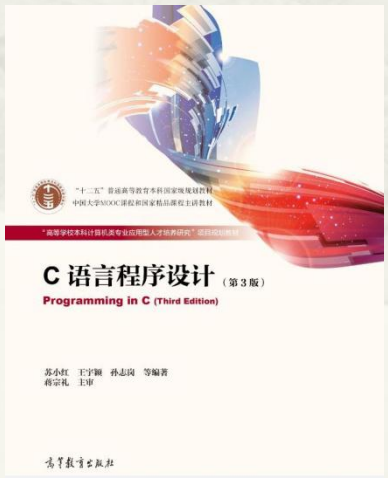
哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家

高级语言程序设计

哈尔滨工业大学（深圳）

计算机科学与技术学院



C语言什么样？

简单的C程序：　“Hello world!”

```
#include <stdio.h>
main()
{
        printf("Hello  world!\n");
}
```

复杂点：加法计算器

```
#include <stdio.h>

int Add(int a, int b)
{
   return (a + b);
}

main()
{
   int  x, y, sum = 0;

   printf("Input two integers:");
   scanf("%d%d", &x, &y);
   sum = Add(x, y);
   printf("sum = %d\n", sum);
}
```

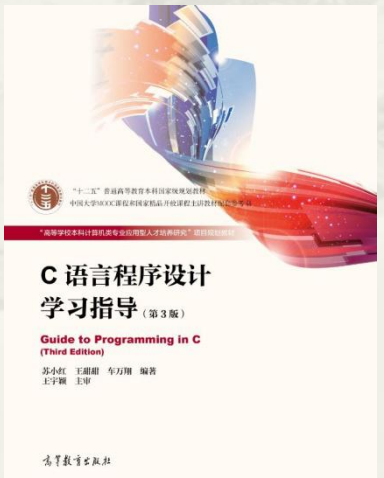
高级语言程序设计要学什么？

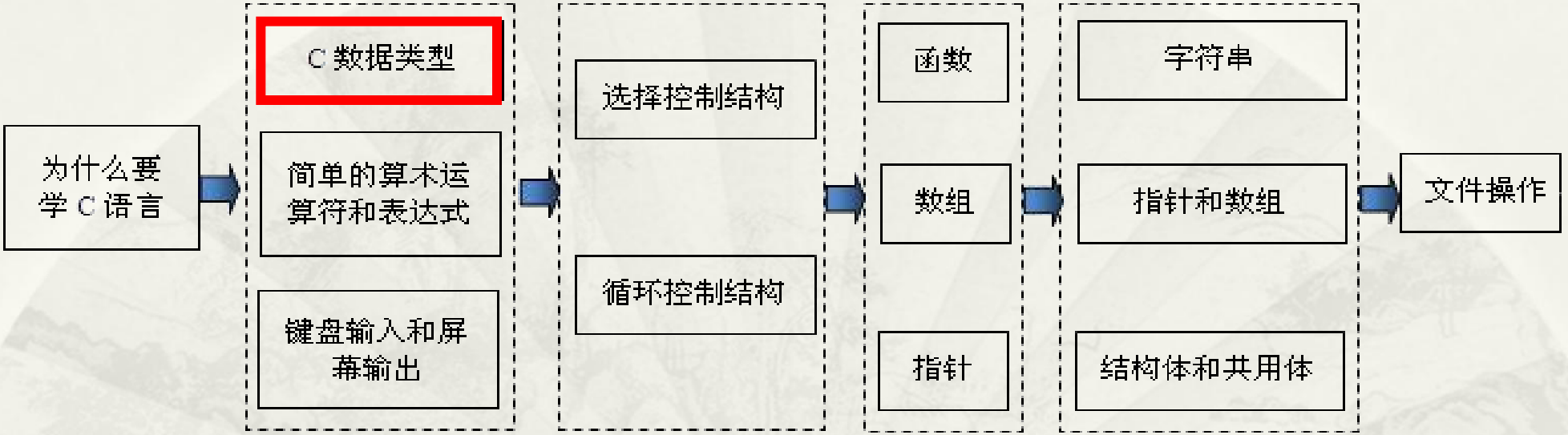
- 要学会和计算机进行沟通交流时要遵从的语法
  - C语言有C的语法、Java语言有Java语言的语法, 等等
- 要理解每种语法后面的语义
  - 虽然每种语言的语法各有不同，但很多语言都有相同或相近的语义，这种语义，往往是我们想要传达给计算机意图
- 要建立起从计算机的角度来思考问题
  - 要能够在大脑中直接用计算机的语言来进行思考
  - 这实际上是思维方式的一种改变

