第2章

基本数据类型

两个相互矛盾方面的共存、斗争以及融合成一个新范畴，就是辩证运动的实质。

—— 恩格斯

机器语言指令和汇编语言指令一样，把对数据的基本操作等同于最简单的机械动作，使得对一个像求和一样的简单计算，都需要几条机器指令才能完成。这不便于程序的编写、阅读、修改和维护。我们把这种基于硬件的程序语言称为低级语言。要克服低级语言的局限性，就要超越计算机硬件结构，建立面向算法处理对象的程序语言。

我们知道，计算机最初是用于数值计算的。而数值计算的基本操作是按类型划分的。如果用操作符和操作对象构成操作符指令，那么算法都可以表示为操作符指令的有穷序列。这种基于类型的存储模式便是高级程序语言。

2.1 变量与字面值常量

计算机处理的数据都是有存储空间的。在C++语言中，每个数据都有类型，类型决定了空间的大小、格式以及对空间可以实施的基本操作。如果空间中的数据不能修改，这个空间就称为常量空间，简称常量。如果空间中的数据可以修改，这个空间就称为对象空间，简称对象。具有名字的对象称为变量。

往空间中存入数据称为写入或赋值，从空间中提取数据称为读取或取值。赋值和取值统称为访问。如果是用空间的名称来访问数据，称为直接访问。

出现最多的常量是字面值常量，简称字面量。所谓字面值常量，就是一种常量空间的名称，这种名称的特点是：它由常量数值和表示该数值类型的后缀字符或界限符等组成。例如：52388L是长整型字面量，其中52388是常量值，L是代表长整型的后缀。普通整型字面量没有表示类型的后缀，它是默认的。例如：345是普通整型字面量。

另一类常量是符号常量。所谓符号常量就是用标识符或名称代表的常量，在以后的学习中我们将不断引入各种符号常量。

一般通过定义来得到变量。变量定义需要指明类型和变量空间的名称。类型规定了变量空间大小、格式和基本操作。变量空间名称即变量名是用户根据命名规则指定的（命名规则见附录A）。变量定义的格式为：

```
类型标识符  变量名1 [ , 变量名2 , 变量名3 , … , 变量名n ] ;
```

例如，定义两个普通整型变量m及n和一个长整型变量k，代码如下：

```
int  m,n;
long  k;
```

这是两条变量定义语句。语句是C++程序的基本组成单位，每条语句以分号结束。其中int是普通整型标识符，long是长整型标识符。

在32位地址机上，普通整型（简称整型）和长整型空间都占4个字节。通过取址操作符“&”可以得到变量空间的地址，即首字节地址，格式为：

```
& 变量名
```

利用操作符“sizeof”可以计算变量空间大小，具体格式为：

```
sizeof(变量或常量名)
```

或

```
sizeof 变量名或常量名
```

或

```
sizeof(类型标识符)
```

例如：sizeof(n)、sizeof(345)和sizeof(int)，它们的值都是4。

对字面量不能寻址，例如&345是非法的，但是可以通过操作符sizeof测试它的空间大小。

变量需要赋值，赋值操作一般通过赋值运算符“=”来实现。赋值运算符有左右两个操作数，分别称为左元和右元。赋值运算是从右元所标识的空间取值，给左元所标识的空间赋值。例如：

```
m=345 ;
n=m;
k=52388L;
```

图2-1 变量和字面值常量

这是三个赋值语句。第一条语句是从字面量345中取值，赋给变量m；第二条语句是从变量m中取值，赋给变量n；第三条语句是从字面量52388L中取值，赋给变量k（见图2-1，图中的矩形框表示变量空间，阴影矩形框表示字面量空间）。

可以在定义变量的同时给变量赋值，称为变量初始化。例如：

```
int m=345;
long k=52388L;
```

可以在一条语句中对几个变量初始化，也可以定义和初始化相间。例如：

```
int m=345,n=345;
long x, y=345;
```

初始化属于赋值，但赋值不等于初始化。例如：

```
long x, y=345;
x=y;
```

y是初始化，x是赋值。

下面通过一个C++程序对普通整型变量和常量做检验。

程序2.1 常量与变量。

```
#include<iostream.h>                    //文件包含命令
int main( )                             //主函数头
{                                       //以第一个左大括号表示主函数体开始
     int   m,n;                         //定义变量
```

```
    m=345 ;                                    //变量赋值
    n=m;
    cout<<"The first C++programming:\n";       //输出一段文本
    cout<<n<<endl;
    cout<<m<<endl;
    cout<<345<<endl;                           //依次输出变量和字面量的值
    cout<<sizeof(n)<<endl;
    cout<<sizeof(345)<<endl;                   //输出变量和常量空间大小
    return(0);                                 //返回操作系统
}
```

