第 5 章

数 据 类 型

数据是程序处理的对象，数据可以依据自身的特点分成不同的类型。计算机内部只有二进制比特位，取值只能是 0 或 1。计算机科学家设计了多种数据类型，用 0 和 1 表示各种不同类型的数据。

5.1 数据类型概述

使用 C++ 语言编写程序时，需要用到各种变量来存储各种信息。变量名保留的是它所存储的值的内存位置。当定义一个变量时，就会在内存中分配一定的存储空间。C++ 提供了种类丰富的内置数据类型和用户自定义数据类型，如图 5-1 所示。

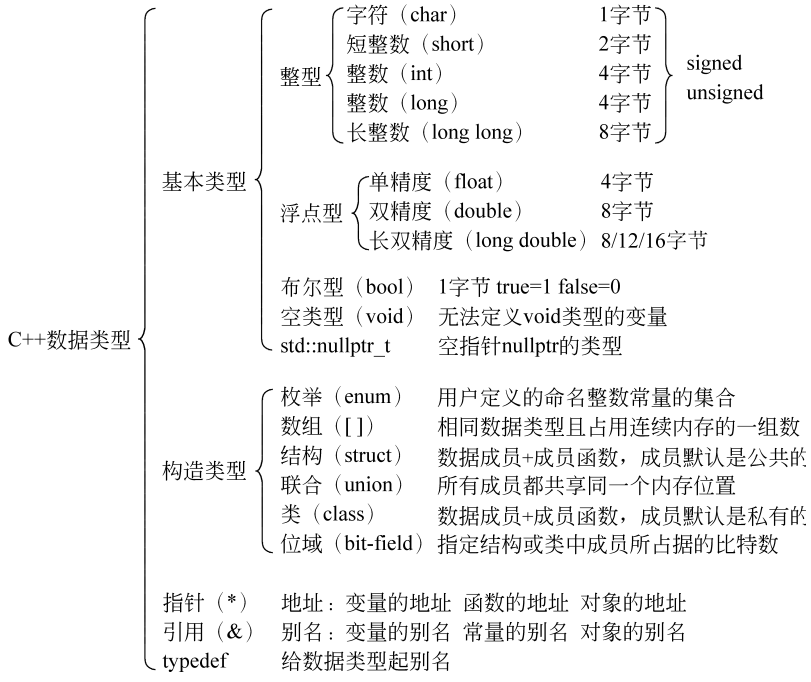

图 5-1 C++ 数据类型

C++ 提供的基本数据类型有整型、浮点型、布尔型、空类型、空指针类型。用户自定义的构造类型有枚举、数组、结构、联合、类、位域。C++ 提供了不同长度的有符号整数和无符号整数。C++ 浮点数类型使用 IEEE-754 标准表示实数的近似值。

指针是一个变量，其值为另一个变量的地址，即内存位置的直接地址。不同类型的指针变量保存不同类型的内存地址。将整数变量的地址取出来，可以保存在整数指针中；将整数指针变量的地址取出来，可以保存在指向整数指针的指针中；将函数的地址取出来，可以保存在函数指针中；将对象的地址取出来，可以保存在对象指针中。

引用变量是一个别名，也就是说，它是某个已存在变量的另一个名字。一旦把引用初始化为某个变量的别名，就可以使用该引用来访问那个变量。常引用可以引用一个常量。对象的引用是一个对象的别名。

C++ 中，还可以使用 typedef 关键词为已有的数据类型定义别名，通过给出有具体意义的别名，可以使程序中的类型使用更明确。

5.2 指针和引用

指针(Pointer)是一个整数，保存的是另外一个变量的地址，即它指向内存中的另一个地方。引用(Reference)是某一个变量的一个别名，对引用的操作与对原变量直接操作完全一样，引用必须在定义时初始化。

如果想表示指针不指向任何位置，C++ 提供空指针 nullptr，它是 std::nullptr_t 类型的常量。C++ 兼容 C 语言的空指针 NULL。nullptr 和 NULL 的区别：nullprt 是指针类型，而 NULL 是整数类型。nullptr 和 NULL 都等价于 0。

