第六章
数据类型和表达式 (1)

主要内容

- 数据的存储和基本数据类型
- 数据的输入和输出
- 类型转换
- 表达式

主要内容

- 数据的存储和基本数据类型
- 数据的输入和输出
- 类型转换
- 表达式

C语言的数据类型

- 基本数据类型
  - 整型
  - 实型（浮点型）
  - 字符型
- 构造数据类型
  - 数组、结构、联合、枚举
- 指针类型
- 空类型（void）

后续学习

基本数据类型的存储

- 整型
- 实型
- 字符型

整型数据存储

- 整数的第一位bit用于表示整数的符号

  1 - 负数

  0 - 正数

  1 000 0001 1000 0001

  0 000 0001 1000 0001

原码、反码、补码

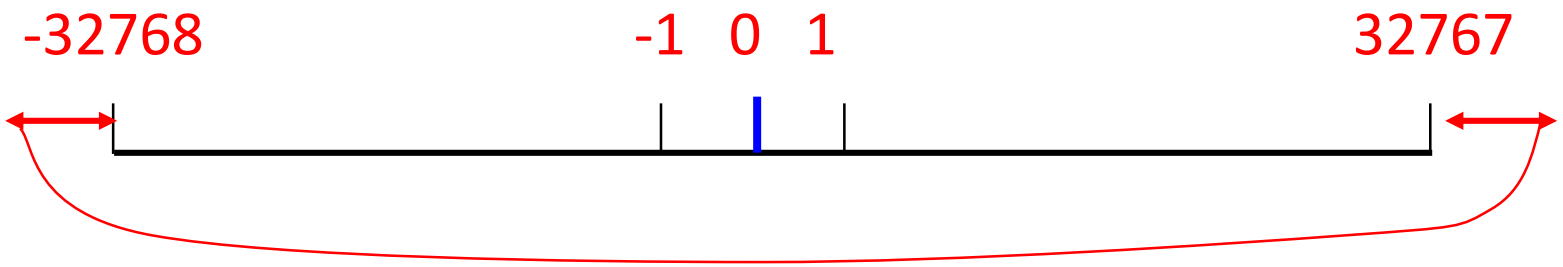
- 正数的原码、反码和补码**相同**
  - 1 的补码　　0 000 0000 0000 0001
  - ……
  - 32767 的补码　0 111 1111 1111 1111

  (2^{15}-1，2个字节的存储单元能表示的最大正数)
- 负数的原码、反码和补码**不同**
  - -1
  - 原码　1 000 0000 0000 0001
  - 反码　1 111 1111 1111 1110　　原码取反
  - 补码　1 111 1111 1111 1111　　反码 + 1

原码、反码、补码

- 32767 = 2^{15}-1
  - 补码　0 111 1111 1111 1111
- -32767
  - 原码　1 111 1111 1111 1111
  - 反码　1 000 0000 0000 0000　　原码取反
  - 补码　1 000 0000 0000 0001　　反码 + 1
- -32768 = -32767-1
  - 补码　1 000 0000 0000 0000
  - = -2^{15}，**2个字节**的存储单元能表示的**最小负数**

2字节整数的补码表示



32767	0111 1111 1111 1111
32766	0111 1111 1111 1110
32765	0111 1111 1111 1101
……	
1	0000 0000 0000 0001
0	0000 0000 0000 0000
–1	1111 1111 1111 1111
–2	1111 1111 1111 1110
……	
–32767	1000 0000 0000 0001
–32768	1000 0000 0000 0000

32767 + 1 = ?

0111 1111 1111 1111

+1

= 1000 0000 0000 000

（–32768）

–32768 – 1 = ???

1000 0000 0000 0000

–1

= 0111 1111 1111 1111

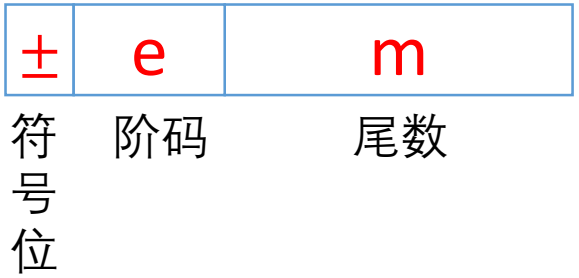
（32767）

浮点型数据存储

- 实型数据的存储

$x = \pm m * r^e$

m – 尾数
r – 基数
e – 阶码



- IEEE754标准规定，常用的浮点数的格式为：

	符号位	阶码	尾数	总位数
单精度浮点数	1	8	23	32
双精度浮点数	1	11	52	64
临时浮点数 (扩展精度浮点数)	1	15	64	80

字符型数据存储

- 占据一个字节
  - 存储ASCII码
  - 2^8=256个ASCII字符

基本数据类型

- 整型

有符号整型	无符号整型	数据长度
int	unsigned [int]	16位或32位
short [int]	unsigned short [int]	16位
long [int]	unsigned long [int]	32位

