ASCII 编码

ASCII 编码（American Standard Code for Information Interchange，美国信息交换标准代码），是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统。

标准 ASCII 码使用 7 位二进制数组合来表示 128 种可能的字符，其中包括

- 10 个数字，
- 26 个大写字母，
- 26 个小写字母，
- 32 个标点符号及特殊符号，
- 34 个控制字符。

每个标准 ASCII 码存储占有 1 个字节（8 位），最高位为 0。如果最高位是 1，则表示的是扩展 ASCII 码，扩展 ASCII 码不是标准的。

下面的一个字节就代表的是字符'A'。

0	1	0	0	0	0	0	1

全部 128 个标准 ASCII 字符如下：

0	NUL 空字符	64	@
1	SOH 标题开始	65	A
2	STX 正文开始	66	B
3	ETX 正文结束	67	C
4	EOT 传输结束	68	D
5	ENQ 请求	69	E
6	ACK 收到通知	70	F
7	BEL 响铃	71	G

8	BS 退格	72	H
9	HT 水平制表符	73	I
10	LF 换行	74	J
11	VT 垂直制表符	75	K
12	FF 换页	76	L
13	CR 回车	77	M
14	SO 不用切换	78	N
15	SI 启用切换	79	O
16	DLE 数据链路转义	80	P
17	DC1 设备控制 1	81	Q
18	DC2 设备控制 2	82	R
19	DC3 设备控制 3	83	S
20	DC4 设备控制 4	84	T
21	NAK 拒绝接收	85	U
22	SYN 同步空闲	86	V
23	ETB 结束传输块	87	W
24	CAN 取消	88	X
25	EM 媒介结束	89	Y
26	SUB 代替	90	Z
27	ESC 换码	91	[
28	FS 文件分隔符	92	\
29	GS 分组符	93	]
30	RS 记录分隔符	94	^
31	US 单元分隔符	95	_

32	(space) 空格	96	`
33	!	97	a
34	“	98	b
35	#	99	c
36	$	100	d
37	%	101	e
38	&	102	f
39	‘	103	g
40	(	104	h
41	)	105	i
42	*	106	j
43	+	107	k
44	,	108	l
45	-	109	m
46	.	110	n
47	/	111	o
48	0	112	p
49	1	113	q
50	2	114	r
51	3	115	s
52	4	116	t
53	5	117	u
54	6	118	v
55	7	119	w

56	8	120	x
57	9	121	y
58	:	122	z
59	;	123	{
60	<	124	|
61	=	125	}
62	>	126	~
63	?	127	DEL 删除

表1. ASCII 表字符集

表示整数的数据类型

在 C++语言里，根据整型变量的取值范围的不同，可以定义 8 种整型类型。

实际上，C++的标准没有规定每种整型的取值范围，也就是整型数据的字节长度。真实的取值范围和计算机的中央处理器（CPU）、操作系统（OS）、C++的集成开发环境（IDE）的编译器版本都有关。

下表的实测结果是基于以下硬件及软件环境的：

- CPU：Intel CORE i7
- OS：Win10 64 位
- IDE：Dev C++ 5.11 （GCC 4.9.2 64bit）

数据类型	类型标识符	占字节数	数值范围
短整型	short [int]	2（16 位）	-32768 ~ 32767 (-2^{15} ~ 2^{15}-1)
整型	int	4（32 位）	-2147483648 ~ 2147483647 (-2^{31} ~ 2^{31}-1)
长整型	long [int]	4（32 位）	-2147483648 ~ 2147483647 (-2^{31} ~ 2^{31}-1)
超长整型	long long [int]	8（64 位）	-9223372036854775808 ~ 9223372036854775807 (-2^{63} ~ 2^{63}-1)
无符号短整型	unsigned short [int]	2（16 位）	0 ~ 65535 (0 ~ 2^{16}-1)
无符号整型	unsigned [int]	4（32 位）	0 ~ 4294967295 (0 ~ 2^{32}-1)
无符号长整型	unsigned long [int]	4（32 位）	0 ~ 4294967295 (0 ~ 2^{32}-1)
无符号超长整型	unsigned long long [int]	8（64 位）	0 ~ 18446744073709551615 (0 ~ 2^{64}-1)

