全国高等职业学校·高等专科学校教材

数据结构

杨 枨 主编

高等教育出版社

全国高等职业学校·高等专科学校教材

数据结构

主编 杨 彬

高等教育出版社

图书在版编目(CIP)数据

数据结构/杨枨主编．—北京：高等教育出版社，2000.5
ISBN 7-04-008383-3
Ⅰ．数… Ⅱ．杨… Ⅲ．数据结构—基本知识
Ⅳ．TP311.12
中国版本图书馆 CIP 数据核字(2000)第 60405 号

责任编辑 杨歆颖　　**特约编辑** 吴燕萍
封面设计 乐嘉敏　　**责任印制** 潘高峰

书　　名 数据结构
主　　编 杨枨

出版发行 高等教育出版社
社　　址 北京市东城区沙滩后街 55 号　　**邮政编码** 100009
电　　话 010-64054588　　**传　　真** 010-64014048
021-62587650　　021-62551530
网　　址 http://www.hep.edu.cn

经　　销 新华书店上海发行所
排　　版 南京理工排版校对公司
印　　刷 江苏如皋印刷厂

开　　本 787×1092　1/16　　**版　　次** 2000 年 7 月第 1 版
印　　张 12.25　　**印　　次** 2000 年 7 月第 1 次
字　　数 300 000　　**定　　价** 13.30 元

凡购买高等教育出版社图书，如有缺页、倒页、脱页等质量问题，请在所购图书销售部门联系调换。

前　言

数据结构是计算机应用专业的一门重要的专业基础课。它的主要任务是讨论数据的各种逻辑结构、物理结构以及相关的算法，使学生能够根据实际问题的需要选择合适的数据结构和设计算法，从而具有设计与编制一定规模的软件系统的能力。

本书的第1章讲述了数据结构与算法的基本概念，作为数据结构和算法的基础。第2章介绍了C语言中的数据类型。第3章和第4章讨论了线性数据结构中的线性表、栈、队列和串及其应用。第5章和第6章则介绍了树和图这2种基本的数据结构及应用。第7章和第8章讨论了查找和排序，介绍了各种实现方法。第9章介绍了常用的文件结构。本书采用结合自然语言的类C语言作为算法描述语言，既便于理解和表达，也便于程序实现。

本书的适用对象主要是计算机专业的高职、高专学生，也适用于广大从事计算机应用工作的工程技术人员。因此，本书在内容组织上力求深入浅出、通俗易懂，并且注重实际的程序设计应用。在本书中，我们有针对性地给出了一些示例程序，以求更好地阐明数据结构的原理与方法。

本书由浙江大学计算机系杨枨主编，杨枨、朱晓芸编著。由于编者水平有限，书中疏漏在所难免，敬请读者批评指正。

编　者

2000年5月

第一章　绪　论

计算机是帮助人们对各种数据进行处理的机器。我们在这里所说的数据指的是计算机所能识别、存储和处理的一切数值、字符、图形等信息。

在计算机发展的早期，计算机主要用于科学计算，处理的是数值数据。当时的数据特点是数据元素之间关系简单，数据量小，形式较一致。因此，程序设计工作者注重算法，而不注重对数据的组织、性质、关系的研究。

随着计算机产业的飞速发展和计算机应用的日益普及，计算机已越来越广泛地应用于各种非数值处理问题，其应用已逐步深入到事务管理、工业控制、通信、教育等领域。这些待处理的数据形式多样，量越来越大，关系越来越复杂。要想对这些数据进行有效地处理，就必须要了解数据的性质、组织方式和相互之间的关系。这样才能采用适当的方式进行存储，选用高效的算法进行处理。这些正是数据结构所研究的内容。也就是说，数据结构是一门对计算机所处理的数据的表示、组织和操作进行研究的学科。