语言学习Learning与习得Acquiring

* 语言学习？No！
* 只有把语言内化成自己的内在的思维模式，我们才叫习得了一门语言，语言习得是能力的培养而不是技能的学习。
* 语言习得的四个层次
  * **词汇**：高级语言往往只有有限的关键词和词语命名规范（靠自己背诵）
  * **语法、句法**：不同的语言结构对应不同的语法（靠自己记忆、掌握）
  * **语义**：进入计算机世界的桥梁，理解每个词语、语法背后对应的计算机系统的变化，是我们课程讲授的重点
  * **语用**：将语言能力从**流畅交流**升华到**职业作家**！

第2章 C数据类型



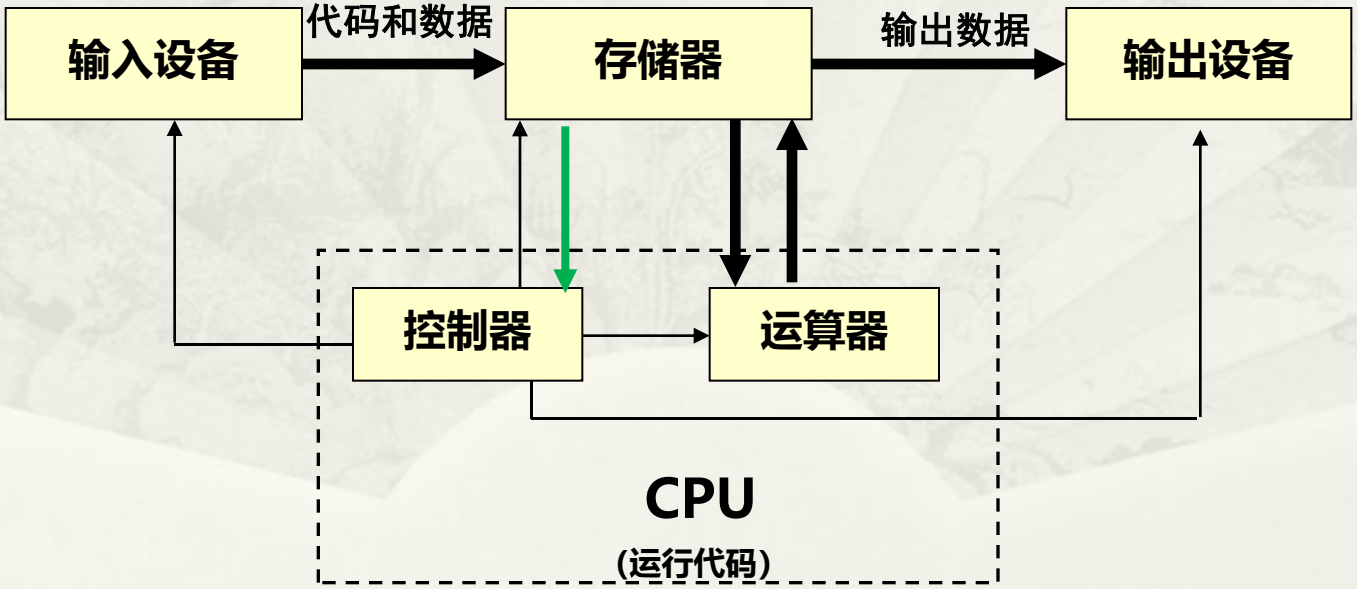

为什么要学C语言
C数据类型
简单的算术运算符和表达式
键盘输入和屏幕输出
选择控制结构
循环控制结构
函数
数组
指针
字符串
指针和数组
结构体和共用体
文件操作

第2章　　学习内容

- 在高级语言中为什么要引入数据类型
- 常量和变量
- 如何定义变量和为变量赋值

问题1：数据在计算机中是如何存储的？

- 冯·诺依曼计算机
  - 指令和数据都同样存储在内存中
  - 都以二进制（Binary）形式存储在内存中



问题2：为什么用二进制存储？

- 为什么用二进制存储，不用十进制？
  - 二进制在电器原件中容易实现，双稳态文件很多
    - 电压的高和低、电容器的充电与放电、脉冲的有与无、晶体的导通与截至
  - 二进制运算比进行十进制运算简单得多
  - 二进制易于实现物理上对数据的存储（如光盘等），且与逻辑判断正好可以形成一一对应关系。
    - 用1表示“真”，用0表示“假”
  - 诠释了计算机的哲学——复杂事物由简单事物构成

