



中國人民大學
RENMIN UNIVERSITY OF CHINA

第3讲 变量运算与输入输出

余力

buaayuli@ruc.edu.cn

内容提要

3.1 C语言的基本数据类型

3.2 C语言的输入和输出

3.3 C语言的运算符与表达式

3.4 数据表示存与类型转换



中國人民大學
RENMIN UNIVERSITY OF CHINA



01. 基本数据类型

数据类型

- 维度1：存在不同的数据类型？
 - 数据可能是整数、实数、字母，等等
 - 思考：为什么C语言要区分不同数据类型？
 - 从程序员的角度.....
 - 从计算机的角度.....
 - C语言支持的数据类型
 - 数值型数据 (short, int, long, float, double)
 - 字符型数据 (char)
 - 用户自定义数据 (pointer, struct, union)

数据类型

■ C程序员可以使用的基本数据类型

	signed	unsigned
short	short int x; short y;	unsigned short x; unsigned short int y;
default	int x;	unsigned int x;
long	long x;	unsigned long x;
float	float x;	N/A
double	double x;	N/A
char	char x; signed char x;	unsigned char x;

- 有符号(signed)与无符号(unsigned)的区分
- 表示整数使用int, short, long三种类型
- 表示实数使用float, double两种类型
- 表示字符使用char类型

字符型数据

- 占用1个字节字符类型的数据
- 在内存中以相应的ASCII码存放。
- 'a'的ASCII码为97，内存中存储形式：

0	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

ASCII码表: <http://asciima.com/>

66	B	82	R	98	b	114	r	
67	C	83	S	99	c	115	s	
68	D	84	T	100	d	116	t	
69	E	85	U	101	e	117	u	
70	F	86	V	102	f	118	v	
71	G	87	W	103	g	119	w	
72	H	88	X	104	h	120	x	
73	I	89	Y	105	i	121	y	
74	J	90	Z	106	j	122	z	
75	K	91	[107	k	123	{	
76	L	92	\	108	l	124		
77	M	93]	109	m	125	}	
78	N	94	^	110	n	126	~	
79	O	95	_	111	o	127	␣	^Backspace 代码: DEL

变量与常量

- 维度2：程序执行过程数据是否变化？
 - 常量：程序执行过程中不发生变化
 - 变量：在程序中可以被赋值而发生变化
- 变量的命名规则
 - 包含字母、数字和‘_’
 - 只能以字母或下划线开头
 - 关键字不能作为变量名
 - 大小写敏感
 - 变量在一个函数范围内不能重名

变量 (1)

■ 小测试：指出下面变量命名的正确性

- `int money$owed;`
- `int total_count;`
- `int score2;`
- `int 2ndscore;`
- `int long;`
- `int x, X;` 定义了几个变量?

变量 (2)

变量代表内存中具有特定属性的一个存储单元，用于存储数据，也就是**变量的值**。

变量a



————— 变量值

内存单元地址XXXX

变量的定义和内存地址的关系

变量 (3)