例 5.1 理解指针。

```
//Pointer.cpp
#include <iostream>
using namespace std;
int main(int argc, char** argv){
    int n = 100;
    int *p = &n; //& 取地址 * 定义指针
    *p = 200;    //* 解引用符
    cout << n << " " << *p << endl; //200 200
    return 0;
}
```

【运行结果】

```
200 200
```

【代码解读】

程序在主函数中定义了整数变量 n，并初始化为 100。“int *p”是定义了一个整数指针变量 p，可用于保存整数变量的地址。p 是一个正在被定义的变量，此时 * 的作用是定义指针变量。作为一个已经存在的变量 n，“&n”的含义是取变量 n 的内存地址，此时 & 是取地址运算符。程序取变量 n 的地址，作为指针变量 p 的初始化值。语句“*p=200;”中，*后面是一个已经存在的指针变量，此时 * 是解引用运算符，即通过整数指针 p 间接访问它所指向的变量 n，结果是将变量 n 赋值为 200。

例 5.2 理解引用。

```
//Reference.cpp
#include <iostream>
```

```
using namespace std;
int main(int argc, char** argv){
    int a = 300;
    int &b = a; //& 用来定义引用, b 是 a 的别名, 定义引用时必须初始化
    b = 500;
    cout << a << " " << b << endl; //500 500
    return 0;
}
```

【运行结果】

```
500 500
```

【代码解读】

程序在主函数中定义了整数变量 a,并初始化为 300。语句"int &b=a;"定义了引用变量 b,将 a 作为引用的初始化值,b 成为 a 的别名。b 是一个正在被定义的变量,此时 & 的作用是定义引用变量。语句"b=500;"执行后,变量 a 的值也被修改成了 500,因为 b 就是 a。

例 5.3 理解指向指针的指针。

```
//PointerPointer.cpp
#include <cstdio>
#include <iostream>
using namespace std;
int main(int argc, char** argv){
    int n = 100;
    int *p = &n;            //取变量 n 的地址,作为指针变量 p 的初值
    int **q = &p;           //取指针 p 的地址, 作为指向指针的指针 q 的初值
    **q = 600;              //2 次解引用访问 n * q 得到 p**q 得到 n
    cout << n << " " << *p << " " << **q << endl; //600 600 600
    cout << sizeof(p) << " " << sizeof(q) << endl; //8 8
    //p 是整数指针,类型是 int*,指向 n
    printf("memory address of n: %p %p\n", p, &n);
    //q 是指向整数指针的指针,类型是 int**,指向 p
    printf("memory address of p: %p %p\n", q, &p);
    return 0;
}
```

【运行结果】

```
600 600 600
8 8
memory address of n: 000000F126AFF7D4 000000F126AFF7D4
memory address of p: 000000F126AFF7F8 000000F126AFF7F8
```

【代码解读】

程序在主函数中定义了整数变量 n,并初始化为 100;取整数变量 n 的地址,作为整数指针变量 p 的初始化值;取指针 p 的地址,作为指向整数指针的指针 q 的初始化值。指针 p 的数据类型是 int*,指针 q 的数据类型是 int**。

对于已经存在的指针 q，* q 是解一次引用得到 p，**q 是解两次引用得到 n。因此，语句“**q＝600；”将变量 n 的值修改为 600。

输出语句中，n、* p、**q 访问的都是整数变量 n，因此输出三个 600。

指针变量自身是一个无符号整数，占用的字节数取决于程序编译成 32 位应用，还是编译成 64 位应用。如果程序编译成 32 位应用，每个指针变量占用 4 字节内存。如果编译成 64 位应用，每个指针变量占用 8 字节内存。指针变量的值是一个内存地址，是无符号的整数，可以使用格式控制符 %p 输出这个值。

如果将程序编译成 X64 指令系统 CPU 的应用程序，则程序执行了 main 函数中前 4 行代码后，n、p、q 之间的关系如图 5-2 所示。n 是一个有符号整数变量，占 4 字节。X64 指令系统采用小端(Little Endian)格式存放多字节数据。Little Endian 格式是在低字节中存放数据的低位有效字节。变量 n 的 4 字节的值依次是 58、02、00、00，理解这个数的时候要从高到低按字节倒着看才行，即 00、00、02、58。0x00000258 就是十进制数 600。