问题3：何为二进制？

- 十进制
  - 用0-9这十个数中的一个表示十进制的一位数
  - “逢10进1”的进位原则，基为10
  - 每位数字都有一个权值，是10的幂次
  - 十进制表示的数值可写成按位权展开的多项式之和
    - 例如，十进制数123.45可表示为
    - $1\times10^{2}+2\times10^{1}+3\times10^{0}+4\times10^{-1}+5\times10^{-2}$

问题3：何为二进制？

- 二进制数
  - 用0或1表示二进制的一位数
  - “逢2进1”的进位原则，基为2
  - 每位数字都有一个权值，是2的幂次
  - 二进制表示的数值可写成按位权展开的多项式之和
    - 例如，二进制数101.11转换为十进制数就是
    - $1\times2^{2}+0\times2^{1}+1\times2^{0}+1\times2^{-1}+1\times2^{-2}$，即5.75

问题4：如何表示二进制的正与负？

- 有符号和无符号整数的区别在于怎样解释其最高位（The Most Significant Bit, MSB)
  - 对无符号整数，其最高位被C编译器解释为数据位
  - 对有符号整数，其最高为被解释为符号位（0为正，1为负）
- 0怎么办？0的表示还能唯一吗？
  - ＋0的二进制是00000000 00000000
  - －0的二进制是10000000 00000000
- 负数以二进制补码（Complement）形式存储
  - 便于用统一的形式来表达0
  - 便于将减法运算也转化为加法运算来处理

问题4：如何表示二进制的正与负？

- 补码如何计算？
  - 正数的反码、补码与其原码都是相同的
  - 对于负数，保持符号位不变，原码→反码→反码+1 →补码
  - +0和-0的补码是相同的

问题4：如何表示二进制的正与负？

- -1的补码如何计算？
  - 将最高位解释为符号位，则该数就是-1
  - 将最高位解释为数据位，则该数就是

$1\times2^0+1\times2^1+1\times2^2+1\times2^3+1\times2^4+1\times2^5+1\times2^6+1\times2^7+1\times2^8+1\times2^9$
$+1\times2^{10}+1\times2^{11}+1\times2^{12}+1\times2^{13}+1\times2^{14}+1\times2^{15}=65535$

问题5：数据在程序中是如何表示的？

- 十进制，二进制的压缩表示：八进制，十六进制
- 二进制和八进制如何相互转换？
  - 二进制→八进制（3位一组）

十进制、二进制、八进制数之间的关系对应表

十进制	0	1	2	3	4	5	6	7
二进制	000	001	010	011	100	101	110	111
八进制	0	1	2	3	4	5	6	7

问题5：数据在程序中是如何表示的？

- 二进制和十六进制如何相互转换？
  - 二进制→十六进制（4位一组，以ABCDEF分别表示10-15）
  - 为什么4位一组？

十进制、二进制、十六进制数之间的关系对应表

十进制	0	1	2	3	4	5	6	7
二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	0	1	2	3	4	5	6	7
十进制	8	9	10	11	12	13	14	15
二进制	1000	1001	1010	1011	1100	1101	1110	1111
十六进制	8	9	A	B	C	D	E	F

问题6：计算机的内存是如何编址的？

- 内存——计算机内的存储部件
- 内存的特点
  - 速度快、可随机访问，但掉电即失
- 内存中的存储单元是一个线性地址表
- 内存地址按字节（Byte）编址
  - 每个字节都用唯一的一个整数字来标识——地址（Address）
  - 地址是一个十六进制无符号整数
    - 32位计算机的内存地址编码是32位，从0x00000000到0xFFFFFFFF

问题7：如何衡量内存空间的大小？

- 计算机内存数据的最基本单元
  - 衡量物理存储器容量的最小单位——位，也称比特
- 一个位（bit）有多大？
  - 只能是0或者1，二进制
    - 世界上有10种人，1种人懂二进制，1种人不懂二进制
  - 一个位无法表示太多数据，所以须将许多位合起来使用
- 8位（bit）可表示0~255间的整数
  - 字节（Byte）