■ 变量的声明

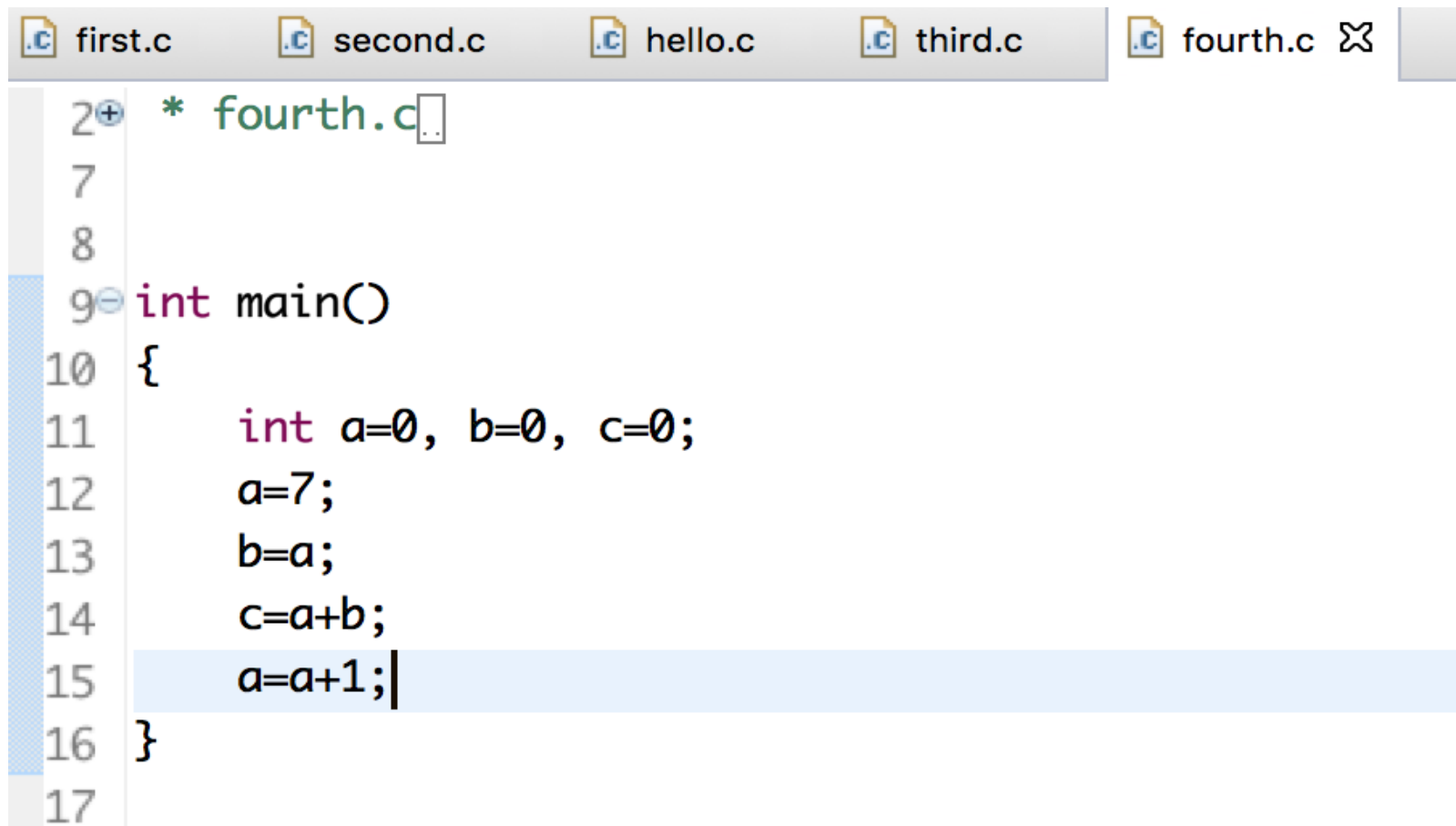
- 语句 `int a;` - 系统实际做了什么?
- 语句 `int a = 30;` - 系统实际做了什么?

■ 变量的赋值

- 变量必须先声明才能赋值 (为什么?)
- 语句 `a = 50;` - 系统实际做了什么?

变量 (4)

- 下面的代码中，变量是怎样赋值的？



```
first.c  second.c  hello.c  third.c  fourth.c ✕  
2+ * fourth.c  
7  
8  
9- int main()  
10 {  
11     int a=0, b=0, c=0;  
12     a=7;  
13     b=a;  
14     c=a+b;  
15     a=a+1;  
16 }  
17
```

常量 (1)

- 数字常量
 - 整型、实型
- 字符常量
 - 'a'、'b'、'2'
- 字符串常量
 - "Hello World"