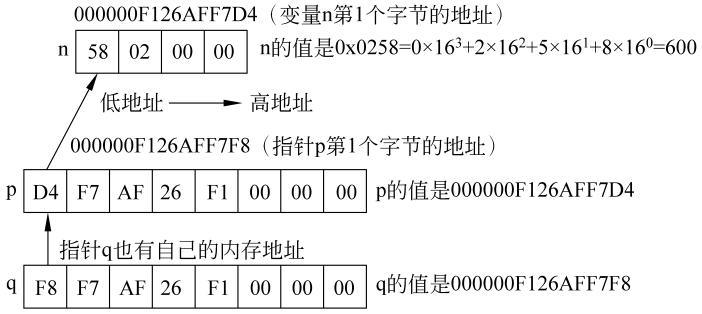

图 5-2　指向指针的指针

变量 n 在内存中的地址是 000000F126AFF7D4。将变量 n 的地址取出来保存在整数指针变量 p 里面。指针 p 的 8 字节的值依次是 D4、F7、AF、26、F1、00、00、00，理解这个数的时候也需要从高到低按字节倒着看。

指针 p 作为一个变量，保存在内存中，也是有地址的：000000F126AFF7F8。取指针 p 的地址保存在变量 q 中，则变量 q 是一个指向指针的指针。

5.3　ASCII 码

ASCII(American Standard Code for Information Interchange)即美国信息交换标准码，是一套标准的单字节英文编码方案。它是由美国国家标准学会(American National Standard Institute，ANSI)制定的，后来它被国际标准化组织(International Organization for Standardization，ISO)定为国际标准，称为 ISO 646 标准。

标准 ASCII 码使用 7 位二进制数来表示 128 个字符，如表 5-1 所示。它使用 7 位二进制数来表示所有的大写和小写字母、数字 0～9、标点符号，以及特殊控制字符，取值范围是 0000 0000～0111 1111(0～127)，其中 0～31 和 127 是控制字符或通信专用字符，其余为可显示字符。扩展 ASCII 码是最高位为 1 的 8 位二进制数，用来表示特殊符号字符、外来语字母和图形符号，取值范围是 1000 0000～1111 1111(128～255)。

表 5-1　标准 ASCII 码

低位		高位																	
		ASCII 码控制字符							ASCII 码打印字符										
		0000				0001			0010		0011		0100		0101		0110		0111	
		0				1			2		3		4		5		6		7	
		值	代码	转义	字符解释	值	代码	字符解释	值	字符	值	字符	值	字符	值	字符	值	字符	值	字符
0000	0	0	NUL	\0	空字符	16	DLE	数据链路转义	32	空格	48	0	64	@	80	P	96	`	112	p
0001	1	1	SOH		标题开始	17	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q
0010	2	2	STX		正文开始	18	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r
0011	3	3	ETX		正文结束	19	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s
0100	4	4	EOT		传输结束	20	DC4	设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t
0101	5	5	ENQ		查询请求	21	NAK	否定应答	37	%	53	5	69	E	85	U	101	e	117	u
0110	6	6	ACK		肯定应答	22	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v
0111	7	7	BEL	\a	响铃	23	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w
1000	8	8	BS	\b	退格	24	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x
1001	9	9	HT	\t	横向制表	25	EM	介质末端	41	)	57	9	73	I	89	Y	105	i	121	y
1010	A	10	LF	\n	换行	26	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z
1011	B	11	VT	\v	纵向制表	27	ESC	取消	43	+	59	;	75	K	91	[	107	k	123	{
1100	C	12	FF	\f	换页	28	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|
1101	D	13	CR	\r	回车	29	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}
1110	E	14	SO		移出	30	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~
1111	F	15	SI		移入	31	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	DEL

