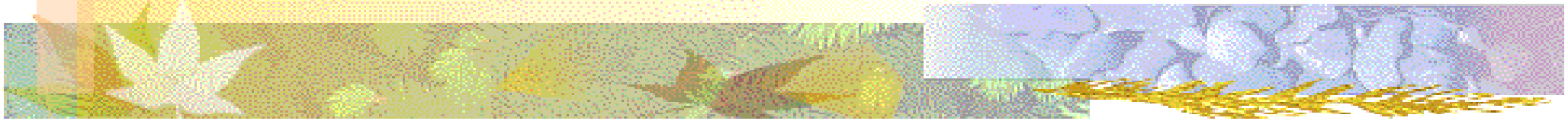
第1章 C数据类型

——变量的类型决定了什么？



哈尔滨工业大学

苏小红

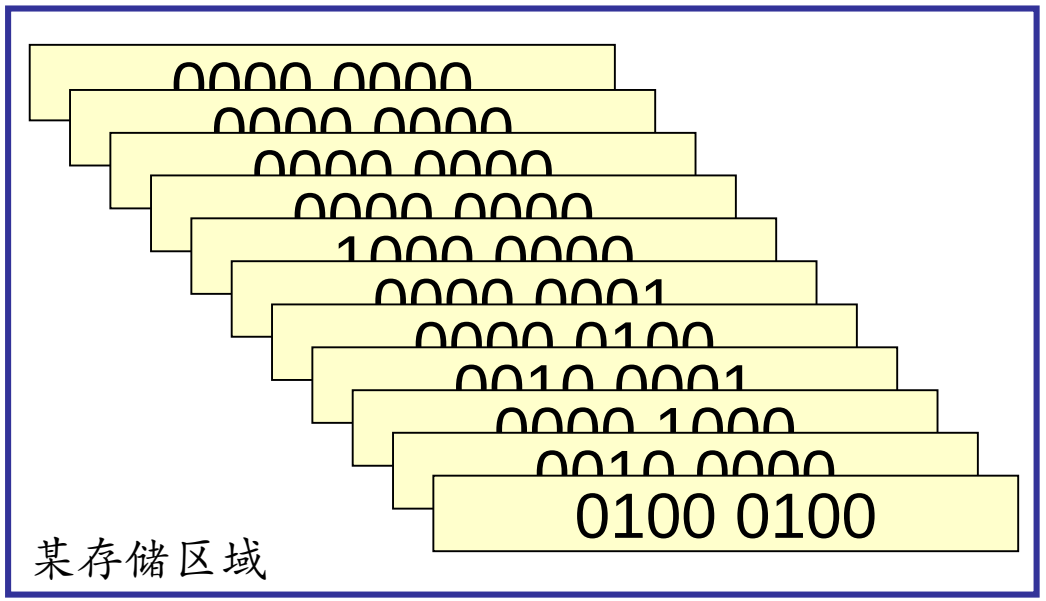
sxh@hit.edu.cn

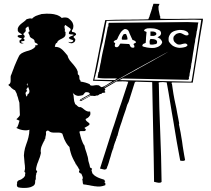
本讲要讨论的主要问题

- 在高级语言中为什么要引入数据类型？C语言中有哪些数据类型？基本数据类型有哪些？
- 变量的类型决定了什么？
- 在C语言中，如何计算变量或类型所占内存空间的大小？