问题8：衡量内存容量的单位是什么？

* 字节（Byte）是最小的可寻址的存储器单位
  - 通常用字节数的多少来衡量内存空间的大小
  - 标识字符的最小单位

衡量内存空间大小的表示单位

英文谓词	中文称谓	换算方法
Bit(b)	(比特)位	
Byte(B)	字节	1 B=1 b
Kilobyte(KB)	千	1KB = 1024 B
Megabyte(MB)	兆	1 MB = 1024 KB
Gigabyte(GB)	吉	1GB = 1024 MB
Terabyte(TB)	太	1 TB = 1024 GB

1 TB = 1024 GB
1 GB = 1024 MB
1 MB = 1024 KB
1 KB = 1024 B
1 B = 8 b

问题9：在高级语言中为什么要引入数据类型？

- 对计算机系统和硬件本身而言，数据类型（Data Type）的概念其实是不存在的
  - 冯·诺依曼体系结构中，程序代码和数据以二进制存储
- 引入数据类型的主要目的
  - 有效地组织数据，把数据分成所需内存大小不同的数据
  - 规范数据的使用
  - 提高程序的可读性
  - 方便用户的使用

问题10：C语言中有哪些数据类型？

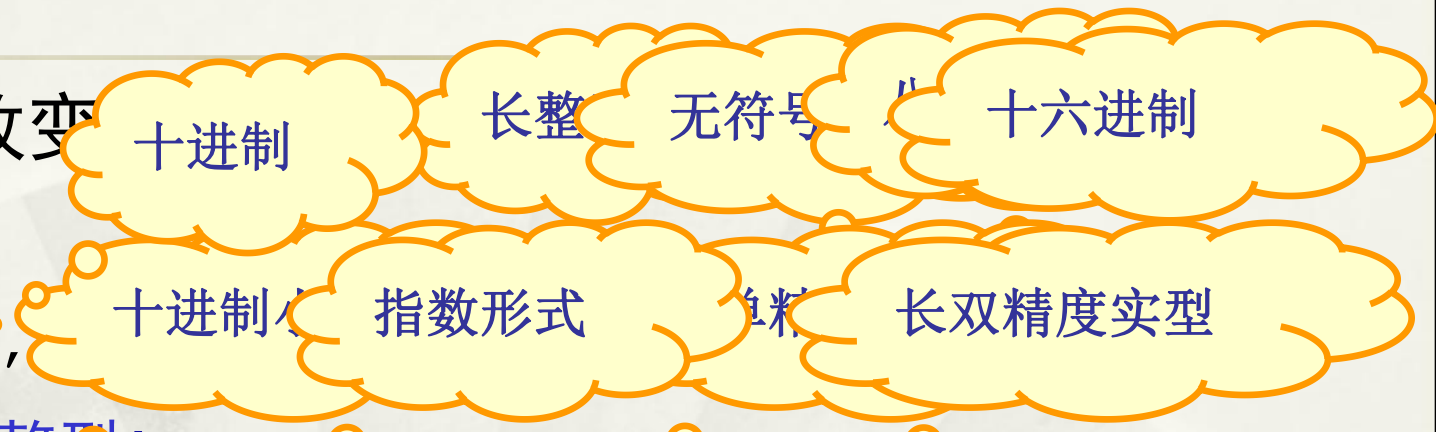
- 数据类型
  - 基本类型
    - 整型
      - 基本整型**int**
      - 长整型**long int**
      - 短整型**short int**
      - 无符号整型**unsigned**
    - 实型（浮点型）
      - 单精度实型**float**
      - 双精度实型**double**
      - 长双精度实型**long double**
    - 字符型**char**
    - 枚举类型
  - 构造类型
    - 数组类型
    - 结构体类型
    - 共用体类型
  - 指针类型
  - 空类型

2.1常量与变量

- 问题11：C语言程序中有哪几种数据形式？
  - 常量（Constant）
    - 在程序中不能改变其值的量
  - 变量（Variable）
    - 其值在程序执行过程中是可以改变的

问题12：常量包括哪几种类型？

- 在程序中不能改变
- 包括：
  - 整型（如 0,67,
    - 默认为基本整型int
  - 实型（如 2.3, 1.2e-5, 2.73F, 2.73L）
    - 默认为双精度实型double
  - 字符型（如 'z', '3', '$'）
    - 字符串（如 "UKM", "1", "5a"）
  - 枚举型



问题13：如何声明变量的类型？

- 变量的值在程序执行过程中是可以改变的
- 变量的声明(Variable Declaration)

  类型关键字　变量名;