- 字符型

  char 8位

- 实型（浮点型）

  单精度浮点型 float 32位

  双精度浮点型 double 64位

扩展的整数类型
short, long, unsigned [int]

有符号整型	无符号整型	数据长度
int	unsigned [int]	16位或32位
short [int]	unsigned short [int]	16位
long [int]	unsigned long [int]	32位

short (有符号)

MIN 1 000 0000 0000 0000 -32768(-2^{15})

MAX 0 111 1111 1111 1111 32767($2^{15}-1$)

unsigned short(无符号)

MIN 0000 0000 0000 0000 0

MAX 1111 1111 1111 1111 65535($2^{16}-1$)

整数类型的取值范围

- int 32位 $[-2^{31}, 2^{31}-1]$
- short [int] 16位 $[-2^{15}, 2^{15}-1]$
- long [int] 32位 $[-2^{31}, 2^{31}-1]$

- unsigned [int] 32位 $[0, 2^{32}-1]$
- unsigned short [int] 16位 $[0, 2^{16}-1]$
- unsigned long [int] 32位 $[0, 2^{32}-1]$

整数常量

- 后缀
  - L、l 表示long [int]，如126L、126l
  - U、u表示unsigned [int]，如12u、12U
  - LU、lu表示unsigned long [int]，如4 294 967 295 LU
- 无后缀，根据整数常量的值判断其类型
  - int 32位 $[-2^{31}, 2^{31}-1]$
  - short [int] 16位 $[-2^{15}, 2^{15}-1]$
  - long [int] 32位 $[-2^{31}, 2^{31}-1]$
  - unsigned [int] 32位 $[0, 2^{32}-1]$
  - unsigned short [int] 16位 $[0, 2^{16}-1]$
  - unsigned long [int] 32位 $[0, 2^{32}-1]$

基本数据类型 - 字符型

- 小写字母：'a' 'b' 'c' … 'z'
- 大写字母：'A' 'B' 'C' … 'Z'
- 数字：'0' '1' '2' … '9'
- 括号、标点符号、运算符

  ( ) { } , . ' " ! # @

  + - * / % > < =

  等等

基本数据类型 - 字符型

- 转义字符
  - 换行符 \n
  - 制表符 \t
  - 反斜杠 \\
  - 双引号 \"
  - 单引号 \'
  - \ddd 1-3位八进制码代表的字符
  - \xhh 1-2位十六进制码代表的字符
- 附录B（ASCII码表，P351）

- %就是%，不是\%
- 在scanf和printf函数中%具有特殊作用：将其后的字符解释为格式字符)
- 所以用%%表示字符%本身

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	**!**
34	22	100010	42	**"**
35	23	100011	43	**#**
36	24	100100	44	**$**
37	25	100101	45	**%**
38	26	100110	46	**&**
39	27	100111	47	**'**
40	28	101000	50	**(**
41	29	101001	51	**)**
42	2A	101010	52	*****
43	2B	101011	53	**+**
44	2C	101100	54	**,**
45	2D	101101	55	**-**
46	2E	101110	56	**.**
47	2F	101111	57	**/**
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	**:**
59	3B	111011	73	**;**
60	3C	111100	74	**<**
61	3D	111101	75	**=**
62	3E	111110	76	**>**
63	3F	111111	77	**?**
64	40	1000000	100	**@**
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	**[**
92	5C	1011100	134	****
93	5D	1011101	135	**]**
94	5E	1011110	136	**^**
95	5F	1011111	137	**_**
96	60	1100000	140	**`**
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	**{**
124	7C	1111100	174	**|**
125	7D	1111101	175	**}**
126	7E	1111110	176	**~**
127	7F	1111111	177	*[DEL]*

基本数据类型 - 字符型

- 字符具有数值特征（值为ASCII码的整数）

  'A' 65 0100 0001

- 适用：算术运算、关系运算

- 整型变量和字符变量的定义和赋值可以互换

  【ASCII码范围内】

  ```
  char  c;
  c = 'A'; 或 c = 65;
  c+1 就是字符'B'
  ```

基本数据类型 - 实型

- 实型（浮点型）数据
- 单精度浮点型 float
- 双精度浮点型 double

	存储	数据精度 (有效数字)	取值范围
float	4字节	7/8位	$\pm(10^{-38} \sim 10^{38})$
double	8字节	15/16位	$\pm(10^{-308} \sim 10^{308})$