例 5.4 标准 ASCII 码中的可显示字符。

```
//ASCII.cpp
#include <stdio.h>
int main(int argc, char** argv){
    for(int i=32; i<=126; i++){
        printf("%3u %02X %c\n", i, i, i);
    }
    return 0;
}
```

【代码解读】

程序将整数 32～126 以三种形式输出：无符号十进制整数、十六进制数、字符。%3u 表示占 3 个字符宽度的无符号十进制整数。%02X 表示占 2 个字符宽度的无符号大写十六进制整数，不足 2 位填 0。%c 表示一个字符。十进制数 32～126 对应的二进制数是 0010 0000～0111 1110，对应的十六进制数是 20～7E，这个范围是 ASCII 码的可显示字符。比如，32 是空格，48 是零，126 是波浪线。

```
0010 0000 =  32 = 0x20 = ' '  空格
0011 0000 =  48 = 0x30 = '0'  零
0111 1110 = 126 = 0x7E = '~'  波浪线
```

例 5.5 英文字母、数字、换行的 ASCII 码。

```
//Character.cpp
#include <iostream>
using namespace std;
int main(int argc, char** argv){
    //英文字母的 ASCII 编码
    if('A'==65){                          //0100 0001
        cout << "A 的编码是 65\n";         //会执行
    }
    cout << (char)97 <<endl;              //'a'=97=0110 0001

    //数字的 ASCII 编码
    char ch = 48;                         //0011 0000
    cout << ch << endl;                   //'0'=48
    ch = '3';                             //0011 0011 字符 3
    if(51==ch){
        cout << "字符 3 和整数 51 是相等的\n";
    }
    int m = ch-'0';                       //减去 48
    cout << "m=" << m << endl;            //数值 3

    //10 是换行符，把 10 作为字符输出会导致换行
    ch = 10;                              //00001010
    cout << ch << ch <<"2LFs\n";          //先做 2 次换行
    cout << (10 == '\n');                 //1
    return 0;
}
```

【运行结果】

```
A的编码是65
a
0
字符3和整数51是相等的
m=3
[空行]
[空行]
2LFs
1
```

【代码解读】

大写字母A的编码是0100 0001,和整数65相等。小写字母a的编码等于十进制的97,因此将97强制转换成字符型输出,结果是输出小写字母a。字符'0'的编码是48,因此将48赋值给字符型变量ch,然后输出ch,结果是输出0。字符'3'和整数51是相等的,字符'3'减去字符'0'等价于51－48,得到数值3。换行的编码是0000 1010,等于十进制的10,将10作为字符输出,结果就是换行。换行是不可显示字符,需要转义表示,'\n'表示换行。流操纵子endl是一个函数,它输出一个换行符'\n'并刷新输出缓冲区。

ASCII码中,0～31和127是控制字符,如10是换行,13是回车。内存中只有0和1,计算机使用比特串表示各种类型的数据,也就是编码。控制字符也使用数值进行编码。

```
0000 1010 =  10 = 0x0A = '\n' 换行(Line Feed,LF)
0000 1101 =  13 = 0x0D = '\r' 回车(Carriage Return,CR)
```

CR最原始的含义是将针式打印机的打印头移动到最左边,即一行的开始(行首),并没有移到下一行的意思。LF直译为“给打印机喂一行”,是移动打印头到下一行的意思。Windows中的文本文件的换行符是"\r\n",而Linux中是"\n",即Windows文本文件用字符CR和字符LF表示换行,Linux只用LF表示换行。

例5.6 将字符数组中的数字转换为整数。

```
//StrToInt.cpp
#include <iostream>
using namespace std;
int main(int argc, char** argv){
    char data[100] = "5678";        //字符数组
    char *p=data;                   //字符指针p指向第一个字符'5'
    int  s = 0;                     //转换后的整数
    while(*p){                      //同*p!='\0',遇到0的时候停止
        s = s*10 + (*p-'0');        //'0'等于48
        p++;                        //指针指向下一个字符
    }
    cout << "s = " << s << endl;
    cout << "2*s = " << 2*s << endl;
    return 0;
}
```