数据类型（Data Type）

- 在冯·诺依曼体系结构中
  - 程序代码和数据都是以二进制存储的
  - 对计算机系统和硬件本身而言，数据类型的概念并不存在

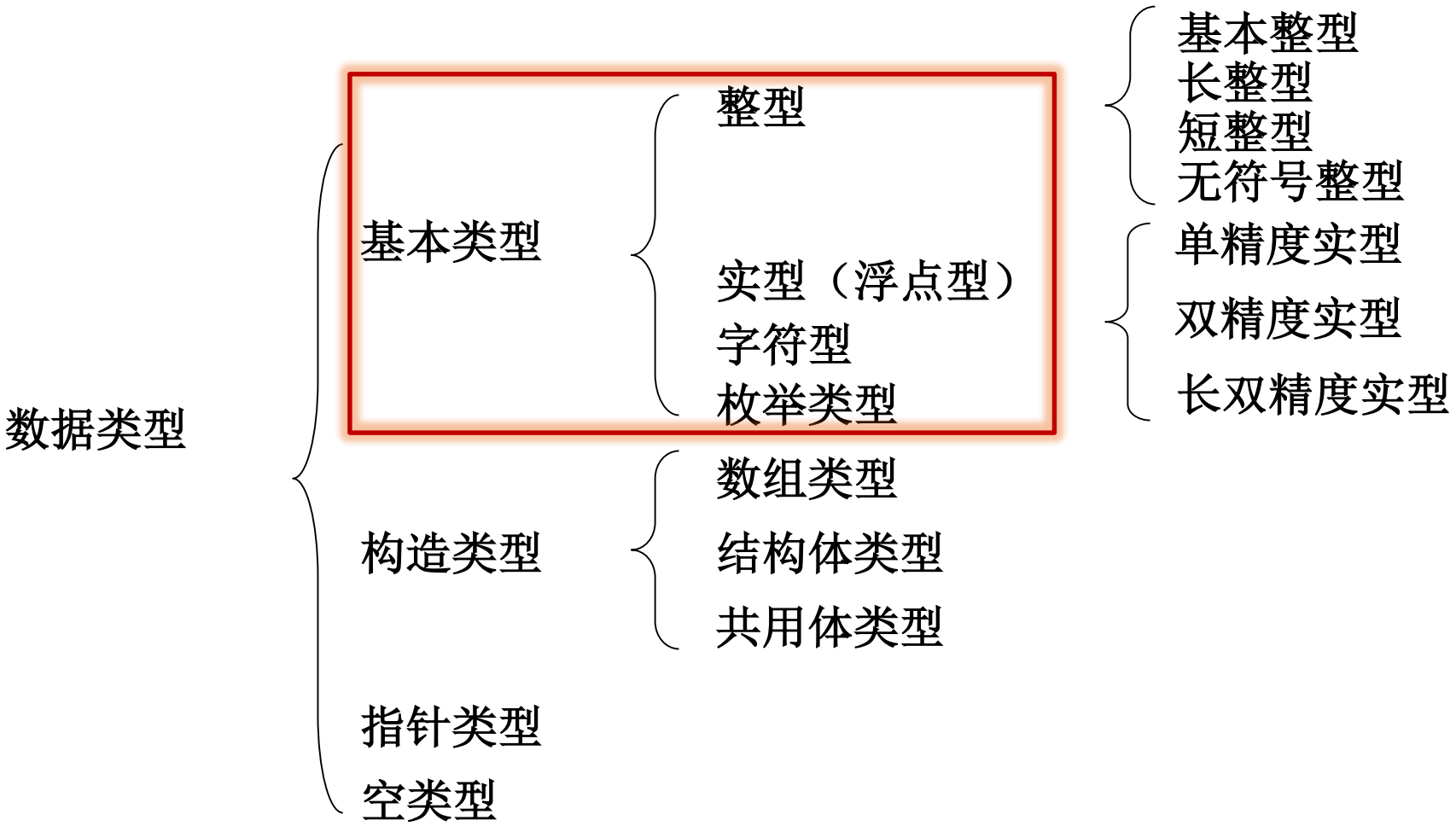




数据类型（Data Type）

- 问题：高级语言为什么要区分数据类型？
  - 更有效地组织数据，规范数据的使用
  - 有助于提高程序的可读性，方便用户的使用
- 在程序设计语言中引入数据类型的好处
  - 带来了程序的简明性和数据的可靠性
  - 有助于提高程序执行效率、节省内存空间