<运行结果>

```
The first C++programming:
345
345
345
4
4
```

下面以程序2.1为例，说明一个C++程序的基本构成。

（1）主函数

在C++语言中，主程序是一个主函数，如下所示：

```
int main()
{
    一组程序语句          //......
    return(0);
}
```

main是主函数名，相当于机器语言主程序的入口地址。一对大括号是函数体，包括一组语句，每条语句以分号结束。

“//……”是注释部分，用于说明语句功能，便于理解，它不是程序语句，不会执行。如果一个注释有多行，每行前都要加注释符“//”。

注释部分也常常采用“/*……*/”的形式，例如：

```
/*把变量m的值赋给变量n*/
```

与注释符“//”不同的是，“/*……*/”可以包含多行注释。例如：

```
/* ……
……
…… */
```

执行主函数就是从第一条语句开始，直到return(0)语句为止。

一条变量定义语句可以同时定义几个变量，变量名彼此之间用逗号分隔，例如：

```
int  m,n;
```

一行可以书写多条语句，例如：

```
int m,n; long k;
```

一条语句如果很长，可以分几行书写，但是变量名和函数名不能分行写。

return(0)语句的功能是把0传递给系统，表示主函数正常结束。

（2）变量定义语句和初始化语句

```
int m,n;                          //变量定义语句
m=345;                            //变量赋值语句
long k=52388L;                    //变量初始化语句
```

（3）子函数和文件包含命令

在C++语言中，子程序一般用子函数来表示。子函数就是非主函数的函数。不过一般把子函数称为函数。编译系统提供了一些函数供程序调用，这些函数称为标准库函数。它们按类划分，有标准输入/输出函数、数学函数等。标准库函数的声明（函数头加分号）按类包含在扩展名为.h的文件（称为系统头文件）中，例如数学函数的声明包含在math.h中。一个程序如果调用了一个标准库函数，就要用文件包含命令包含该函数声明所在的系统头文件。例如#include<math.h>就是文件包含命令。

（4）输入/输出运算符

在C++中，从键盘输入一般用对象cin加提取符“ >> ”来表示，在显示器上输出一般用对象cout加插入符“ << ”来表示。使用它们需要包含头文件iostream.h。它们的特点是：一个运算符一次只能操作一个对象；有默认的输入/输出格式。

提取符或插入符可以连续使用，例如：

```
cout<<n<<endl;
```

endl表示换行。又例如：

```
cin>>n>>m;
```

表示从键盘输入两个数据，分别给n和m赋值，数据以空格或换行符来分隔，以换行符结束。

要输出一段文本，需要在文本两端加双引号，例如：

```
cout<<"The first C++programming:\n";
```

其中“ \n ”是控制字符，表示换行。等价格式如下：

```
cout<<"The first C++programming:"<<endl;
```

如果输出一个字符，那么可以用单引号，例如：

```
cout<<n<<','<<m;          //输出变量n和m的值，用逗号分隔
```

（5）程序开发基本步骤

一个C++程序从编写到执行出结果一般需要6个步骤：编辑（edit）、预处理（preprocess）、编译（compile）、连接（link）、装载（load）和执行（execute），如图2-2所示。

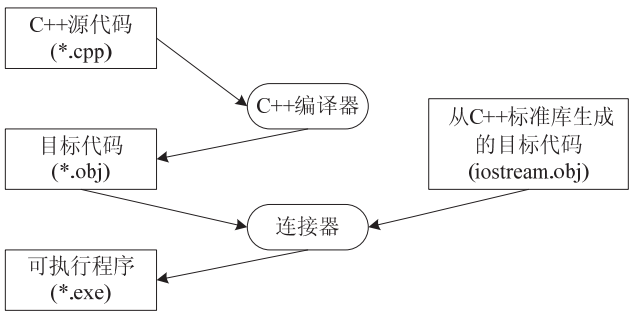

图2-2　编译和连接

编辑是通过编辑器书写C++程序文本，形成源代码（source code），扩展名为.cpp。

预处理是执行程序中的预处理指令。每条预处理指令以符号#开头，结尾不带分号。以程序2.1为例，#include<iostream.h>是预处理指令，该预处理命令将子程序库中的标准输入/输出函数生成中间文件。