- 使用变量的基本原则
  - 变量必须先声明，后使用
- 一条声明语句可声明若干个同类型的变量
  - `int a, b, c;`
- 声明的顺序无关紧要

```
1    main()
2    {
3        int   a;
4        float b;
5        char  c;
6        a = 1;
7        b = 2.5;
8        c = 'A';
9    }
```

问题14：如何给变量赋值？

(1)声明变量的同时为变量赋值(Assignment)

——变量的初始化（Initialize）

* 未被初始化的变量的值会是什么？
  * 其值为随机数（乱码）

```
int    a = 1;
float  b = 2.5;
char   c = 'A';
```

问题14：如何给变量赋值？

(2) 赋值表达式语句：

变量 = 表达式 ；

规则：

变量 ← 表达式　　左值和右值类型一致

Valid Example:

```
int a;
a = 0;
```

Invalid Example:

```
int a;
a = 5.75;
```

问题14：如何给变量赋值？

- 简单赋值（Simple Assignment）
- Syntax:

变量　　=　　表达式 ;

每个赋值表达式都有一个值

问题14：如何给变量赋值？

- 多重赋值（Multiple Assignment）
- Syntax:

变量1 = 变量2 = 表达式；

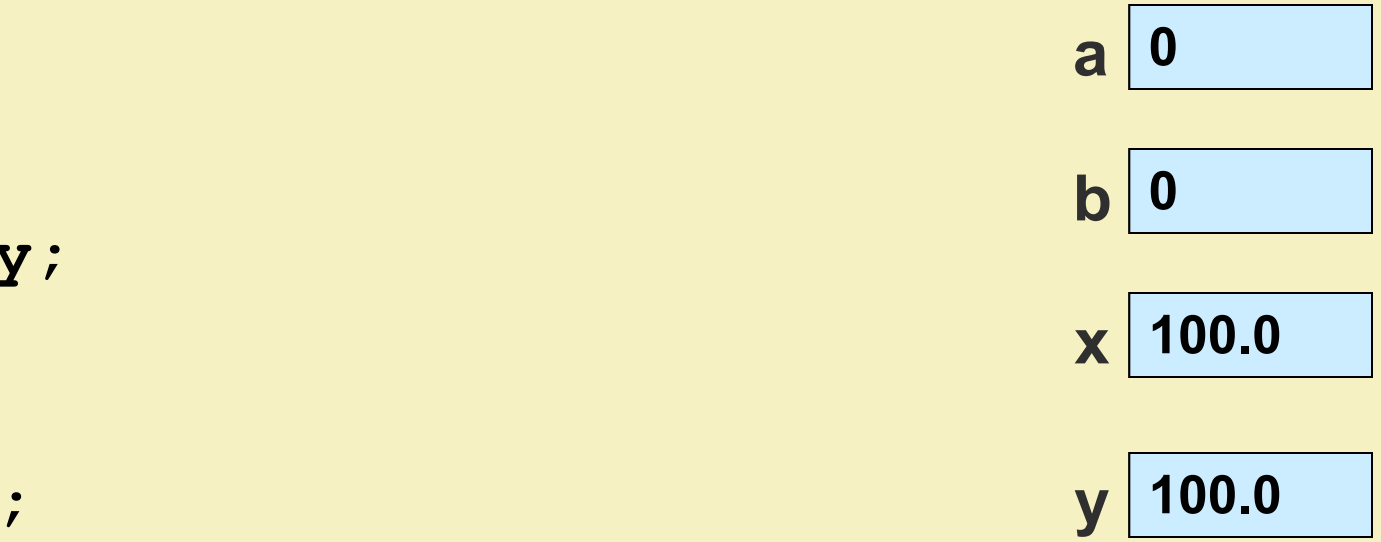
变量1 =（变量2 = 表达式）；

从右向左赋值

问题14：如何给变量赋值？

Example:

```
int a, b;
float x, y;
. . .
a = b = 0;
x = y = 100.0;
```



问题15：变量有哪些属性?