C语言中的数据类型

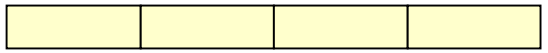

数据类型
基本类型
整型
基本整型
长整型
短整型
无符号整型
实型（浮点型）
单精度实型
双精度实型
长双精度实型
字符型
枚举类型
构造类型
数组类型
结构体类型
共用体类型
指针类型
空类型

变量的类型决定了什么？

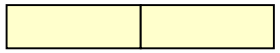
* 占用内存空间的大小
* 数据的存储形式
* 合法的表数范围
* 可参与的运算种类

(1)不同类型数据占用的内存大小不同

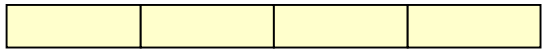
- int——基本整型，C标准未规定，系统相关
  - 在目前大多数系统上占4个字节



- short int，简写为short
  - 短整型，2个字节



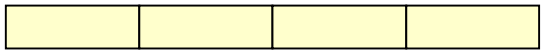
- long int，简写为long
  - 长整型，4个字节



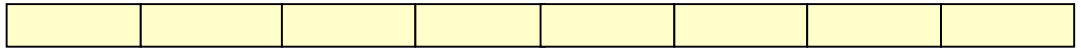
- unsigned——无符号整型（正整数和0）
  - 用来修饰int、short和long

(1)不同类型数据占用的内存大小不同

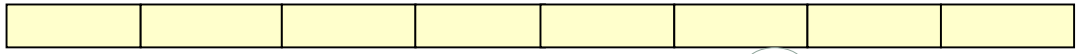
- float
  - 单精度实型，4个字节



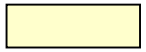
- double
  - 双精度实型，8个字节

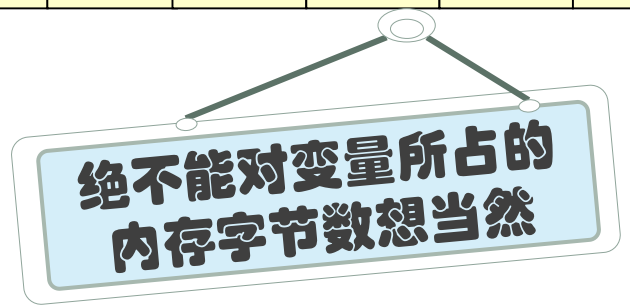


- long double
  - 长双精度实型， IEEE规定10个字节，系统相关
  - VC++中占8个字节



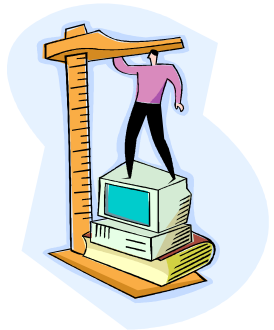
- char
  - 字符型，1个字节





如何计算变量或类型占内存的大小

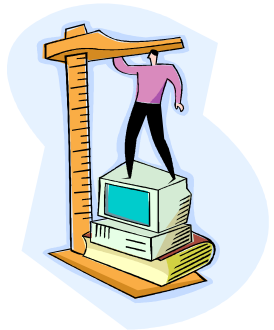
- 问题：如何计算变量占内存空间的大小？
  - 用sizeof运算符
  - 一元运算符
- 用sizeof运算符计算变量占内存空间的大小的好处
  - 增加程序的可移植性
  - 编译时执行的运算符，不会导致额外的运行时间开销