编译是进行语法检查，然后将源文件和中间文件分别翻译为二进制代码，形成目标代码片段（object file）。

连接是把不同的二进制代码片段连接成完整的可执行程序（executable file）。

装载将可执行程序放入存储器，然后在中央处理器的控制下执行可执行程序。

（6）程序常见错误

1）拼写错误。例如：

```
int temp;
tmp=5;                    //把temp错写为tmp
```

编译系统报错：'tmp' : undeclared identifier（tmp是没有声明的名称）。

2）如果变量没有赋初值，它的值是一个不可预测的值。例如：

```
int temp;
cout<<temp;               //变量a没有赋初值，结果是一个不可预测的值
```

编译系统警告：local variable ' temp ' used without having been initialized（局部变量temp没有初值）。

3）语句之后丢失分号。例如：

```
int a,b                   //丢失分号。正确的书写格式是：int a,b;
a=3;
b=4;
```

编译系统报错：missing ';' before identifier 'a'（名称a之前缺少分号）。而且系统的错误标志指向的是第2条语句，而不是第1条语句，这是为什么呢？因为语句可以换行，定义语句“ int a,b; ”可以分两行书写，其中的分号可以单独出现在第2行，所以系统是在语句“ a=3; ”之前还没有遇到分号时才报错。

4）对程序语句中出现的中文输入方式下的逗号、分号、小括号等分隔符，编译系统都要报错：unknown character '0xa3'。

5）定义了变量，却没有访问。例如：

```
int a,b;
a=5;
cout<<a;                  //只访问了a，没有访问b
```

编译系统要提示：'b' : unreferenced local variable（变量b没有访问）。

6）调用了标准库函数，却没有包含相应的系统头文件。例如：

```
int main( )
{
     int   m=345 ;
     cout<<m;
     return(0);
}
```

因为没有包含头文件iosteam.h，所以系统提示：'cout' : undeclared identifier（cout是没有声明的标识符）。

2.2 整型

1. 整型种类

整型根据空间大小不同而分为普通整型（int也称基本整型，简称整型）、长整型（long）和短整型（short）。根据符号有无，上述各类型又分为（符号）整型和无符号整型（unsigned）。一种类型占几个字节，因系统而定。

空间大小和存储格式：在32位操作系统中，短整型变量占2个字节，其他整型变量占4个字节。（符号）整型变量以高位（左第1位）表示符号（0代表正，1代表负），剩余位表示数值。数值以补码形式存放。短整型变量的数值范围是 -2^{15}～$2^{15}-1$（-32 768~32 767）。普通整型和长整型变量的数值范围是 -2^{31}～$2^{31}-1$（-2 147 483 648~2 147 483 647）。无符号整型变量没有符号位，数值位多一个，最大值增加一倍，最小值为0。

2. 基本操作

对整型数据的基本操作主要有：算术运算（+，-，*，/，%）、关系运算（<，<=，>，>=，!=）、逻辑运算（&&，||，!）。例如：

```
45/20=2          //整除
15/20=0          //整除
14%5=4           //求余
35%7=0           //求余
```

编写程序：将一个正整数在显示器上逆序输出。

基本方法：反复使用整型的基本操作整除和求余。步骤如下：

1）用10对该整数求余，结果是该整数在十进制下的个位数，然后输出。

2）用10对该整数进行整除，结果是该整数在十进制下的十位数降阶为个位数，然后求余输出。

3）用100对该整数进行整除，结果是该整数在十进制下的百位数降阶为个位数，然后求余输出。

4）依次类推。

程序2.2　将3位整数按逆序输出。

```
#include<iostream.h>
int main()
{
     int i,n,re;
     cin>>n;                    //从键盘输入一个整数578，给n赋值
     i=n%10;                    //步骤1)。求n的个位数。i=8
     cout<<i;
     re=n/10;                   //步骤2)。降阶。re=57
     i=re%10;                   //求n的个位数。 i=7
     cout<<i;
     re=n/100;                  //步骤3)。降阶。re=5
     i=re%10;                   //求n的个位数。i=5
     cout<<i<<endl;
     return(0);
}
```