数据精度和取值范围

- 数据精度 与 取值范围是两个不同的概念：
  - float x = 1234567.89; ×
    - 虽在取值范围内，但无法精确表达。
  - float y = 1.2e55; ×
    - y 的精度要求不高，但超出取值范围。
- 并非所有实数都能在计算机中精确表示

实数的常量表示

- 普通表示

  -12345.678

  符号+整数部分+小数点+小数部分

- 科学计数法表示

  -1.2345678E5

- 实型常量的类型都是double

- 用f作为后缀，表示浮点数常量

  3.14f

主要内容

- 数据的存储和基本数据类型
- 数据的输入和输出
- 类型转换
- 表达式

数据的输入输出

printf (格式控制字符串, 输出参数1, ... , 输出参数n);

scanf (格式控制字符串, 输入参数1, ... , 输入参数n);

格式控制字符串： “%d%f%c” “k = %d, x = %f, h = %c”

输入数据：变量名称取地址 “&”

- 格式控制说明符 %
  - 字符char： %c
  - 实数float： %f
  - 实数doulbe： %lf
  - 整数int： %d

整型数据的输入输出

- 扩展整数的格式控制符

	十进制	八进制	十六进制
int	%d	%o	%x
long	%ld	%lo	%lx
unsigned	%u	%o	%x
unsigned long	%lu	%lo	%lx

示例：整型数据输出格式

```
# include <stdio.h>
void main(void)
{
    printf("%d, %o, %x\n", 10, 10, 10);
    printf("%d, %d, %d\n", 10, 010, 0x10);
    printf("%d, %x\n", 012, 012);
}
```

运行结果是什么？

10, 12, a
10, 8, 16
10, a

八进制整型常量：以数字0开头，由0~7组成

十六进制整型常量：以0X或0x开头，由0~9、A~F或a~f 组成

输出格式的宽度控制

```
int a, b;
scanf("%o%d\n", &a, &b);
printf("%d %5d\n", a, b);
```

如果输入　　17 17　　运行结果是什么？

15　17

- 宽度控制 %md 表示：数据输出宽度为m，含符号位
  - 若实际宽度不足m个，左边补充空格
  - 若大于m，则按照实际宽度输出
  - %-md：左对齐，其它如上

实型数据的输入和输出

- float：%f 或 %e
  - 以小数或指数形式输入一个单精度浮点数
- double： %lf 或 %le
  - 以小数或指数形式输入一个双精度浮点数
- 输出 printf()
  - float 和double使用相同的格式控制说明
  - %f：小数形式输出浮点数，保留6位小数
  - %e：指数形式输出，小数点前有且仅有1位非零数字

实型数据输出示例

```
double d = 3.1415926;

printf("%f,□%e\n", d, d);
printf("%5.3f,□%-5.2f,□%.2f\n", d, d, d);
```

左对齐，不足位数右边补空格

一共5位，小数2位，小数点1位

右对齐，不足位数左边补空格

一共5位，小数3位，小数点1位

```
3.141593, 3.141593e+00
3.142,□3.14□,□3.14
```

实型数据输入输出示例

```
/*假定float的精度为7位，double的精度为16位*/
# include <stdio.h>
int main(void)
{
    float f;
    double d;

    printf("input f, d:");
    scanf("%f%lf", &f, &d);
    printf("f = %f\n d = %lf \n", f, d);
    d = 1234567890123.12;
    printf("d = %lf \n", d);

    return 0;
}
```

```
input f, d:
1234567890123.123456
1234567890123.123456
f = 1234567954432.000000
d = 1234567890123.123500
d = 1234567890123.120100
```

字符型数据输入输出

- scanf() 和 printf()

  %c

  ```
  char ch;
  scanf("%c", &ch);
  printf("%c", ch);
  ```

- getchar() 和 putchar()

  ```
  char ch;
  ch = getchar( );
  putchar(ch);
  ```

  输入输出一个字符

输入输出字符示例

```
# include <stdio.h>

int main(void)
{
    char ch1, ch2, ch3;

    scanf("%c%c%c", &ch1, &ch2, &ch3);
    printf("%c%c%c%c%c", ch1, '#', ch2, '#', ch3);

    return 0;
}
```

AbC

A#b#C

A bC

A# #b

输出字符型数据

```
/* 字符b、B的ASCII码分别是98、66 */
# include <stdio.h>
int main(void)
{
    char ch = 'b';

    printf("%c, %d\n", 'b', 'b');
    printf("%c, %d\n", 98, 98);
    printf("%c, %d\n", 97, 'b'-1);
    printf("%c, %d\n", ch - 'a' + 'A', ch - 'a' + 'A');

    return 0;
}
```

```
b, 98
b, 98
a, 97
B, 66
```

