chapter3 处理数据

1）climits中的符号常量：

climits 定义了符号常量，这里总结了部分常用是：

符号常量	表示
CHAR_BIT	char的位数
INT_MAX	int的最大值
LONG_MAX	long的最大值
SHAR_MAX	short的最大值
LLONG_MAX	long long 的最大值
CHAR_MAX,CHAR_MIN	char的最大值和最小值
SCHAR_MAX,SCHAR_MIN	signed char的最大值和最小值
UCHAR_MAX,UCHAR_MIN	unsigned char的最大值和最小值

```
#include <iostream>
#include <climits>
using namespace std;
int main(int argc, char const *argv[])
{

    int n_int = INT_MAX;

    short n_short = SHRT_MAX;

    long n_long = LONG_MAX;

    long long n_llong = LLONG_MAX;

    std::cout << "int is       " << CHAR_BIT << "  bytes." << '\n';

    std::cout << "short is     " << sizeof(short) << "  bytes." << '\n';

    std::cout << "long is      " << sizeof(long) << "  bytes." << '\n';

    std::cout << "long long is " << sizeof(long long) << "  bytes." << '\n';
```

```
    std::cout << "Maxinum values:" << '\n';

    std::cout << "int       : " << n_int << '\n';

    std::cout << "short     : " << n_short << '\n';

    std::cout << "long      : " << n_long << '\n';

    std::cout << "long long : " << n_llong << '\n';

    std::cout << "Bits per byte : " << CHAR_BIT << '\n';

    return 0;

}
```

结果：

```
int is       8  bytes.
short is     2  bytes.
long is      4  bytes.
long long is 8  bytes.
Maxinum values:
int       : 2147483647
short     : 32767
long      : 2147483647
long long : 9223372036854775807
Bits per byte : 8
```

2）ASCII 表：

ASCII码对照全表

二进制	十进制	十六进制	符号	解释
0000 0000	0	0	NUL	空字符
0000 0001	1	1	SOH	标题开始
0000 0010	2	2	STX	正文开始
0000 0011	3	3	ETX	正文结束
0000 0100	4	4	EOT	传输结束
0000 0101	5	5	ENQ	询问
0000 0110	6	6	ACK	收到通知
0000 0111	7	7	BEL	铃
0000 1000	8	8	BS	退格
0000 1001	9	9	HT	水平制表符
0000 1010	10	0A	LF	换行键
0000 1011	11	0B	VT	垂直制表符
0000 1100	12	0C	FF	换页键
0000 1101	13	0D	CR	回车键
0000 1110	14	0E	SO	移出
0000 1111	15	0F	SI	移入
0001 0000	16	10	DLE	数据链路转义
0001 0001	17	11	DC1	设备控制1
0001 0010	18	12	DC2	设备控制2
0001 0011	19	13	DC3	设备控制3
0001 0100	20	14	DC4	设备控制4
0001 0101	21	15	NAK	拒绝接收
0001 0110	22	16	SYN	同步空闲
0001 0111	23	17	ETB	传输块结束
0001 1000	24	18	CAN	取消
0001 1001	25	19	EM	介质中断
0001 1010	26	1A	SUB	替换
0001 1011	27	1B	ESC	换码符
0001 1100	28	1C	FS	文件分隔符
0001 1101	29	1D	GS	组分隔符
0001 1110	30	1E	RS	记录分隔符
0001 1111	31	1F	US	单元分隔符
0111 1111	127	7F	DEL	删除

二进制	十进制	十六进制	符号
0010 0000	32	20	`
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

二进制	十进制	十六进制	符号
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

二进制	十进制	十六进制	符号
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	g
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

二进制	十进制	十六进制	符号	解释
1000 0000	128	80	€	欧盟符号
1000 0001	129	81		
1000 0010	130	82	‚	单低 9 引号
1000 0011	131	83	ƒ	带钩的

二进制	十进制	十六进制	符号
1010 0000	160	A0	
1010 0001	161	A1	¡
1010 0010	162	A2	¢

二进制	十进制	十六进制	符号
1100 0000	192	C0	À
1100 0001	193	C1	Á
1100 0010	194	C2	Â

二进制	十进制	十六进制	符号
1110 0000	224	E0	à
1110 0001	225	E1	á
1110 0010	226	E2	â

https://blog.csdn.net/zhang992283

3）缩放因子表示方法：AeB / AEB

eg: 1e6 = 10^6 5.98E24 = 5.98 x 10^24 9.11e-34

（1）指数可以是正数，也可以是负数

（2）E与e在此处等价

（3）数字中不可以有空格 eg： 7.2 e6 非法

4）c++11中的auto声明： 自动推断变量的类型

例如：

```
auto n = 100;        // n is int
auto x = 1.5;        // x is double
auto y = 1.3e12L;    // y is long double
```

auto的真正优势在处理复杂类型中得以体现，比如STL中的类型：

```
[approach1]
std::vector<double> hbx;
std::vector<double>::iterator hhh = hbx.begin();

[approach2]
```

```
std::vector<double> hbx;
auto hhh = hbx.begin();
```

对比可知：

```
std::vector<double>::iterator = auto
```