<运行结果>

```
578                      //从键盘输入的一个整数
```

```
875                              //逆序输出结果
```

3. 整型常量

整型常量前加符号0x，表示十六进制常量，包括数字0~9和a~f（A~F），其中a~f表示10~15。例如0x12fe，表示12fe是十六进制整数。

整型常量前加符号0，表示八进制常量，包括数字0~7。例如0127，表示127是八进制整数。

在整型常量后面加符号L或U（l或u），表示长整型或无符号整型。例如52388L表示长整数，40000U表示无符号整数。如果编译系统中普通整型（int）和长整型（long）的长度相同，后缀L就可以省去，但是加上后缀，有利于程序“移植”。

整型字面值常量可以写成十进制、八进制或十六进制的形式，例如20可以表示为如下的三种形式之一：

```
• 20                 //十进制数
• 024                //八进制数。前面的0表示24是八进制数
• 0x14               //十六进制数。前面的0x表示14是十六进制数
```

程序2.3 整型常量。

```
#include<iostream.h>
int main()
{
     int A,OctalA,HexA;                    //普通整型变量
     A=20;                                 //普通整数20
     OctalA=024;                           //八进制整数20
     HexA=0x14;                            //十六进制整数20
     cout<<A<<endl;
     cout<<OctalA;
     cout<<HexA<<endl;
     return(0);
}
```

<运行结果>

```
20
20
20
```

2.3 字符型

1. 字符型常量

字符型（char）数据包括大小写字母、数字、标点符号及特殊字符。字符字面量用单引号界限符表示类型，例如'A'，'6'。注意'6'和6的区别：'6'是字符字面量，6是普通整型字面量。在字处理、文本输入/输出和数据通信中，应用最广泛的是128个标准字符——ASCII字符集。标准字符分为可见字符和控制字符，控制字符用于数据通信和设备控制，例如，换行、响铃。

128个标准字符和数值0至127一一对应，这种对应使字符型数据能以一个字节的整数形式存储（只用7位，2^7=128），后者称为字符的代码。常见ASCII字符代码见表2-1。

表2-1 ASCII代码字符集

	0	1	2	3	4	5	6	7	8	9
3			◆	!	"	#	$	%	&	’
4	(	)	*	+	,	—	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	C
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~			

注：◆表示空格。

标准字符和数值的对应是精心安排的，简化了大小写字符代码之间的转换等操作。例如：大写字符代码加32就转换到小写字符代码。表2-2为ASCII字符代码范围。

表2-2 ASCII字符代码范围

ASCII字符	十进制码	二进制码
空格	32	00100000
十进制数字字符	48~57	00110000~00111001
大写字母	65~90	01000001~01011010
小写字母	97~122	01100001~01111010

2. 基本操作

因为字符型的存储是整型（只占一个字节），所以字符型数据的基本操作与整型的基本操作相同，包括算术运算（+，-，*，/）、关系运算（<，<=，>，>=，!=）、逻辑运算（&&，||，!）、求余（%）。只要操作结果在0～127之间，操作就有意义。

3. 输出格式

字符型变量的默认输出格式是字符。如果要输出字符型变量的十进制代码，需要强制类型转换。

强制类型转换的格式是：

(转换的类型标识符)变量

例如：

```
char ch='A';
cout<<ch<<end1;          //默认输出字符A
cout<<(int)ch<<end1;     //输出代码65
```

一个字符可以赋值给整型变量，但是要输出该字符，需要加强制类型转换，因为整型变量的默认输出格式是整型。例如：

```
int n='A';
cout<<(char)n<<endl;     //按字符型显示，结果是A
```

一个0~127之间的整型数也可以赋值给字符型变量，相当于字符代码。例如：

```
char ch=65;                    //相当于char ch='A';
```