字符运算

- 大小写英文字母转换

  'B' - 'b' = 'A' - 'a'

  ……

  'Z' - 'z' = 'A' - 'a'

大写字母 = 小写字母+ 'A' - 'a'

小写字母 = 大写字母+ 'a' - 'A'

- 数字字符和数字转换

  9 - 0 = '9' - '0'

  8 - 0 = '8' - '0 '

  ……

  1 - 0 = '1' - '0'

数字字符 = 数字 + '0'

数字 = 数字字符 - '0'

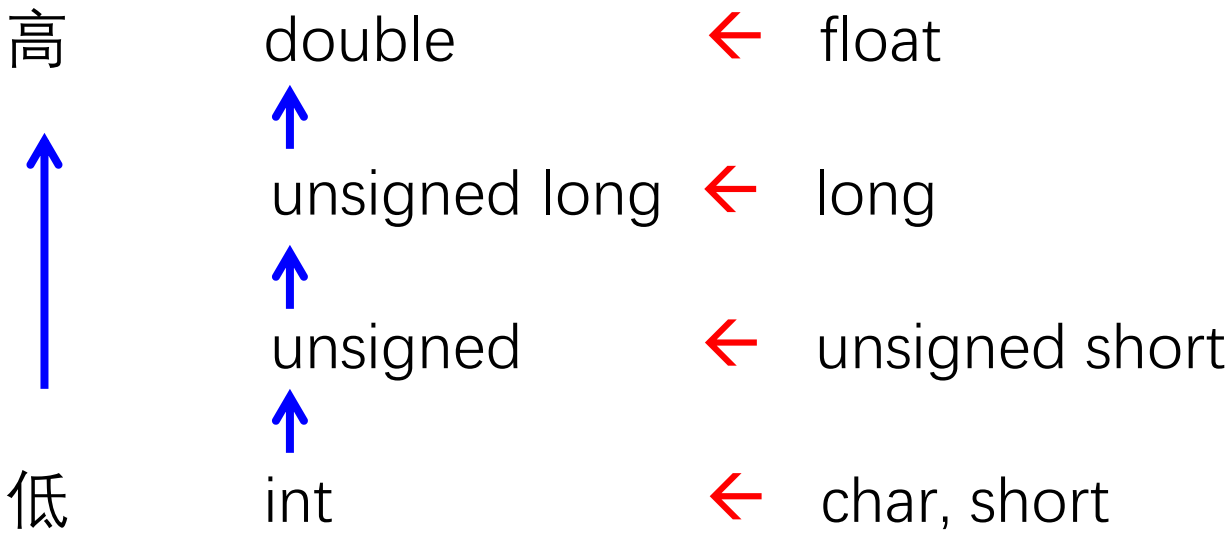
主要内容

- 数据的存储和基本数据类型
- 数据的输入和输出
- 类型转换
- 表达式

类型转换

- 不同类型数据的混合运算，先转换为同一类型，再运算。
- 自动类型转换
  - 非赋值运算的类型转换
  - 赋值运算的类型转换
- 强制类型转换

自动类型转换（非赋值运算）



- 水平方向：自动
- 垂直方向：低 → 高
- 短 → 长
- 带符号 → 无符号

自动类型转换（非赋值运算）

'A' + 12 – 10.05

65

77

66.95

自动类型转换（赋值运算）

变量 = 表达式

- 计算赋值运算符右侧表达式的值
- 将赋值运算符右侧表达式的值赋给左侧的变量

将赋值运算符右侧表达式的类型
自动转换成
赋值号左侧变量的类型

自动类型转换（赋值运算）

```
double x;
x = 1;
```

x = 1.0

```
short a = 1000;
char b = 'A';
long c;
c = a + b;
```

c = 1065

```
int ai;
ai = 2.56;
```

ai = 2

```
short bi;
bi = 0x12345678L
```

bi = 0x5678

例子：自动转换

long a = -1;

unsigned long b, c;

b = a;　　问 b = ?

c = a+1;　　问 c = ?

c = c + a;　　问 c = ?

b=2^n-|a|　n -- 变量b的位数

1是int，转换成long，a+1

先将a转换成unsigned long

强制类型转换

强制类型转换运算符

(类型名) 表达式

(double)3

(int)3.8

(double)(5/2)

(double)5/2

主要内容

- 数据的存储和基本数据类型
- 数据的输入和输出
- 类型转换
- 表达式