如何计算变量或类型占内存的大小

- **一般形式：**

语法形式	运算结果
sizeof(类型)	类型占用的内存字节数
sizeof(变量或表达式)	变量或表达式所属类型占的内存字节数



(2)不同数据类型的表数范围不同

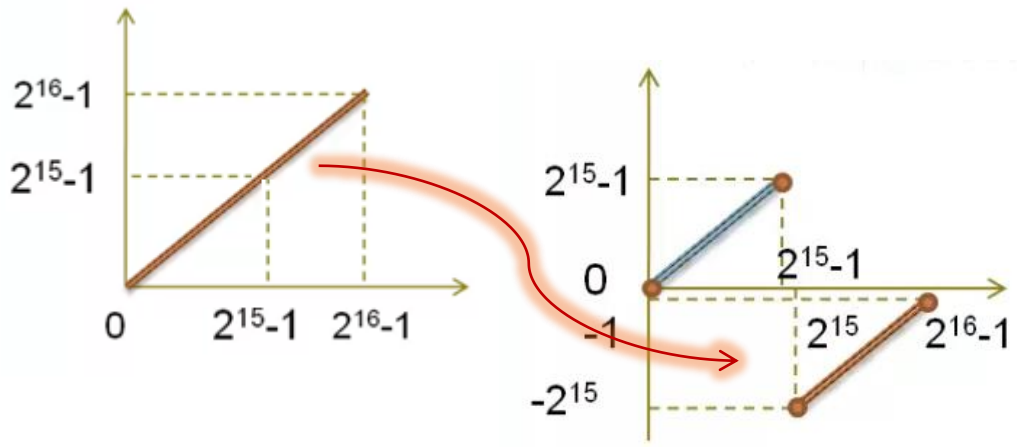
有符号和无符号整数的表数范围也不同

Visual C++

数据类型	占内存的字节数	下限值	上限值
char (signed char)	1	-128	127
unsigned char	1	0	255
short int (signed short int)	2	-32768	32767
unsigned short int	2	0	65535
unsigned int	4	0	4294967295
int (signed int)	4	-2147483648	2147483647
unsigned long int	4	0	4294967295
long int (signed long int)	4	-2147483648	2147483647
float	4	-3.4×10^{-38}	3.4×10^{38}
double	8	-1.7×10^{-308}	1.7×10^{308}
long double	8	-1.7×10^{-308}	1.7×10^{308}

(2)不同数据类型的表数范围不同

* 以2字节（16位）短整型为例
  * 有符号整数的最高位是符号位，使其数据位比无符号整数的数据位少了1位
  * 有符号整数能表示的最大整数的绝对值仅为最大无符号整数的一半



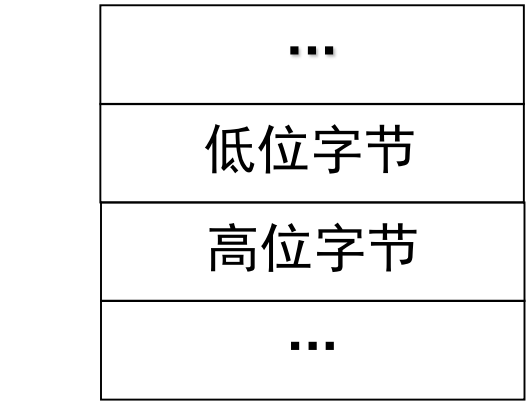
无符号短整型（最高位是数据位）		有符号短整型（最高位是符号位）	
二进制补码	十进制	二进制补码	十进制
00000000 00000000	0	00000000 00000000	0
00000000 00000001	1	00000000 00000001	1
00000000 00000010	2	00000000 00000010	2
00000000 00000011	3	00000000 00000011	3
…		…	
01111111 11111111	32767	01111111 11111111	32767
10000000 00000000	32768	10000000 00000000	-32768
10000000 00000001	32769	10000000 00000001	-32767
…		…	
11111111 11111110	65534	11111111 11111110	-2
11111111 11111111	65535	11111111 11111111	-1

(3)不同类型数据的存储形式不同

- **整型数**
  - **一个多字节的数据是如何存放到存储单元中的呢？**

高位字节	低位字节

  - **小端次序**
    - **便于计算机从低位字节向高位字节运算**
  - **大端次序**
    - **与人们从左到右的书写顺序相同，便于处理字符串**