在计算机科学中，数据结构与算法是密不可分，缺一不可的。简而言之，算法指的是对计算机求解问题的全过程及具体步骤的描述，计算机程序正是按照算法所描述的步骤对某种结构的数据进行加工处理的。如果没有数据结构，计算机就缺少了处理的“原料”；而没有算法，计算机就缺少了求解问题的方法。因此，著名的计算机科学家 N. Wirth 在论述算法与数据结构之间的关系时，就曾指出：“ 程序 = 数据结构 + 算法 ”。因此，本书在讲述数据的结构关系时，同时对施加于其上的操作及其实现也进行描述。也就是说，数据结构这门课程本身也包含着算法，即计算机在处理具有结构关系的数据时涉及到的基本操作及其实现的方法。

1.1　数据结构的基本概念

数据是程序设计的“原料”及结果。在计算机科学中，凡是在计算机中出现的信息都是数据。数据具有不同的类型，计算机存储和处理的方法也各不相同。在每种高级程序语言中都提供了不同的数据类型，以供程序设计使用。在程序中，每个数据都属于某种类型，它有一定的取值范围和表示形式，以及相对于该类型数据可执行的操作的种类。因此，数据类型指的是一个值的集合以及定义在该集合上的操作的集合。数据类型可分为固有数据类型和用户自定义数据类型 2 种。当使用由语言所提供的数据类型(即固有数据类型)时，由于语言已定义了该类型数据的取值范围、表示形式及所允许的操作，用户只需按规定直接使用。也就是说，固有数据类型可看作是程序设计语言中已实现的数据结构。如果用户在程序设计中自定义数据类型，则需要用户在固有数据类型的基础上，从具体应用中抽象出适当的数学模型，找出描述这个数学模型的新的数据结构，提供从结构的建立到施加在其上的各种操作的实现算法。

简而言之，数据指的是一切可以由计算机识别、存储和处理的描述客观事物的符号，如数值、字符、图像、声音等。

数据元素是用于描述数据的基本单位。数据元素可由数据项组成。数据项是数据表示中不可分割的最小标识单位。

数据对象是具有相同性质的数据元素所组成的集合。一般情况下，数据对象中所包含的数据元素之间具有一定的组织形式和结构关系。数据结构指的就是对数据元素及其相互关系的描述。也就说数据结构包含了2方面的内容，一方面是对数据元素的有限集合即数据对象的描述，另一方面是对数据对象中数据元素的关系的描述。

数据结构中的数据关系是独立于计算机的，也就是说，是一种逻辑关系。数据结构有3种基本类型：线性数据结构、树型数据结构和图型数据结构。

如下表所示的工资表，其逻辑组织结构就是一种线性数据结构：

表 1-1 线性数据结构示例

工 号	姓 名	工 资	奖 金	合 计
001	张 三	500.00	100.00	600.00
002	李 四	600.00	100.00	700.00
003	王 五	550.00	80.00	630.00
004	赵 六	700.00	120.00	820.00
⋮	⋮	⋮	⋮	⋮

在这张工资表中，每一行记录了每名职工的工资收入，它是构造这张工资表的最基本的数据单位，即数据元素，也可称为结点。每一行又可分成5个独立的数据项：工号、姓名、工资、奖金和合计，这5个数据项是不可分的。整个工资表就是一个数据对象。工资表中的每个数据元素之间的组织关系是一种线性结构。即每个数据元素都有且只有1个前驱，每个数据元素也都有且只有1个后继。只有第1个数据元素没有前驱，最后一个数据元素没有后继。