【运行结果】

```
s = 5678
2 * s = 11356
```

【代码解读】

变量 data 是一个拥有 100 个字符的字符数组，里面保存的是 53、54、55、56、0，后面还有 95 字节的 0。53 是字符'5'的 ASCII 码，54 是字符'6'的 ASCII 码，55 是字符'7'的 ASCII 码，56 是字符'8'的 ASCII 码。

C 语言中，空字符 NUL 被定义为字符串的结束符。空字符是 ASCII 码中的第一个字符，编码是 0000 0000，使用转义字符'\0'表示，和整数 0 相等。常量字符串"5678"的末尾有一个隐含的空字符'\0'，也就是一字节中每个比特都是 0 的字符。

C 的语法允许在定义字符数组时使用常量字符串初始化字符数组。程序中使用常量字符串"5678"初始化字符数组 data，编译器是将 53、54、55、56 和隐含的'\0'复制到了字符数组中前 5 个元素的位置。

数组作为局部变量定义时，如果没有初始化，数据是不确定的。如果程序给出了部分初值，编译器会将剩余的空间清零。data 是 100 个字符的字符数组，前 5 个字符使用常量字符串"5678"初始化，后面 95 个字符编译器负责清零。"5678"是位于全局数据区的常量字符串，是不可以被修改的。data 是主函数内部定义的局部变量，是一个拥有 100 个字符的数组，是可以被修改的。局部变量 data 占用的内存空间在栈上分配，每个线程拥有一个栈。

要计算 2 * 5678 的值，需要将字符串转换为整数。如图 5-3 所示，指针 p 首先指向字符数组中的第一个字符，然后逐个指向后续的每一个字符。循环条件是 * p，其中 * 是解引用运算符，指针变量 p 的值是某个字符的内存地址，* p 表示取 p 指向的字符。while 循环共执行了 5 次，每次执行的判断分别是 while(53)、while(54)、while(55)、while(56)、while(0)，当判断到while(0)时循环结束。

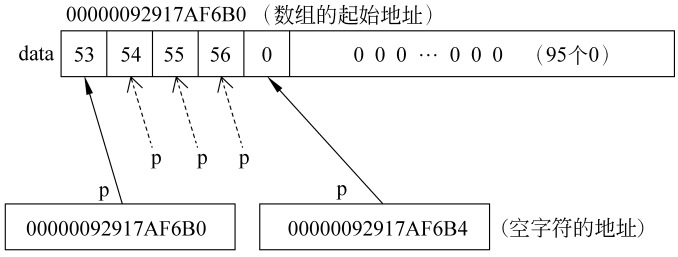

图 5-3 遍历字符数组中的字符串

在循环内部，* p－'0'将 p 指向的字符转换成对应的数值，并将得到的数值作为整数 s 新的个位。p＋＋将指针 p 指向数组的下一个元素。循环结束后，s 的值就变成了整数 5678。

5.4 整 数

整数可以表示某个特定范围的所有整数。int 类型是默认的基本整数类型，它默认是有符号整数。有符号整数使用二进制补码表示负数、零和正数，加上 signed 修饰符也表示有

符号整数。unsigned 修饰符表示只能保存非负值的无符号整数。

大小修饰符指定整数类型的宽度，即位数。C++ 支持 short、long 和 long long 修饰符。short 类型至少 16 位，long 类型至少 32 位，long long 类型至少 64 位。

使用 signed、unsigned 或大小修饰符时，可以省略 int 关键字。修饰符和 int 类型（如果存在）可以按任何顺序显示。例如，short unsigned 和 unsigned int short 指同一类型。编译器将以下类型组视为同义词。

- short、short int、signed short、signed short int
- unsigned short、unsigned short int
- int、signed、signed int
- unsigned、unsigned int
- long、long int、signed long、signed long int
- unsigned long、unsigned long int
- long long、long long int、signed long long、signed long long int
- unsigned long long、unsigned long long int