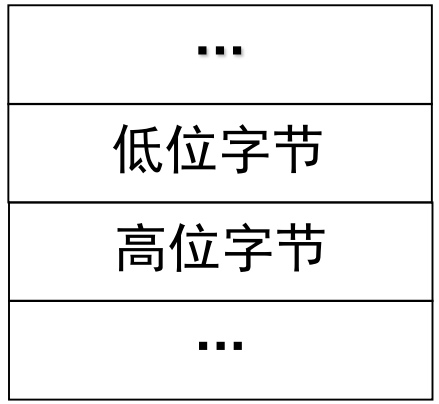
小端次序（Little-endian）

大端次序（Big-endian）

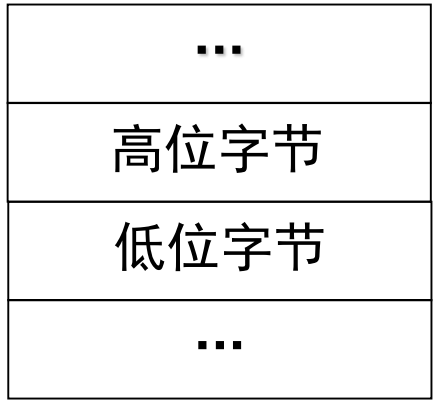
(3)不同类型数据的存储形式不同

- **整型数**
  - 一个多字节的数据是如何存放到存储单元中的呢？

高位字节	低位字节



小端次序（Little-endian）



大端次序（Big-endian）

- 问题：如何存储**实型数**呢？
  - 关键：确定小数点的位置

(3)不同类型数据的存储形式不同

- **整型数**
  - 一个多字节的数据是如何存放到存储单元中的呢？

- 问题：如何存储**实型数**呢？
  - 关键：确定小数点的位置

(3)不同类型数据的存储形式不同

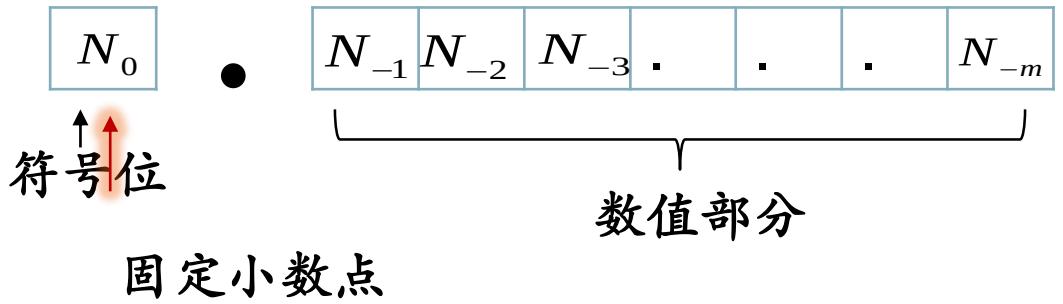
- 问题：如何表示实型数？
  - 小数形式
  - 指数形式——科学计数法

整数部分	小数部分

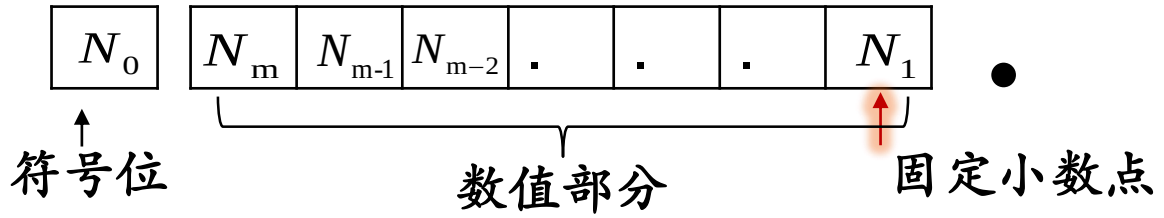
- 定点数（Fixed Point）
  - 小数点的位置固定
  - 定点整数
  - 定点小数