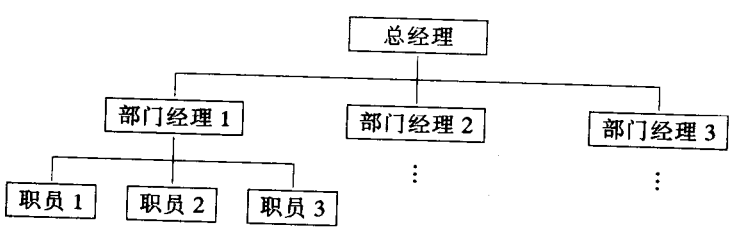


图 1-1 树型数据结构示例

如图1-1所示，是一张某公司的人事关系示意图，它是一种典型的树型数据结构。树型数据结构表现的是一种层次组织关系，组成这种关系的数据元素是结点，每一个结点又有若干个数据项(在结点中称为域)组成，结点与结点之间的联系仅限于上下2层之间的一对多关系，即每一个上层结点可与多个下层结点相关联，但每一个下层结点却只能与一个上层结点相联系。在图1-1中，每一个人员的信息都用一个结点来描述，结点中可包括姓名、年龄、文化程度、职位等信息，这些信息在图中都被省略了。第1层的结点代表的是总经理，他管理着若干个部门经理，而部门经理则处于第2层，每一个部门经理又领导着若干个员工。

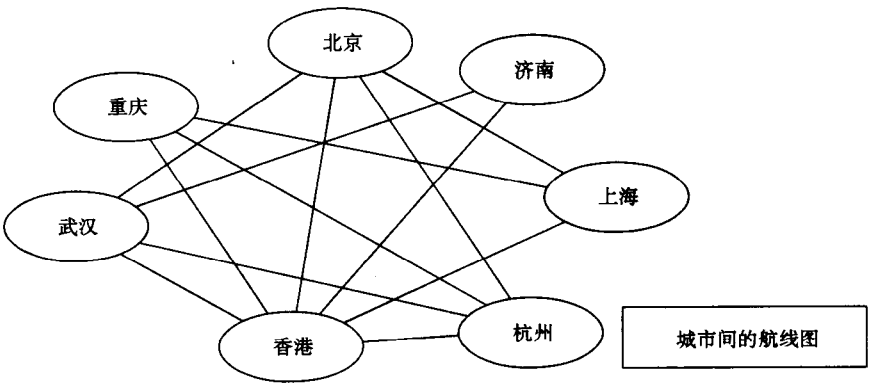


图 1-2 图型数据结构示例

图 1-2 中显示的是国内若干城市之间的航线图，它是一个典型的图型数据结构。组成这种数据结构的数据元素是顶点，在图中每个顶点都可与若干个顶点相连，是一种多对多的关系。在航线图中，对每个城市的描述可用一个顶点来实现，而关于每个城市的基本信息，如城市的名称、机场的位置、航线的多少等可用顶点中的域，即数据项来描述，这些描述在航线图中被省略了。

与数据的逻辑结构相对应的是数据的存储结构，它指的是数据元素及其逻辑关系在计算机存储器中的存储形式，它依赖于计算机，也称之为数据的物理结构。数据的存储结构分 2 类：顺序存储结构与非顺序存储结构。在顺序存储结构中，逻辑上相邻的数据元素在存储器中的物理位置也相邻。也就是说，数据元素之间的逻辑关系可通过数据元素在存储器中的相对位置来体现。在非顺序存储结构中，数据的逻辑关系与其物理位置无关，在逻辑上相邻的数据元素可存放在存储器的任意位置上，数据元素之间的逻辑关系是通过指针来映射的。非顺序存储结构也称为链式存储结构。

1.2 算法的概念与描述

算法描述的是计算机解决问题的具体步骤的序列，这个序列必须是有限的。它有 5 个基本特点：

(1) 确定性：算法的每一个步骤都必须有确定的含义，不能有二义性和模糊性，也就是说对于同样的初始状态应得出相同的结果。

(2) 可行性：算法中的每一步都应在有限的执行时间内完成，否则该算法就不可能得到实现。

(3) 有穷性：算法应在有限的执行步骤里完成。

(4) 输入：每个算法都应具有零个或多个描述问题的初始状态的输入。

(5) 输出：每个算法都应具有至少一个以上的输出作为它的结果。

算法总是施加在特定类型的数据结构之上的。在算法的设计过程中，对数据的各种运算都是定义在数据的逻辑关系之上的，是处于比较抽象层次上的操作，也就是说，主要考虑的是“做什么”的问题。而究竟这些操作如何实现依赖于数据所采用的存储结构。当然，在