如果大于127，系统就会警告，指出数据被截断（truncation）。

程序2.4 测试字符和字符代码，以及字符大写和小写的关系。

```
#include<iostream.h>
int main()
{
    char ch1,ch2;
    ch1='A';                                    //或ch1=65; 字符型常量也可以用代码表示
    ch2=ch1+32;                                 //ch2的值为小写字符a
    cout<<ch1<<','<<ch2<<endl;                  //输出字符
    cout<<(int)ch1<<','<<(int)ch2<<endl;        //输出字符十进制代码
    ch1=ch1+1;                                  //ch1的值改为B
    ch2=ch2+1;                                  //ch2的值改为b
    cout<<ch1<<','<<ch2<<endl;                  //输出改变后的字符
    return(0);
}
```

<运行结果>

```
A,a
65,97
B,b
```

4. 转义字符

有一些字符是控制字符，它们是不可显示字符，如换行、回车、换页、响铃，还有一些字符已经有了特殊的用处，如单引号已用作字符常量界限符等，这些字符不能再简单地用一个字符来表示，我们用反斜杠开头的字符序列来表示，这时，反斜杠之后的字符或字符序列不再是本来的含义，而是被转换为另外的含义，称为转义字符（见表2-3）。例如，转移字符'\n'表示换行，'\r'表示回车。

单引号如果是字符常量，必须加反斜杠才能与字符常量的界限符区分。也就是说，要写成'\''，而不能写成'''。

表2-3 常用转义字符

转义字符	代 码	功 能
\0	0	空字符（null） //字符型0元素
\a	7	音符（bell）
\b	8	退格（backspace）
\t	9	水平制表（horizontal tab）
\n	10或0x0a	换行（newline）
\v	11或0x0b	垂直制表（vertical tab）
\f	12或0x0c	换页（formfeed）
\r	13或0x0d	回车（carriage return）
\"	34或0x22	双引号（double quote）
\'	39或0x27	单引号（single quote）
\\	92或0x5c	反斜杠（backslash）
\ddd		1到3位八进制数所代表的字符
\xhh		1到2位十六进制数所代表的字符

基于转义字符，字符常量有多种表示，以字符N为例，有以下几种表示：

• 'N'; //字符常量N

• '\x4e';　　//十六进制转移字符表示的N
• '\116';　　//八进制转移字符表示的N

程序2.5　检测字符型常量和转义字符。

```
#include<iostream.h>
int main()
{
     char ch1,ch2,ch3,ch4,ch5,ch6;
     ch1='N';                                    //字符常量N
     ch2=78;                                     //78是字符常量N的十进制代码
     ch3=0x4e;                                   //0x4e是字符常量N的十六进制代码
     ch4=0116;                                   //0116是字符常量N的八进制代码
     ch5='\x4e';                                 //'\x4e'表示十六进制整数代表的字符常量N
     ch6='\116';                                 //'\116'是八进制整数代表的字符常量N
     cout<<ch1<<'\t'<<ch2<<'\t'<<ch3<<endl;      //\t表示按水平制表格式输出字符
     cout<<'\a';                                 //\a代表一声铃响
     cout<<ch4<<'\t'<<ch5<<'\t'<<ch6<<endl;      //\t表示按水平制表格式输出字符
     cout<<'\a';
     return(0);
}
```

<运行结果>

```
N       N       N
N       N       N
```

程序分析：0x4e和'\x4e'是有区别的，前者表示字符常量N的十六进制代码，而后者表示十六进制代码所对应的字符；如果作为字符串成员看待，前者（0x4e）表示4个字符（0，x，4，e），后者（'\x4e'）仅代表一个字符N。

2.4　实型

1. 实型种类

浮点实型按精度不同，划分为单精度浮点型（简称单浮点型float）、双精度浮点型（简称双浮点型double）和长精度浮点型（简称长浮点型long double）。

空间大小和表示格式：单精度浮点型占4个字节，其余占8个字节。实数的表示有定点格式和指数格式（即科学记数法）。定点格式（小数形式）用小数点分开整数和小数。指数格式包括尾数和指数两部分，用字符E（或e）分开，指数部分表示10的多少次方。例如：

```
3.14159                          //定点格式
1.234E20=1.234*10^20             //指数格式
```

