表的主题。

显然，在本书的第3章至第8章，顺序表起到承上启下的作用。但是顺序表在传统教法中是数据结构的内容，在一般的C、C++和数据结构的系列课程中是最后的环节。讲授方法是：概念、伪码和实现——从一般到个别。而在本书中，顺序表是作为改进的数组而存在的，在本书代表的C、C++和数据结构的系列中，它是首要环节。讲授方法是在一个数组求和程序的不断完善中自然生成的——从个别到一般。

在程序序列生成的过程，用到若干科学方法，如对称性和类比，这些方法使学习更轻松，更有效，避免了知识碎片化。所谓对称性是指变换下的不变性。很多程序都是由前一个程序经过变换而来，但是原来程序的基本功能不变。经过变换，程序的用户体验更好，复用性更高，功能更强，阅读和调试更容易。即使像顺序表，顺序表类，标准模板库中的Vector和List这样重要的代码，都是经过对称变换生成的。类比是大家熟悉而见效的方法，主题程序序列的生成很多都用到这种方法，例如，C字符串分别与数组和顺序表进行类比，文件与顺序表进行类比。

本书不包含递归，这是因为递归是非线性算法的必要手段，而非线性算法是数据结构的主要内容。将正确的内容放在正确的地方，可以使科学方法有用武之地，事倍功半。例如，将汉诺塔和二叉树中序遍历类比，快速排序和二叉树前序遍历类比，通过这些类比，不仅可以轻松地实现复杂的递归，而且可以轻松地将递归转为迭代。

本书配有用Authorware开发的课件，与章节一一对应，将抽象算法形象化，运行过程虚拟化，代码和算法同步化。毕竟，文不如表，表不如图，一个精准的演示，胜过千言万语。

3

从中学到大学，学习的逻辑内容发生了根本变化：中学学习形式逻辑，大学学习辩证逻辑，即辩证法。恩格斯说，"一个民族想要站在科学的最高峰，就一刻也不能没有理论思维"。这里的理论思维主要是指辩证逻辑思维。

在中学，形式逻辑的主要模本是几何学，在大学，辩证逻辑的主要模本是高等数学和物理学。在提倡全民编程的时代，是否应该将程序设计也作为辩证逻辑的模本呢？程序设计既具有高等数学的逻辑性，又具有物理学的实验性，而且作为计算机科学的核心，已经深入到每一个研究领域，促进着每一个领域的变革。我们坚信，将程序设计作为辩证逻辑的模本，是一种理想的选择。

C、C++和数据结构，就是在这个方向上进行整合的，本书也是在这种整合中进行布局的：复合类型是依赖其他类型而存在的类型，这不是辩证逻辑意义上的概念吗？将指针和数组视为最重要的复合类型，由它们的相互依赖和相互作用的结果构成本书的主要章节，这不是丰富了辩证逻辑的科学内涵吗？

4

如何使用本书,这里提出若干建议。

一是注重概念。本书中的概念都是程序设计的阶段性概括,它可以帮助学生更好地理解程序,再现程序。因此,作业和测试应设简答题、选择题和判断,用以检查概念。

二是注重模仿。学习编程从模仿开始,顺序表基本函数,字符串基本函数和链表基本函数都是应该模仿的程序。因此,作业和测试应设模仿题:自定义顺序表基本函数、自定义字符串基本函数和自定义链表基本函数。每一道题都包含一个主函数,用来调用相应的基本函数,检验是否正确。不要担心学生死记硬背,因为程序只有理解记忆,才能调试通过。

三是注重文档。作业和测试要包含两类程序设计题:一类是根据内部文档编写程序,即根据功能描述和步骤说明来编写程序;二是根据功能描述编写程序,同时要给出步骤说明。

四是以赛带学。阶梯搭建起来了,人人都可以登临,比赛是最好的促进学习的手段。比赛内容分为三种:“微马”“半马”和“全马”。C 语言部分只是“微马”,主要内容是第二部分:自定义顺序表基本函数、自定义字符串基本函数和自定义链表基本函数。