我们的算法讨论中，我们对于存储结构的描述是在高级程序设计语言的基础上进行的，而不是像在机器语言中用内存地址来直接描述数据的存储结构。在每种高级程序设计语言中所定义的数据类型都对应于一定的物理结构，实际上就是对数据的存储结构的一种抽象。在本书中我们所讨论的存储结构指的是在 C 语言的数据类型基础上的一种较为抽象的虚拟存储结构。

在本书中，我们采用的是结合自然语言的一种类 C 语言的描述方法来描述算法。用自然语言描述便于人们相互交流和理解，也可以更加突出重点，而不拘泥于局部的细节实现，并且条理性更好，也更便于表达。同时，结合 C 程序设计语言的描述使算法的表达更加准确，也更便于最终的程序实现和对算法性能的分析。

下面是对本书中算法描述语言的简介。

1. 数据类型的说明

包括 C 语言中所有的数据类型，如：整型、实型、字符型、数组、指针、结构、联合等。除了这些以外，还可以定义新的数据类型名来取代已有的数据类型名。

定义形式为：

typedef 已有数据类型名 新的数据类型名；

例如：

typedef int integer;

2. 变量和符号常量

变量的说明形式为：

数据类型 变量名序列；

常量的说明形式为：

#define 符号常量名 常量值

其中，符号常量名常采用具有一定含义的标识符来命名。

例如：

#define TRUE 1

3. 数据运算

主要包括算术运算、关系运算和逻辑运算 3 种。

算术运算符有：+、-、*、/、%、++、--等；

关系运算符有：>、<、==、>=、<=、!=；

逻辑运算符有：!、&&、||；

另外还有指针运算符：*、&，分量运算符：.、→和下标运算符：[]。

4. 赋值语句

赋值语句的形式为：

变量名 = 表达式；

在表达式中，除了可以出现上述数据运算之外，还可以出现常用的数学符号，如：$\sum$、$\cup$、$\cap$、$\in$等。

5. 控制语句

控制语句有以下 2 类。

(1) 选择语句，包括条件语句和多项选择语句。

条件语句的形式为：

```
if(条件表达式)
    语句块;
或:
if(条件表达式)
    语句块 1;
else
    语句块 2;
```

多项选择语句的形式为：

```
switch(表达式){
    case 常量表达式 1:语句块 1;break;
    case 常量表达式 2:语句块 2;break;
    …
    default:语句块 n;
}
```

(2) 循环语句，包括 for 语句、while 语句和 do-while 语句。

for 语句的形式为：

```
for(循环变量初始表达式;终止条件表达式;循环变量修改表达式)
    语句块;
```

while 语句的形式为：

```
while(条件表达式)
    语句块;
```

do-while 语句的形式为：

```
do {
    语句块;
} while(条件表达式);
```

6. 函数的定义、声明与调用

函数的定义形式为：

```
返回类型 函数名(形式参数列表){
    函数定义语句块;
};
```

函数的声明形式为：

```
返回类型 函数名(形式参数列表);
```

函数的调用形式为：

```
函数名(实际参数列表);
```

7. 输入、输出

输入语句的形式为：

```
scanf(格式字符串,输入变量序列);
```

输出语句的形式为：

printf(格式字符串,输出表达式序列);

8. 结束语句

异常结束语句的形式为:

exit(异常代码);

9. 注释

注释语句的形式为:

/* 注释内容 */

1.3 小 结

本章对数据结构及算法中的基本概念进行了介绍,并且用示例对数据结构类型进行了说明。

本书中对算法的描述采用的是一种结合了自然语言和 C 语言特点的类 C 描述语言。本章对这种语言的语法进行了说明。

第二章　C语言中的数据类型

我们把在计算机中进行存储和处理的所有信息都看作是数据。在所有的程序中，都要用到数据，而数据都有一定的表示方法和使用方式。想要正确地进行程序设计，就必须要对数据的组织结构有清晰的了解。正确地理解数据的表示及其使用方法是非常重要的。