表1. C++语言中的整数数据类型

参加各个级别计算机竞赛时（比如 NOIP），可以认为 int 的上限约是2×10^9，long long 的上限约是9×10^{18}。

类型标识符中的[int]表示在声明变量的时候[int]是可以省略的。以 short [int]为例，下面的两种声明方式都是合法的：

```
short a1;
short int a2;
```

更多的整型变量的声明方式如下：

```
short n1;
int n2;
long n3;
long long n4;
unsigned short n5;
unsigned n6;
unsigned long n7;
unsigned long long n8;
```

表示实数的数据类型

在 C++语言里，实数也叫浮点数，有三种数据类型。

数据类型	类型标识符	占字节数	数值范围	有效位数
单精度实型	float	4（32 位）	-3.4E-38 ~ 3.4E+38	6 ~7 位
双精度实型	double	8（64 位）	-1.7E-308 ~ 1.7E+308	15 ~ 16 位
长双精度实型	long double	16（128 位）	-3.4E-4932 ~ 1.1E+4932	18 ~ 19 位

表2. C++语言中的实数数据类型

实型变量的定义方式如下：

```
1. float f1;
2. double f2;
3. long double f3;
```

表示字符的数据类型

在 C++语言里，字符型表示的字符为 ASCII 码表范围。每个字符存储时占用 8 位，即一个字节。

字符型变量的定义方式如下：

```
1.  char c;
```

表示布尔值的数据类型

在 C++语言里，布尔类型表示逻辑运算的“真”和“假”。

- “真”用 true 表示，内部用 1 存储。
- “假”用 false 表示，内部用 0 存储。

布尔数据存储时占用 8 位，即一个字节。

布尔变量的定义方式如下：

```
1.  bool b;
```

运算符 sizeof

运算符 sizeof 可以用来计算一个变量（对象）或一种数据类型所占用的内存字节数。

【问题描述】

分别定义 int, short 类型的变量各一个, 并依次输出它们的存储空间大小(单位: 字节)。

【参考程序】

```
#include<iostream>
using namespace std;
int main()
{
    int x;
    short y;
    //sizeof 返回整型变量所占的内存字节数
    cout << sizeof(x) << endl;
    //sizeof 返回短整型变量所占的内存字节数
    cout << sizeof(y) << endl;
    //sizeof 返回整形数据类型所占的内存字节数
    cout << sizeof(int) << endl;
    //sizeof 返回短整形数据类型所占的内存字节数
    cout << sizeof(short) << endl;
    return 0;
}
```

【运行结果】

```
4
2
4
2
```

程序的顺序结构

每条语句按自上而下的顺序依次执行一次就是程序的顺序结构。

简单计算器

【问题描述】

发挥你的聪明才智，编写自己的简单计算器。

输入两个数字，分别能够实现加、减、乘、除运算，并输出结果。

可以是一个程序也可以是四个程序。

【参考代码】

```
//求两个数的加法
#include<iostream>
using namespace std;
int main()
{
    double x, y;
    cout << "Please input the first addend: ";
    cin >> x;
    cout << "Please input the second addend: ";
    cin >> y;
    cout << x << "+" << y << "=" << x+y << endl;
    return 0;
}

//求两个数的减法
#include<iostream>
using namespace std;
int main()
{
    double x, y, a;
    cout << "Please input minuend: ";
    cin >> x;
    cout << "please input subtrahend: ";
    cin >> y;
    cout << x << "-" << y << "=" << x-y << endl;
    return 0;
}

//求两个数的乘法
#include<iostream>
using namespace std;
```

```
int main()
{
    double x, y;
    cout << "Please input the first factor: ";
    cin >> x;
    cout << "Please input the second factor: ";
    cin >> y;
    cout << x << "*" << y << "=" << x*y <<endl;
    return 0;
}
```

```
// 求两个数的除法
#include<iostream>
using namespace std;
int main()
{
    double x, y;
    cout << "Please input dividend: ";
    cin >> x;
    cout << "Please input divisor: ";
    cin >> y;
    cout << x << "/" << y << "=" << x/y << endl;
    return 0;
}
```

【单词列表】

addend	加数
minuend	被减数
subtrahend	减数
factor	乘数
dividend	被除数
divisor	除数