5

把 C、C++ 和数据结构整合为辩证逻辑的模本,使更多的人以健康跑的形式不仅可以有效地学习,而且可以轻松快乐地学习。本书和后续的《C++ 语言程序设计》《数据结构与算法》构成一个“程序设计学习健康跑”系列,由王立柱教授编写完成,前后一共改版 5 次,相应的改革先后在天津师范大学、北京商务学院、对外经贸大学和湖北工业大学实验推广,得到很多专业人士和朋友的支持和鼓励,借此机会,表示衷心感谢。本书肯定还存在许多需要改进的地方,希望能得到读者更多具有建设性的意见。

编　者

2018 年 11 月 28 日

目　录

第1章 C语言程序基本结构

C语言和多数编程语言一样，核心概念是类型。

——Bjarne Stroustrup

计算机程序是用编程语言表示的一组语句，一条语句是一条指令，每条指令一般都指示计算机完成一个数据处理步骤，计算机按步骤完成一个特定的数据处理任务。完成一个特定的数据处理任务所需的有限步骤称为**算法**。

在C编程语言中，数据按类型划分，例如，整型、实型、字符型等。而任何类型的数据在计算机处理中都首先需要存储。计算机存储器也称**内存**，由一些连续的字节构成，每一个字节有8位，每一位存储0或1。**数据类型**决定了数据内存单元的大小、存储格式以及对数据可以实施的基本操作。

例如：一个整数345，其类型为整型，它占4个连续字节，这是内存空间的大小；一个字节有8位，4个字节一共32位，用1位存储符号，其他存储数值，这是存储的格式；对整数可以实施加(+)、减(-)、乘(*)、整除(/)、求余(%)等运算，这是对整数可以实施的基本操作。

C语言程序设计是在数据类型的基础上展开的。下面的学习从整型开始，整型、实型和字符型是C语言基本类型，而且一种类型的学习，很容易平移到另一种类型。

1.1 第一个C语言程序

如何编写并运行一个C语言程序呢？下面通过一个具体实例来熟悉它的基本过程。这个实例是，将整数345在显示器上按位逆序输出，即输出543。

1.1.1 编程基本过程

1. 对象和变量

计算机所处理的数据既要按类型划分，又要存储，这就产生了一个概念：对象。所谓**对象**是指与类型相关联的内存单元。有名称且其值可以改写的对象称为**变量**。用名称来读写对象中的数据称为**直接访问**（也称**直接寻址**）。

程序员一般通过变量定义得到变量。变量定义格式如下：

类型 变量1,变量2, …,变量*n*;

类型分基本类型、复合类型和程序员定义类型，基本类型的标识符是系统**保留字**，也称**关键字**，例如，整型是基本类型，关键字是int。变量名是程序员根据命名规则来命名的（参见附录A）。变量名要尽可能反映数据的意义，以助于阅读理解。逗号是变量分隔符。分号是语句结束符。以整型变量定义为例：

```
int n;                          // 定义一个整型变量n
```

其中，int表示整型，n是变量名。执行这条语句之后，程序员就得到一个名称为n的整型对象。由双斜杠（//）引导的文字是注释，用于语句的解释，帮助阅读理解，它不是语句。一个双斜杠只能引导一行注释。

有了变量之后，一般用赋值操作给变量赋值，例如：

```
n=345;                          // 给变量n赋值
```

其中“=”是赋值操作符。赋值操作从赋值操作符的右元读取值，写入左元。

变量的定义和赋值可以合并为一条语句，称为变量**初始化**：

```
int n=345;                      // 整型变量n初始化
```

2. 已知程序

编写一个程序一般都要调用已知的程序，用已知求解未知。已知程序主要分三类：数据类型提供的基本操作，C语言自带的库函数，程序员自己设计的函数。

（1）数据类型提供的基本操作，这是C语言的根基。它们一般用运算符表示。例如，整型的基本操作有加（+）、减（-）、乘（*）、整除（/）、求余（%）等。要把一个整数按位逆序，需要用到整除（/）和求余（%），例如，假设整型变量n的值是345，那么用10求余（n%10）可以得到个位数5；用10整除，再用10求余（n/10%10）可以得到4，等等。