数据类型是对数据的存储表示及对该类型数据所能执行操作的抽象。数据类型包括固有数据类型(即程序语言所提供的数据类型)和用户自定义数据类型2种。只有在了解固有数据类型的基础上，用户才有可能根据应用的实际情况构造出适当的自定义数据类型。因此，本章将对C语言中所提供的固有数据类型进行一个概括和总结。C语言中的数据类型分为基本数据类型、派生数据类型和复合数据类型。C语言中的数据有常量和变量之分，常量指的是数据的值在程序运行的过程中不能被改变的量，而变量指的是数据的值在程序运行的过程中可以被改变的量，常量和变量分别属于以上3种类型。

2.1　C语言中数据的存储表示

2.1.1　位

计算机采用二进制格式来存储信息。在二进制系统中采用位(bit)来存储数据，位是在计算机中存储信息的最小单位。一个位有2种可能的取值：0或1，真或假，状态的开或关等。在物理实现上，0或1分别对应着电位的高或低。因为单个位仅仅能存储二值数据，位往往需要将它们组合在一起，以组成一个能容纳更大范围数值的数据单位。

2.1.2　字节

字节是一个8位的组，1个字节可以取256种不同的值。

2 * *8 = 256 (每位可取2个值，一共有8位)

最高有效位							最低有效位	
7	6	5	4	3	2	1	0	二进制位数
128	64	32	16	8	4	2	1	相应的十进制权值

在字节的二进制形式表示中，位按从左到右的顺序排列，最左边的位称为最高有效位(第7位)，最右边的位称为最低有效位(第0位)。字节常用于存储字符，也用于存储十进制数(0～255或－127～＋128)。

2.1.3　二进制编码的十进制数

十进制数(0～9)可用4个二进制位来表示。其对应关系如下所示：

二进制位组合	十进制数值
0000	0
0001	1

0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

对十进制数而言，二进制位的组合 1010、1011、1100、1101、1110 和 1111 是无效的。如果计算机用 1 个字节来存储 1 个 BCD 数字，则这个数字被称为普通的二进制编码的十进制数。其中低 4 位用于表示有效的十进制数值，而高 4 位二进制数空闲不用，其取值可以是全 0 或全 1。

如果每个字节存储 2 个 BCD 数字，则称这 2 个数字为紧缩的二进制编码的十进制数。这种情况经常发生在数据在通信链路上进行传输时。紧缩二进制编码的十进制数可减少用于数据传输的时间，因为每个字节可用于传送 2 个 BCD 数字。

例如，用紧缩的 BCD 格式存储数字 56。

b7	b6	b5	b4	b3	b2	b1	b0	二进制位
0	1	0	1	0	1	1	0	数字 56 对应的紧缩 BCD 格式

高 4 位组存放值 5，与此同时低 4 位组存放值 6。

2.1.4 美国标准信息交换码

美国标准信息交换码(ASCII 码)是用于信息交换的一种标准代码，它用 7 个二进制位的 128 种组合来表示：字符 A～Z(包括大小写)、特殊字符，＜.?:等、数字 0～9 及用于设备控制的特殊控制码。

表 2-1 是对应于每个字符的编码组合(十六进制表示)：

表 2-1 ASCII 编码表

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
01	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
02	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
03	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
04	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
05	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
06	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
07	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

计算机通常是以 8 位为单位来存储信息的，因此 ASCII 码的第 8 位未被用到，通常设为 0。某些系统可以用 8 位来表示图像或不同的语言符号(如希腊的字符)。控制码多数用于通信和打印机。在 ASCII 键盘上按住[Ctrl](控制)键并且按下另一个键 (A～Z, {, \, },

^, <-)即可产生。

例如,将字符串"Hello."以 ASCII 码表示(十六进制)。

H	e	l	l	o	.	字符
48	65	6C	6C	6F	2E	相应的 ASCII 码

因此,该字符串由以下字节序列表示:

48 65 6C 6C 6F 2E

2.2 基本数据类型

本节对 C 语言中所使用的各种基本数据类型进行介绍。C 语言中的基本数据类型包括整数类型(int)、字符类型(char)、实数类型(float 和 double)3 种类型,它们由硬件系统直接支持实现。我们将说明数据如何被存储、访问及其在程序中的典型使用方法。

2.2.1 字符

字符用于表示非数字符号的数据,如:英文字母(大小写)、#、$、%、* 等。

每个字符以 8 位存储,1 个字节可表达 256 种不同的字符。

在 C 语言中,字符常量是以单引号括起来的字符,如'a'。计算机系统中,通常用 ASCII 码来存储字符,即并不是把字符本身存放到存储器中,而是将该字符的相应 ASCII 码值存放到存储器中。如'a'在存储器中是以 ASCII 码值 97(十进制)来存放的。

字符变量以如下方式定义和使用:

```
char  ch1, ch2;
ch1 = 'x';
ch2 = 'y';
```

上述例子表明变量 ch1、ch2 是字符类型,存储器分别为它们分配 8 位存储空间以存储它们的值。后面的 2 条赋值语句分别把字符'x'、'y'赋给字符变量 ch1 和 ch2,也就是把'x'和'y'相应的 ASCII 码值 120 和 121 放到存储单元中。

2.2.2 字符串

字符串是字符的序列。每个字符被相继地存储,分别占用存储器的 8 位存储空间。字符串常量是用一对双引号括起来的字符序列。字符串常量"Hello"将在存储器中存为:48 65 6C 6C 6F 0。

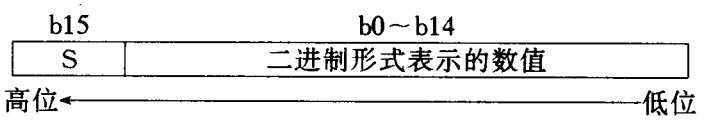
C 语言中,在每个字符串的结尾都加上结束标志'\0'。由于'\0'的 ASCII 码值为 0,所以"Hello"在存储器中的长度为 6 个字节,而不是 5 个字节。在 C 语言中,没有专门的字符串变量,字符串如果需要存放在变量中,就需要用字符数组来存放。

2.2.3 整数

数字信息用 ASCII 码是不能有效地进行存储的。假设用 ASCII 码存储数字 123 456,这就需要占用 6 个字节的存储空间,并且很难表示数值的符号为正或负(尽管我们可在它之

前加 +或 - 符号)。为了有效地存储数字,必须采用另外的编码方案。

整数的长度有 16 位(整型 int、短整型 short int)和 32 位(长整型 long int)之分,每种长度中又有有符号和无符号之分。下面以 16 位整数为例来讨论其表示。

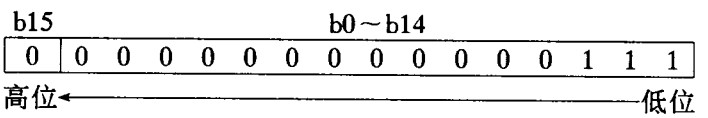


符号位(第 15 位)表示该数字是正还是负,逻辑 1 表示负,逻辑 0 表示正。数字的值以二进制形式存储在 0～14 位中。例如,整数 +7 的二进制表示为:

(1) 符号位是 0。

(2) 十进制值 7 对应二进制数 000 0000 0000 0111。

因此存储在存储器里的整数 +7 被表示为:



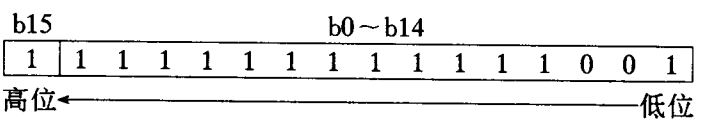
在表示负数时符号位为 1,其余各位是由该负数的绝对值的二进制表示的位序列求反之后的结果值加 1 后得到的二进制位序列。例如,整数 -7 的二进制表示为:

(1) 符号位是 1。

(2) 十进制数 -7 的绝对值的二进制表示是二进制数 000 0000 0000 0111。

(3) -7 的绝对值的二进制表示的位序列求反之后的结果值加 1 后得到的二进制位序列为:111 1111 1111 1001

因此存储在存储器里的整数 -7 被表示为:



整型常量有 3 种表示方法:十进制、八进制和十六进制。

(1) 十进制:如:12、-34。

(2) 八进制:以 0 开头的数,如:012 表示八进制数 12,即十进制的 10。-011 表示八进制数 -11,即十进制的 -9。

(3) 十六进制:以 0x 开头的数,如:0x12 表示十六制数 12,即十进制的 18。-0x11 表示十六进制数 -11,即十进制的 -17。

C 语言的整数类型及其值域如下所示:

类型名	位数	值域
short [int]	16	-32 768～32 767
signed short [int]	16	-32 768～32 767
unsigned short [int]	16	0～65 535
int	16	-32 768～32 767
signed int	16	-32 768～32 767
unsigned int	16	0～65 535

long [int]	32	-2, 147, 483, 648～2, 147, 483, 647
signed long [int]	32	-2, 147, 483, 648～2, 147, 483, 647
unsigned long [int]	32	0～4, 294, 967, 295

在C语言的程序中使用的整型常量被解释成为能满足存储该数的最小值域的数据类型。如果无后缀整型常量的值在-32 768～32 767范围内,就认为它是int型;如果其值超过了该范围,而在-2, 147, 483, 648～2, 147, 483, 647范围之内,就认为它是long int型。如果在整型常量后面带上后缀"L",则作为长整型long int,如1234L;如果带上后缀"U",就作为无符号整型,如123UL就表示无符号长整型数123。

在C语言中,整型变量的定义和使用方法如下:

```
int i;
short s;
long l;
unsigned int ui;
unsigned short us;
unsigned long ul;
ui = 12;
us = 34;
ul = 1234L;
i = - 12;
s = - 34;
l = - 1234L;
```

在C语言中,整型常量无unsigned型,但一个非负的整型常量可以赋给unsigned型变量。另外,由于字符在存储器中存放的是ASCII码值,这种存储形式与整数的存储形式相类似,所以在C语言中,整型和字符型可以互用。如果将一个整数存放在一个字符变量中,则这个变量中存放的是与此整数值作为ASCII码值相对应的字符,C程序将把该字符作为处理的对象。如果在一个整型变量中存放了一个字符,那么该变量保存的字符所对应的ASCII码值。如:

```
int i;
char ch;
i = 'A';
ch = 65;
```

在i中存放的是字符A所对应的ASCII码值65,而ch中存放的是与ASCII码值65相对应的字符A。

2.2.4 实数

实数也称为浮点数。在C语言中,实数分为单精度(float)、双精度(double)和长精度(long double)3种。实数有2种表示方法:

(1) 十进制数形式。如:1.234。

(2) 指数形式。如:1.2E3, -2.1e-3。E(或e)之前必须有数字,且其后的指数必须为整数。

C语言的实数类型及其值域如下所示：

类型	位数	值域
float	32	3.4E－38 ～ 3.4E＋38
double	64	1.7E－308 ～ 1.7E＋308
long double	80	3.4E－4 932 ～ 1.1E＋4 932