(3)不同类型数据的存储形式不同

* 定点数
  * 定点小数（纯小数）——小数点位于符号位和最高数值位之间

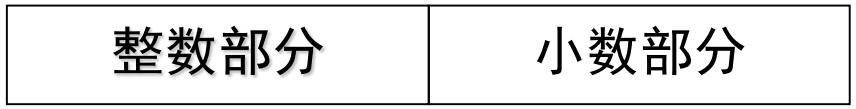


  * 定点整数——小数点位于数值位的最低位



(3)不同类型数据的存储形式不同

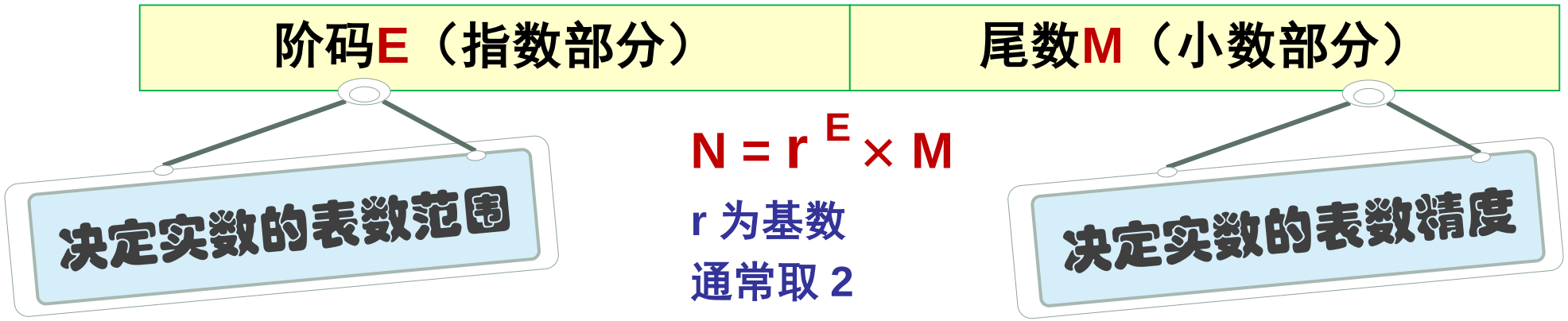
- 问题：如何表示实型数？
  - 小数形式
  - 指数形式——科学计数法



  - 定点数（Fixed Point）
    - 小数点的位置固定
  - 浮点数（Floating-Point）
    - 小数点的位置不固定

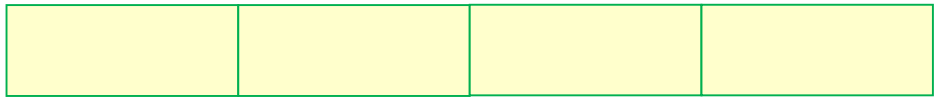
(3)不同类型数据的存储形式不同

* 浮点数实现小数点位置可浮动的主要原因
  * 将实数拆分成了阶码（Exponent）和尾数（Mantissa）分别存储
  * 对于同样的尾数，阶码的值越大，则浮点数所表示的数值的绝对值就越大