（2）C语言自带的库函数，这是常用的C语言程序。在显示器上输出一个数值要用到C

语言自带的函数printf( )。C语言自带一些常用函数,这些函数称为**标准库函数**。标准库函数按类划分,每一类都包含在一个扩展名为.h的文件中。这些文件称为**系统头文件**。例如,printf( )包含在stdio.h中,常用数学函数包含在math.h中。一个程序要调用一个标准库函数,必须使用文件包含命令,将该函数所在的头文件加入到程序中。文件包含命令的格式如下:

```
#include< 系统头文件 >
```

例如:

```
#include<stdio.h>
```

命令不是语句,结尾不带分号。

printf( )的使用格式如下:

```
printf(格式控制字符串,输出参数);
```

其中,格式控制字符串以双引号为界限符,包含格式说明符,用以指定输出参数的类型和输出格式。例如:

```
printf("%d",n);
```

其中%d是格式说明符,n是输出参数,%d指定输出参数的值按整型十进制格式输出。格式说明符和输出参数要对应,类型要一致。毕竟,C语言处理的数据都是某种类型的数据。

关于printf( )的其他内容,随着程序设计的逐步深入而引入。

(3)程序员自己设计的函数。这部分内容从第2章开始介绍。

3. 程序设计

编写程序,将整型变量n的值按位逆序输出,已知n的值是345。

程序设计一般包含算法设计和编程语言描述即实现。如果算法比较简单,或者编程语言的语句功能很强,那么算法的每一个步骤都可以直接用语句来描述。现在的设计就属于这种情况。

(1)输出个位数。用10对n求余,得到个位数5。然后调用函数printf( )将5输出到显示器。

```
printf("%d",n%10);          // 输出n的个位数5
```

(2)输出十位数。用10整除n,把n的值降为34,然后输出其个位数4。

```
n=n/10;                     // 把n的值降为34
printf("%d",n%10);          // 输出n的个位数4
```

(3)输出百位数。继续用10整除n,把n的值降为3,然后输出其个位数3。

```
n=n/10;                     // 把n的值降为3
printf("%d",n);             // 输出n的个位数3
```

把上面的语句依次编辑到C程序框架,就得到一个C程序。C程序框架是一个主函数框架,如下所示:

```
int main( )                 // 主函数头
{                           // 主函数体开始
    一组C语言语句           // 每条语句以分号结束
```

```
    return 0;                    //把0传递给系统,表示程序正常结束
}                                //主函数体结束
```

程序 1.1 将一个整数在显示器上按位逆序输出。

```
#include<stdio.h>                //标准输入输出函数库
int main( )
{
    int n=345;                   //整型变量n初始化
    printf("%d",n%10);           //输出n的个位数5
    n=n/10;                      //把n的值降为34
    printf("%d",n%10);           //输出n的个位数4
    n=n/10;                      //把n的值降为3
    printf("%d",n);              //输出n的个位数3

    printf("\n");                //输出一个换行符

    return 0;
}
```

程序运行结果:

```
543
```

程序分析:

(1) 变量定义或初始化要置于其他操作语句之前。

(2) 变量必须先赋值,后处理。一个变量如果没有赋值,那么它存储的是一个无意义的值。处理这种变量,其结果也是无意义的,称之为“垃圾入,垃圾出”(garbage in, garbage out)。

(3) 语句 printf("\n")的功能是输出一个换行符,使输出结果更清楚。它没有对应的输出参数。一个反斜杠加一个字符 n,构成一个转义字符(参见附录 B.3),用于控制,表示换行。

(4) 程序 1.1 的语句是自上而下依次执行的,这种语句关系称为**顺序结构**。

(5) 注释部分是一个程序的内部文档,用于解释一条语句或一组语句。

(6) 一个整型对象占 4 个字节,一个字节 8 位,一共 32 位。最高位表示符号(0 代表正,1 代表负),其余的位表示数值。数值范围是 -2^{31}~$2^{31}-1$,即 -2 147 483 648~2 147 483 647,超出这个范围称为**算术运算溢出**。

1.1.2 集成开发环境

用 C 语言编写的程序称为 C **源代码**或 C **程序文本**,C 源代码文件的扩展名是 .c 或 .h。编写工作一般是在计算机上利用**编辑器**完成的。C 源代码必须转换为计算机可以执行的**机器代码**或**目标代码**,完成这一转换过程的是**编译器**。目标代码文件的扩展名在 Windows 中是 .obj,