- 变量的类型(Type)—决定变量被分配内存空间的大小

```
int  a = 0;
```

⟺

```
int a;
a = 0;
```

变量的地址(Address)

变量被分配的内存大小

0x0037b000

变量的值(Value)

a

变量名(Name)

00011000
00000111
00000011
00000110
10000000
00000001
00000100
00100001
00001000
00100000
01000100

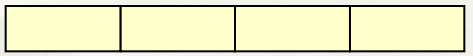
某存储区域

2.3数据类型（Data Type）

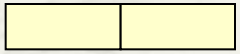
- 问题16：变量的类型决定了什么？
  - 占用内存空间的大小
  - 数据的存储形式
  - 合法的取值范围
  - 可参与的运算种类

(1)不同类型数据占用的内存大小不同

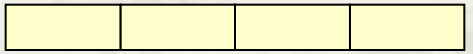
- **int**——基本整型，C标准未规定，系统相关
  - 在目前大多数系统上占4个字节



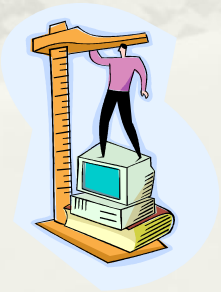
- **short int，简写为short**
  - 短整型，2个字节



- **long int，简写为long**
  - 长整型，4个字节

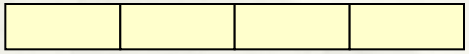


  - **unsigned**——无符号整型（正整数和0）
  - **用来修饰int、short和long**

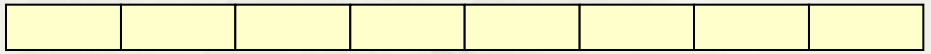


(1)不同类型数据占用的内存大小不同

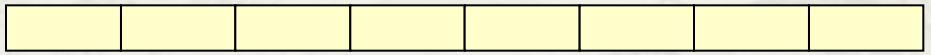
- **float**
  - 单精度实型，4个字节



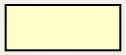
- **double**
  - 双精度实型，8个字节

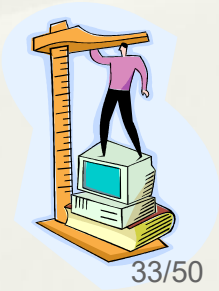


- **long double**
  - 长双精度实型，系统相关
  - **VC++中占8个字节**



- **char**
  - 字符型，1个字节





(2)不同类型数据的存储形式不同

- **Intel CPU上的整型数**

低位字节	高位字节

- 实型数
  - 定点数（Fixed Point） ---------- 12.34567
  - 浮点数（Floating Point）---------- $12.34567*10^0$

$123456.7*10^{-4}$

$1.234567*10^1$

阶码j（指数部分）	尾数S（小数部分）

阶码符号	阶码的数值	尾数符号	尾数的数值

$N = S \times r^j$　　r 为基数，通常取 2 或 10 等

(2)不同类型数据的存储形式不同

- ANSI C未规定3种浮点类型的长度、精度和表数范围
- 有的系统使用更多的位来存储小数部分（尾数）
  - 增加了数值有效数字位数，提高了数值精度，但表数范围缩小
- 有的系统使用更多的位存储指数部分（阶码）
  - 扩大了变量值域（即表数范围），但精度降低
- 浮点数并非真正意义上的实数，只是其在某种范围内的近似

阶码j（指数部分）		尾数S（小数部分）	
阶码符号	阶码的数值	尾数符号	尾数的数值

$N=S\times r^{j}$

(2)不同类型数据的存储形式不同

- 字符型数据以二进制编码方式存储在内存中
  - 一个字节保存一个字符（英文字母、数字、控制字符）
  - 字符常数就是一个普通整数
- 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
  - 每个字符具有一个0~127之间的编码值
  - 可从ASCII码表中查出
  - 最高位被用作奇偶校验位

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

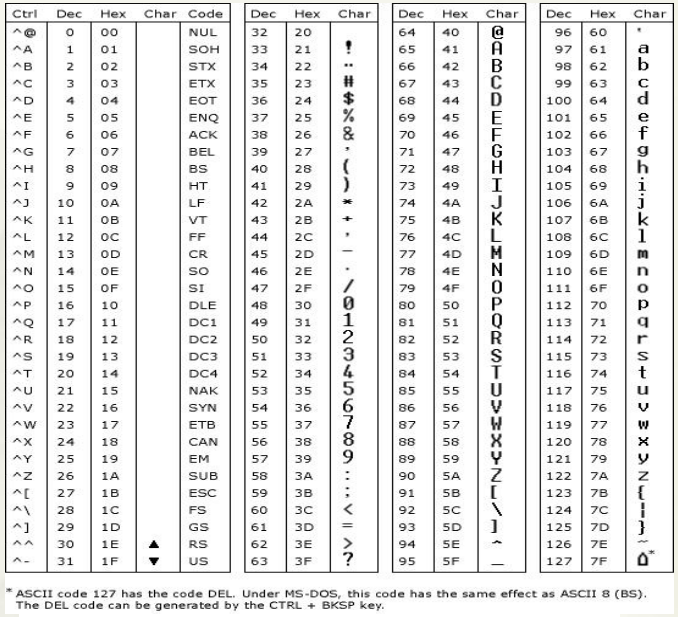
Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂*