实数的存储采用浮点的方法，这需要把实数分成2部分：指数和尾数。其中指数表示2的幂次数，而尾数表示位于0和1之间的小数值。

例如，实数12.5的存储表示。

(1) 实数首先被转换为如下表示形式：

$2^n * 0.xxxxxx$

其中，n 表示指数部分，0.xxxxxx 则是尾数部分。则：

12.5/2 = 6.25

6.25/2 = 3.125

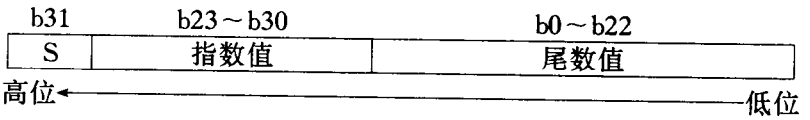
3.125/2 = 1.562 5

1.562 5/2 = 0.781 25

即：$12.5 = 2^4 * 0.781\ 25$。

计算机工业为浮点数的存储制定了标准，即IEEE 754标准，并且使用存储器的32位(为单精度)、64位(为双精度)或80位(为长精度)。

单精度格式如下所示：



其中0～22位共23位表示尾数值，23～30位共8位表示指数值，第31位是符号位。当尾数值为负时，符号位为1；当尾数值为正时，符号位为0。指数值按其与127之间的偏移量的方式来存储，而尾数值则采用一种二进制的编码方式来存储。

(2) 指数值的存储方法：存储的是指数值与127之间的余量，也就是说，在存储的时候，127被加到指数值上，在读取时，需减去127才得到真正的指数值。

12.5的指数值在存储时应为：

4 + 127 = 131(对应的二进制数为10000011)

因为12.5对应的尾数值为正，因此符号位S为0。

(3) 尾数的存储方法：尾数首先被表示成$\sum 2^n$，其中 n 为负数。如：

第1位尾数 $= 2^{-1} = 0.5$，

第2位尾数 $= 2^{-2} = 0.25$，

第3位尾数 $= 2^{-3} = 0.125$，

第4位尾数 $= 2^{-4} = 0.062\ 5$，

第5位尾数 $= 2^{-5} = 0.031\ 25$，

……

直到2^{-23}。存储时按从左到右的方向存放。即第1位尾数存放在二进制表示的第22

位，第2位尾数存放在二进制表示的第21位……第23位尾数存放在二进制表示的第0位。

12.5的尾数0.781 25 = 0.5 + 0.25 + 0.031 25 = $1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+0\times2^{-4}+1\times2^{-5}+0\times2^{-6}...0\times2^{-23}$。因此，尾数0.781 25存储时的二进制表示为：第1，2，5位为1，其余各位都为0，即：

11001000000000000000000

(4) 尾数的规范化和指数值的调整。

为了扩大实数的表示范围，还需要对尾数进行规范化，即对尾数的二进制表示进行左移，直到它的二进制表示中的第1位1被移走为止(每左移一次，相当于对尾数乘2，其指数值就应被减1)。这样，规范化以后的尾数的第1位总是1，可以由硬件自动插入，而不需要在二进制表示中保留。也就是说，它是尾数的隐含的第24位，其值永远为1。

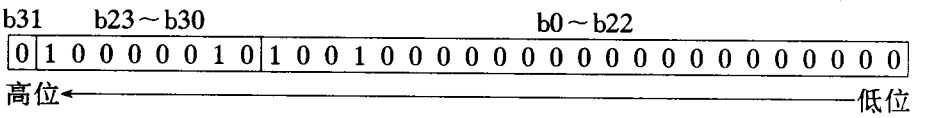
经过规范化后，尾数0.781 25的二进制表示变为：

10010000000000000000000

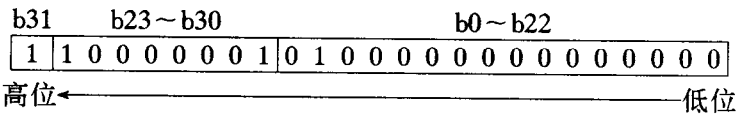
指数值也被调整为：

131 - 1 = 130(对应的二进制数为10000010)

最后可得实数12.5的存储表示为：



例如：从如下的32位浮点表示来读出对应实数的十进制数值：



符号位为1，因此可知这是一个负数。

指数值为：

129 - 127 = 2

尾数值为：

1.0 + 0.25 = 1.25

1.0来自尾数的规范化过程中被移走的第24位，因此，实数的值为：

$-1.25*2^2=-5.0$

C语言中对实数变量的定义和使用方式如下所示：

```
float r1, r2;
r1 = 3.141 59;
r2 = r1 * 2.0;
double d1, d2;
d1 = 3.141 592 675;
d2 = d1 * r1 ;
```