字面值常量：在默认状态下为双浮点型，例如，314.159是双浮点型常量。单浮点型常量要加后缀F（或f），例如，314.159F或314.159f。长浮点型加后缀L（或l），例如，3.141 59E2L。3.141 59是默认双浮点型字面量，3.141 59F是单浮点型字面量。

2. 基本操作

对实型数据的基本操作有：算术运算（+，-，*，/）、关系运算（<，<=，>，>=，!=）、逻辑运算（&&，||，!）。同样是除法，实数除法和整数除法不同：前者保留小数，后者是整除，舍去小数。

2.5 布尔型

布尔型只占用一个字节，仅取两个值：1和0，字面值是true和false。

程序2.6 检验布尔型。

```
#include<iostream.h>
int main()
{
     bool b1,b2;                          //定义布尔型变量
     b1=true;                             //给布尔型变量赋值
     b2=false;
     cout<<sizeof(bool)<<endl;            //输出布尔型空间大小
     cout<<b1<<'\t'<<b2<<endl;            //输出布尔型变量的值
     return(0);
}
```

<运行结果>

```
1
1   0
```

要避免整型、实型等与布尔型做比较运算。例如：

```
(a<b)>c
```

a<b的结果是布尔型值，如果c是整型变量，系统就会提出警告unsafe use of type 'bool' in operation（运算中的布尔型使用不安全）。实际上，在正常的程序中，一个整型和一个布尔型也不应该进行比较运算。

再例如：

```
(a<b)<(a>c)
```

（a<b）和（a>c）的结果都是布尔型值，虽然系统没有报错，但是两个布尔型不应该做比较运算。

2.6 typedef 名字

typedef的作用是给已存在的数据类型引入一个别名（也称同义词或助记符），我们把这个别名称为typedef名字。typedef的应用格式为：

```
typedef 已有的类型名 类型别名;
```

例如，Pascal语言用INTEGER表示整型，熟悉Pascal语言的人，为了保留习惯，可以利用typedef机制，使INTEGER成为C++语言int的同义词，如下所示：

```
typedef int INTEGER;
```

2.7 基本数据类型的深入探讨

2.7.1 字面值常量的意义

对基本类型的字面量不能实施寻址操作&，因此我们不必关心它们在存储空间中的位置。但是我们要注意字面量在保持逻辑一致性上的意义。例如，操作符sizeof有两种用法：sizeof

(类型标识符)和sizeof(数据空间标识符)。引入字面量的概念之后，sizeof(124)、sizeof(3.13f)和sizeof('A')等就可以归结为第二种用法。特别是对赋值操作符而言，引入字面量的概念之后，不论右元是变量还是常量，赋值操作都有了统一的描述：从右元所标识的空间取值，赋值给左元所标识的空间。

2.7.2 左值和右值

在赋值操作中，任左元的操作数都可以任右元，例如基本类型变量；但是任右元的未必都可以任左元，例如常量。我们把既可以是左元也可以是右元的操作数称为左值（lvalue，发音为ell-value），把仅可以任右元的操作数称为右值（rvalue，发音为are-value）。例如，基本类型的变量是左值，字面量是右值。因为任左元的操作数都可以任右元，所以一般都把左元等同于左值。

习题

一、单选题

1. 下列（ ）选项可以作为合法的变量名。

A. 3X　　B. file_1　　C. int　　D. X+Y;

2. 下列（ ）选项是合法的实型字面常量。

A. 8E3.1　　B. E5　　C. 234　　D. 234

3. 下列（ ）选项整数值最大。

A. 012　　B. 0x12　　C. 12　　D. 120

4. 下列（ ）选项中的字符与其他三个选项中的字符不相等。

A. 'a'　　B. 'A'　　C. '\x41'　　D. '\101'

二、读程序

1. 写出下面程序的运行结果。

```
#include <iostream.h>
int main()
{
     int i=050,j=50,k=0x50;
     cout<<i<<','<<j<<','<<k<<endl;
     return(0);
}
```

2. 写出下面程序的运行结果。

```
#include <iostream.h>
int main()
{
     short int a=0x7ffc,b=a+1;
     cout<<a<<','<<b<<endl;
     return(0);
}
```

三、编程题

1. 将5位整数按逆序输出。

2. 编程测试整型、单浮点型、双浮点型和字符型字面值常量的空间大小。