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

(2)不同类型数据的存储形式不同

- 奇偶校验
  - 校验代码传输过程中是否出现错误
  - 偶校验规定，若一个字节的编码序列中1的个数是奇数，则校验位置1，否则置0
  - 奇校验与其正好相反
- ASCII扩展码
  - 利用了最高位，值扩展到128~255
  - 存放英文制表符、部分音标字符和其他符号



Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

(2)不同类型数据的存储形式不同

- **问题17：汉字如何存储？
  - 汉字编码，兼容ASCII码，连续的2个字节，仅在其第7位均为1时认为是汉字
  - GB2312（国标2312），6763字
  - GB13000.1，20902字
  - GB18030，27533字
  - BIG5，13000字
- 其他国家的语言文字呢？

(3)不同数据类型的取值范围不同

Visual C++下各数据类型所占字节数和取值范围

数据类型	所占字节数（bytes）	取值范围
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}

Visual C++中，单精度占4个字节，而双精度和长双精度型都占8个字节

(3)不同数据类型的取值范围不同

- Most significant bit(MSB) is sign(最高位为符号位)
- 有符号整数和无符号整数的取值范围不同

Type	Min value	Max value
short int	-32768 (-2^{15})	32767 ($2^{15}-1$)
unsigned short int	0	65535 ($2^{16}-1$)
unsigned char	0	255

(3)不同数据类型的取值范围不同

- 问题18：当向变量赋超出其表数范围的数值时，结果会怎样呢？
  - Assign a short with larger number?
  - 产生数值溢出（Overflow），得到奇怪的结果

(3)不同数据类型的取值范围不同

- 生活中数值溢出的例子
  - 20世纪末的千年虫问题——对年份仅记录其后两位
    - 在99年存钱，到01年取出，该怎样计算利息呢？
  - 第一代身份证号码中的出生年——百岁老人？婴儿？
  - 1996年，阿丽亚娜火箭因浮点数转换成整数发生溢出而导致发射失败

(3)不同数据类型的取值范围不同

- 数值溢出的危害
  - 编译器对它熟视无睹
  - 在平台间移植时会出现问题，导致数据丢失或者溢出
  - 当程序从高位计算机向低位计算机移植（比如从64位系统移植到32位系统）时，可能出现溢出

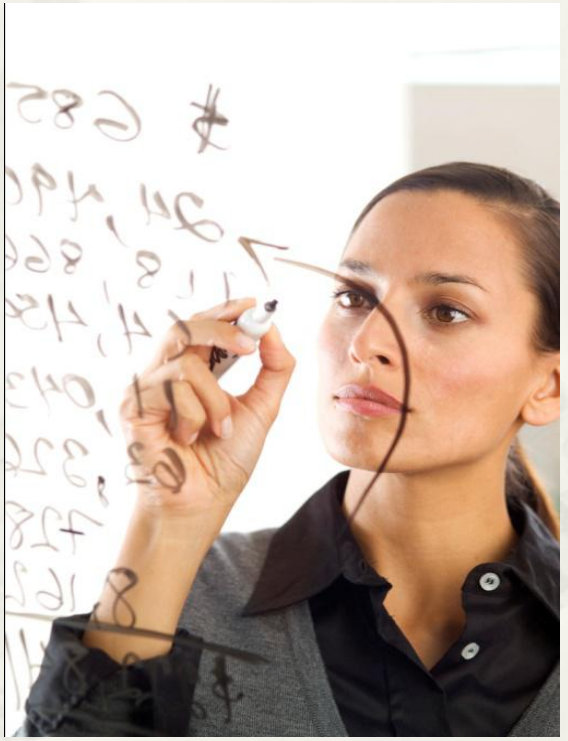
(3)不同数据类型的取值范围不同

- 对策
  - 了解处理问题的规模，选择取值范围更大的变量类型
  - 同种类型在不同的平台其占字节数不尽相同
  - 不要对变量所占的内存空间字节数想当然
  - 用sizeof获得变量或者数据类型的长度