在 C++ 主流编译器中，各种整数类型的表示范围如表 5-2 所示。

表 5-2　整数类型

类　型	说　明	字节数	表示范围	常量举例
char	字符	1	$-2^7 \sim 2^7-1$	'A'
unsigned char	无符号字符	1	$0 \sim 2^8-1$	'A'
short	短整数	2	$-2^{15} \sim 2^{15}-1$	(short)100
unsigned short	无符号短整数	2	$0 \sim 2^{16}-1$	(unsigned short)100
int	整数	4	$-2^{31} \sim 2^{31}-1$	−100
unsigned int	无符号整数	4	$0 \sim 2^{32}-1$	100
long	整数	4	$-2^{31} \sim 2^{31}-1$	−100L
unsigned long	无符号整数	4	$0 \sim 2^{32}-1$	100L
long long	长整数	8	$-2^{63} \sim 2^{63}-1$	−200LL
unsigned long long	无符号长整数	8	$0 \sim 2^{64}-1$	200LL

整数的字面值（Literal）是整数的具体表示形式，C++ 支持十进制（Decimal）、八进制（Octal）、十六进制（Hexadecimal）的书写形式。编译器负责转换成二进制，计算机内部只有二进制。编程时，用非 0 数字开头的数字序列表示十进制数，0 开头的数字序列表示八进制数，0X 或 0x 开头的数字序列表示十六进制数。A、B、C、D、E、F 是十六进制中的一位数，A 等于十，B 等于十一，C 等于十二，D 等于十三，E 等于十四，F 等于十五。F+1 的值是 0x10，即十六进制中的十六。整数还可以加后缀，后缀有 L、l、U、u 四种。L 和 l 表示长整数，U 和 u 表示无符号整数。如“023l”（小写 L）是八进制长整数，等于十进制数 19。

char 是为处理 ASCII 码而设计的，只有 1 字节（8bit），表示范围是 −128～+127，其中非负整数部分 0～127 表示全部标准 ASCII 码。unsigned char 的表示范围是 0～255。C++ 使用单引号表示一个字符常量，不可显示字符和部分可显示字符需要转义表示。char 和

unsigned char 都是整数的子类型，可以作为整数参与运算。例如：

```
char ch = 'A' + 2;
cout << ch << endl; //C
cout << (int)ch << endl; //67
```

新式 C++ 增加了更多的字符类型：wchar_t、char8_t、char16_t、char32_t。char 和 wchar_t 的字符编码是不确定的，而 char8_t、char16_t、char32_t 的编码假设是 UTF-8、UTF-16 和 UTF-32。

- wchar_t ：宽字符类型，每个 wchar_t 类型占 2 字节，16 位宽。
- char8_t：UTF-8 字符的类型。
- char16_t：UTF-16 字符的类型。
- char32_t：UTF-32 字符类型。

例 5.7 －1 等于最大整数。

```
//integer.cpp
#include <cstdio>
int main(int argc, char** argv){
    int x = 0xFFFFFFFF;
    std::printf("x = %d\n", x);     //十进制有符号整数
    std::printf("x = %u\n", x);     //十进制无符号整数
    std::printf("x = %x\n", x);     //十六进制 小写 a~f
    std::printf("x = %X\n", x);     //十六进制 大写 A~F
    return 0;
}
```

【运行结果】

```
x = -1
x = 4294967295
x = ffffffff
x = FFFFFFFF
```

【代码解读】

内存中只有 0 和 1，值是多少取决于怎么去理解它。整数变量 x 的值初始化为 0xFFFFFFFF，变量 x 的 4 个字节的内容是 1111 1111 1111 1111 1111 1111 1111 1111。

按 int 类型理解，x 是－1；按 unsigned int 类型理解，x 是 $2^{32}-1$，即 4 294 967 295。有符号整数采用二进制补码表示，零和正数的补码是其自身，负数的补码是绝对值相等的正数按位取反再加 1。例如，求－1 的补码：

```
0000 0000 0000 0000 0000 0000 0000 0001     －1 的绝对值＋1
1111 1111 1111 1111 1111 1111 1111 1110     ＋1 按位取反
                                    ＋1     加 1
1111 1111 1111 1111 1111 1111 1111 1111     －1 的补码
```

例 5.8 －128 等于＋128。

```
//neg128.cpp
#include <cstdio>
using namespace std;
```