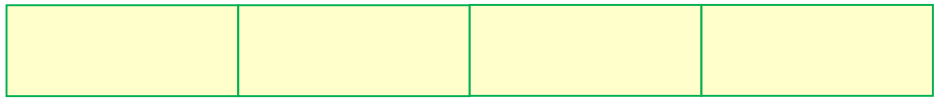


(3)不同类型数据的存储形式不同

- 同样是4个字节（32位）

定点数 

$-2^{31} \sim 2^{31}-1$

单精度浮点数 

$-3.402823466\times10^{38}\sim3.402823466\times10^{38}$

  - 定点数表数范围受其二进制位数的限制——值域都是有限的

- 在计算机中通常是用**定点数**来表示整数和纯小数
- 用**浮点数**表示既有整数部分、又有小数部分的实数

(3)不同类型数据的存储形式不同

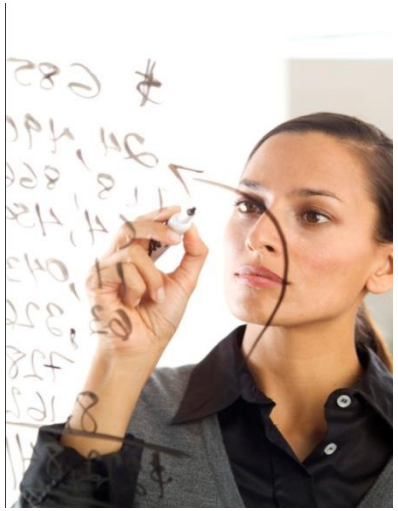
- 字符型数据（英文字母、数字、控制字符）
  - 以二进制编码方式存储，一个字节保存一个字符
- 字符编码方式
  - 取决于计算机系统所使用的字符集
  - ASCII（美国标准信息交换码）字符集
  - 每个字符有一个编码值（查ASCII码表）
  - 字符常数就是一个普通整数

‘H’	0	1	0	0	1	0	0	0	72

十进制 ASCII 码	字　符	十进制 ASCII 码	字　符	十进制 ASCII 码	字　符
0	NUL	43	+	86	V
1	SOH(^A)	44	,	87	W
2	STX(^B)	45	-	88	X
3	ETX(^C)	46	.	89	Y
4	EOT(^D)	47	/	90	Z
5	EDQ(^E)	48	0	91	[
6	ACK(^F)	49	1	92	\
7	BEL(bell)	50	2	93	]
8	BS(^H)	51	3	94	^
9	HT(^I)	52	4	95	-
10	LF(^J)	53	5	96	`
11	VT(^K)	54	6	97	a
12	FF(^L)	55	7	98	b
13	CR(^M)	56	8	99	c
14	SO(^N)	57	9	100	d
15	SI(^O)	58	:	101	e
16	DLE(^P)	59	;	102	f
17	DC1(^Q)	60	<	103	g
18	DC2(^R)	61	=	104	h
19	DC3(^S)	62	>	105	i
20	DC4(^T)	63	?	106	j
21	NAK(^U)	64	@	107	k
22	SYN(^V)	65	A	108	l
23	ETB(^W)	66	B	109	m
24	CAN(^X)	67	C	110	n
25	EM(^Y)	68	D	111	o
26	SUB(^Z)	69	E	112	p
27	ESC	70	F	113	q
28	FS	71	G	114	r
29	GS	72	H	115	s
30	RS	73	I	116	t
31	US	74	J	117	u
32	Space(空格)	75	K	118	v
33	!	76	L	119	w
34	"	77	M	120	x
35	#	78	N	121	y
36	$	79	O	122	z
37	%	80	P	123	{
38	&	81	Q	124	|
39	'	82	R	125	}
40	(	83	S	126	~
41	)	84	T	127	del
42	*	85	U		

(4)不同数据类型可参与的运算不同

- 整型
  - 加、减、乘、除、求余
- 实型
  - 加、减、乘、除
- 字符型
  - 加、减（整数）
  - 对ASCII码值的运算
- 指针类型
  - 加、减（整数）和比较运算



小结

- 不同类型的变量
  - 占用内存空间的大小不同
    - 用sizeof运算符计算变量占内存空间
  - 数据在内存中的存储形式不同
  - 合法的表数范围不同
  - 可参与的运算种类 不同

讨论

- 计算机为什么采用浮点数而非定点数来表示实数？
- 浮点数是实数的精确表示吗？
- 既然浮点数相对于整数能够表示更大的数，那么是否可以用浮点数取代整数呢？