在UNIX中是.o。一个C源代码一般由若干部分构成,不同部分一般由不同人编写。例如,程序1.1的主函数是我们编写的,库函数中的标准输出函数是别人编写的。每一部分称为一个**翻译单元**。由每一个翻译单元转换而成的目标代码需要链接,生成**可执行程序**,完成这一工作的是**链接器**。可执行程序文件的扩展名是.exe。它们之间的关系如图1-1所示。

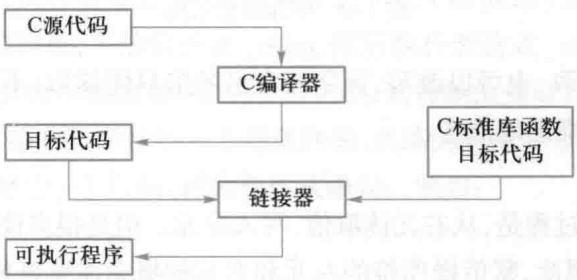

图1-1 从C源代码到可执行程序示意图

编辑器、编译器和链接器都是软件,都是程序员的开发工具。把它们一体化的软件称为"**集成开发环境**"(integrated development environment, IDE)。集成开发环境不止一种,具体使用哪一种视具体情况而定。

一个C源代码,在编译、链接和运行过程中,难免出现错误,查找并修改错误的过程称为**程序调试**。

由编译器发现的错误称为**编译时错误**。语法不正确的都属于这类错误。例如,变量名不符合命名规则,关键字拼写有误,语句结束时丢失分号,括号不匹配。另外,标点符号用了中文符号,而没有用西文符号也是编译时错误。

由链接器发现的错误称为**链接时错误**。例如,调用了并不存在的函数。

程序运行时才发现的错误称为**运行时错误**或**逻辑错误**。以程序1.1为例,把语句n=n/10写成n/100,运行结果不正确,这是算法错误,是典型的逻辑错误。最难调试的是运行时错误或逻辑错误,因为一般没有任何错误提示。

除逻辑错误之外,其他错误一般都有错误提示。必须从第一个错误开始查找和修改,而每修改一个错误,都要重新编译,因为后面的错误可能是前面的错误引起的。

集成开发环境不同,错误的归类也不同,但是错误提示中一般都会指出错误类型。

1.1.3 字面常量、左值和右值

1. 字面常量

C程序所处理的数据都是存储在对象中的数据,因此,在C程序中,数据都是用对象表示的。程序1.1中的345也不例外,它和变量n一样,也有内存单元,这个单元也是对象。只是这种对象的值不可改变。把其值不可改变的对象称为**常量**。而且,345是一种特殊的常量,它的标识符由其存储的值和表示类型的后缀或界限符所组成,这种常量称为**字面常量**。例如,52388L是长整型字面常量,其中52388是存储的值,L表示长整型(长整型与整型的区别主要

是占用字节的多少）；345 是普通整型字面常量，它的类型是默认的，这时，常量的标识符就是常量的值，如图 1-2 所示。

图 1-2 字面常量示例

变量的值既可以读取，也可以改写，而字面常量的值只能读取，不能改写。这种差异可以用左值和右值的概念来准确地描述。

2. 左值和右值

赋值操作符的执行过程是，从右元读取值，写入左元。但是很多读写操作并不是显式地用赋值操作符来表示的，因此，赋值操作符的左元和右元的概念需要延伸，延伸后的概念便是左值和右值。

如果一个对象的值可以读取，那么这个对象称为**右值**（r-value，读作 are-value）；如果一个对象的值可以改写，那么这个对象称为**左值**（l-value，读作 ell-value）。左值肯定是右值，反之不然。变量是左值，因此也是右值，而字面常量只能是右值。例如：

```
345=n;                    // 错!
```

编译器就会报错：'=': left operand must be l-value。意思是，赋值操作符的左元必须是左值。

1.1.4 表达式

在程序 1.1 中，345、n、n=345、n%10 和 n=n/10 有一个统一的名称：**表达式**。何谓表达式？常量和变量是表达式，由操作符依法连接起来的表达式依然是表达式。

每一个表达式都有一个确定的值。在计算机中，数据都是用对象表示的，表达式的值也不例外，不过这个对象可能是显式对象，也可能是隐式对象。例如，表达式 n 和 345，它们的值存储在各自所表示的对象中，这是显式对象。