Type	Usual size	Supercomputer
int	32 bits	64 bits
short int	16 bits	32 bits
long int	32 bits	128 bits
float	32 bits	64 bits
double	64 bits	128 bits
char	8 bits	

(4)不同数据类型可参与的运算不同

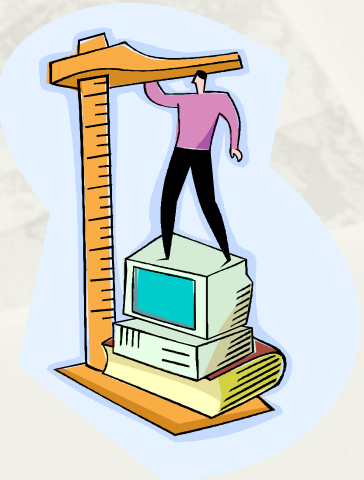
- 整型
  - 加、减、乘、除、求余
- 实型
  - 加、减、乘、除
- 字符型
  - 加、减（整数）
  - 对ASCII码值的运算



2.4如何计算变量或类型占内存的大小

- 问题19：如何计算变量占内存空间的大小？
  - 用sizeof运算符
- C运算符，并非函数
  - 计算类型占用的字节数
- 一般用：**sizeof(变量名)**

语法形式	运算结果
sizeof(类型)	类型占用的字节数
sizeof(表达式)	表达式值所属类型占用的字节数



例2.2
在TC和VC、CB下的运行结果

```
#include  <stdio.h>
main()
{
      printf("Data type       Number of bytes\n");
      printf("------------ ----------------------\n");
      printf("char                %d\n",sizeof(char));
      printf("int                 %d\n",sizeof(int));
      printf("short int           %d\n",sizeof(short));
      printf("long int            %d\n",sizeof(long));
      printf("float               %d\n",sizeof(float));
      printf("double              %d\n",sizeof(double));
}
```

C程序的开发工具

- **Visual C++**
  - Windows平台上最流行的C/C++集成开发环境之一
- **Code::Blocks**（简称CB，http://www.codeblocks.org）
  - 是近年出现并获得关注的C/C++开发环境
  - 免费，开放源码，跨平台
  - **CB只是一个IDE（Integrated Development Environment，集成开发环境），没有内置的编译器和调试器**
  - 但可支持多种编译器，例如GCC编译器和GDB调试器
  - **Code::Blocks + GCC (Compiler) + GDB (Debuger)**
  - **http://book.sunner.cn**

认识C语言从运行这个程序开始

■ 第一个程序范例——打印"Hello World!"

```
#include <stdio.h>
main()
{
    printf("Hello  world!\n");
}
```

以#开头，编译预处理指令

头文件

Every C program must have a main function
The execution of C program starts from main() function

■ 考考你：

– 如何把"Hello"和"world!"分别打印在两行？

问题20：C程序中有哪些常见符号？

- 关键字（Keyword）
  - 又称保留字（Reserved Word）
    - A word that has special meaning in C
- 标识符（Identifier）
  - 系统预定义标识符
    - A word having special meaning but may be redefined (but is not recommended!!)
  - 用户自定义标识符
    - 变量名，函数名，……

```
#include <stdio.h>
/*This is an example*/
int Add(int a, int b)
{
   return (a + b);
}

main()
{
   int x, y, sum = 0;

   printf("Input two integers:");
   scanf("%d%d", &x, &y);
   sum = Add(x, y);
   printf("sum = %d\n", sum);
}
```

问题20：C程序中有哪些常见符号？

- 运算符（Operator）
  - 详见附录C
- 分隔符（Separator）
  - 空格、回车/换行、逗号等
- 其他符号
  - {和}标识函数体或语句块
  - /*和*/或//是程序注释的定界符
- 常量（Constant）

```
#include <stdio.h>
/*This is an example*/
int Add(int a, int b)
{
   return (a + b);
}

main()
{
   int  x, y, sum = 0;

   printf("Input two integers:");
   scanf("%d%d", &x, &y);
   sum = Add(x, y);
   printf("sum = %d\n", sum);
}
```

小结

- 常量和变量
- 整型和实型
- 如何定义一个变量
- 如何为变量赋值
- **sizeof**运算符


Thank you for
your attention !
SuXiaoHong


Q&A

- https://www.bilibili.com/video/av5345693?from=search&seid=8037982121916559391