每一个操作符和其操作数所构成的表达式都是一个基本操作，而基本操作的本质是程序，表达式的值是这个程序的结果。这个值一般都存储在一个隐式对象中。以表达式 n%10 为例：假设 n 的初值是 345，计算机从对象 n 和 10 中分别取值 345 和 10，用 10 对 345 求余，将结果 5 存入一个隐式对象。这时 n 的值并没有变化，还是 345，而表达式的值是隐式对象的值。

表达式 n=n/10 由两个操作符构成。假设 n 的初值是 345。系统先执行表达式 n/10：从对象 n 和 10 中分别取值，计算后将结果 34 存入一个隐式对象，不妨假设这个隐式对象是 _temp。然后执行表达式 n=_temp，结果 n 的值为 34，这也是表达式的值，但它存储在一个隐式对象中。

在 C 语言中，如果表达式的值存储在一个隐式对象，那么表达式只能是右值，不能是左值。下面举例说明：

```
x=y=20                    // 正确。相当于 x=(y=20)
```

```
(x=y)=20                    //错
```

表达式 x=y=20 的执行顺序是,先执行表达式 y=20,y 的值是 20,表达式的值也是 20,但存储在一个隐式对象,不妨假设为 _temp,然后执行表达式 x=_temp,x 的值是 20,表达式的值也是 20,但存储在一个隐式对象中。

表达式(x=y)=20 的执行顺序是,先执行表达式 x=y,x 的值是 y 的值,表达式的值也是 y 的值,但存储在一个隐式对象,不妨假设是 _temp,然后执行表达式 _temp=20,这就错了,因为隐式对象不能是左值。值得一提的是,在 C++ 语句中,这种赋值是可行的,但这绝不是一个简单的语法问题,而是对 C 语言局限性的一个创造性的、具体的解决方案,是学习 C++ 的起点。

一个表达式加分号便是一个语句,称为**表达式语句**。例如:

```
n=345;
x=y=20;
n=n/10;
```

1.1.5 对象的地址

计算机存储器(也称内存)是一个连续的字节序列(一个字节有 8 位),每一个字节都有一个用整数表示的地址。地址从 0 开始,相邻的字节,地址相差 1。存储器有多少字节,取决于地址线有多少。在 32 位操作系统中,地址线有 32 根,地址取值范围是 $0\sim2^{32}-1$,因此,存储器的字节数是 2^{32}(4 GB)。

在 C 语言中,一个对象占几个字节即一个对象的大小由类型决定。例如,一个整型对象占连续 4 个字节,首字节的地址是该对象的地址,如图 1-3 所示。

在一个函数中定义的对象,例如在函数 main( )中定义的 n,其内存单元在函数执行时由系统生成即分配,在函数结束时,由系统撤销即收回。函数的一次执行时间称为该对象的**生命周期**。在这个周期,可以利用取址符 & 获得该对象的地址。地址是计算机的硬件特征,利用地址可以提高程序的效率。

但是对字面常量不能寻址。例如,&345 是错的。为什么呢?因为这种对象的生命周期只是它所在的表达式的一次执行时间。对这种“转瞬即逝”的对象寻址是没有意义的。类似的,如果表达式的值存储在一个隐式对象,那么对表达式不能寻址,例如,&(n%10)是错的。

现在得知,一个对象至少有三个值:它存储的值、它的地址和它的字节数。利用取址符 & 可以取得对象的地址,利用操作符 sizeof 可以计算对象的字节数。

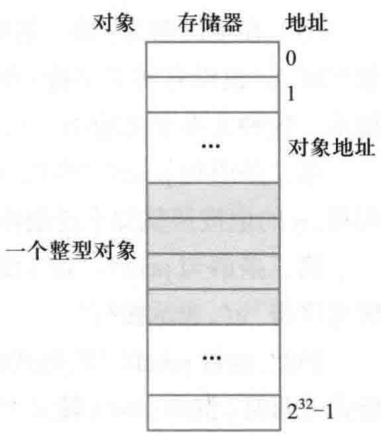

图 1-3 地址

程序 1.2 输出一个整型变量的值、地址和字节数。

```
#include<stdio.h>
int main( )
```