常量的书写格式

■ 整型常量

- 在C语言中，整型常量可以用不同进制：
- 八进制：以0开头的数字序列（注意是数字 0，不是字母 o）
- 十六进制表示：以0x开头的数字序列
- 十进制：其它数字序列。

	short	int	long
八进制	%ho	%o	%lo
十进制	%hd	%d	%ld
十六进制	%hx 或者 %hX	%x 或者 %X	%lx 或者 %lX

常量的书写格式

- 长整型数据要在数字后加字母L或l。例:

-12L (十进制)

774545L (十进制)

076L (八进制, 32768)

0100000L (八进制, 62)

0X12l (十六进制, 18)

0x8000l (十六进制, 32768)

- **注意:** 12L与12值相等, 但占用的存储空间不同。

进制输入输出

```
01. #include <stdio.h>
02. int main()
03. {
04.     short a = 0b1010110; //二进制数字
05.     int b = 02713; //八进制数字
06.     long c = 0X1DAB83; //十六进制数字
07.
08.     printf("a=%ho, b=%o, c=%lo\n", a, b, c); //以八进制形式输出
09.     printf("a=%hd, b=%d, c=%ld\n", a, b, c); //以十进制形式输出
10.     printf("a=%hx, b=%x, c=%lx\n", a, b, c); //以十六进制形式输出（字母小写）
11.     printf("a=%hX, b=%X, c=%lX\n", a, b, c); //以十六进制形式输出（字母大写）
12.
13.     return 0;
14. }
```

实型常量

- 实型常量只能用十进制表示，不能用八进制或十六进制表示。
- 实型常量可以用小数或指数表示，例：

34.5 3.14 .345 345. 1e2 1.5e-3

- 字母e(或E)之前必须有数字，且e后指数必须为整数
- 规范化的指数形式：
 - 在字母e（或E）之前的小数部分中，
 - 小数点左边应有一位（且只能有一位）非零的数字。
 - 例如：123.456可以表示为：123.456e0, 12.3456e1, **1.23456e2**, 0.123456e3, 0.0123456e4, 0.00123456e

示例： π 值的几种表示形式

■ 日常表示法

- 3.14159×10^0
- 0.31159×10^1
- 0.0314159×10^2
- 31.4159×10^{-1}
- 314159×10^{-3}

■ C语言的表示形式

- $3.14159e0$
- $0.314159e+1$
- $0.0314159e+2$
- $31.4159e-1$
- $3141.59e-3$

字符常量

- 字符型常量是用单撇号括起来的一个字符,如:
- 'a' 'A' '?' '#' '8' " ' '
- “'”称为定界符, 不是字符常量的一部分。
- 不能用 “ ” 代替 “ ' ”, “ ” 在一些C语言中为字符串的定界符。
- 字符常量中不能直接使用 “'”、 “\”作为字符, 必须使用转义字符 “\'”和 “\\”。

转义字符

字符形式	功能
\n	换行
\t	横向跳格
\v	竖向跳格
\b	退格
\r	回车
\f	走纸换页
\\	反斜杠字符 “ \ ”
\'	单引号（撇号）字符
\ddd	1 到 3 位 8 进制 ASCII 码
\xhh	1 到 2 位 16 进制 ASCII 码

字符串常量

- 字符串常量是指在C语言中用一对双撇号括起来的零个或多个字符序列。如：
"Hello" "Programming in C" "A" ""
- 字符串以双撇号 (" ") 为定界符，但双撇号本身不属于字符串。
- 字符串中字符个数称为该字符串的长度。
- 字符串常数存储在机器中时，系统自动在字符串的末尾加一个“字符串结束标志”，它的转义字符为“\0”。

例：字符串的存储

- 字符串：

I say:'Goodby!'

可写成：

"I say:\'Goodby!\'"

- 字符串 "hello" 存储为：

h	e	l	l	o	\0
---	---	---	---	---	----

- 实际上每个字符都以ASCII码存储

104	101	108	108	111	0
-----	-----	-----	-----	-----	---

符号常量的使用

//已知单价和数量，求总价

```
#include <stdio.h>
```

```
#define PRICE 30
```

```
int main( )
```

```
{
```

```
    int num, total;
```

```
    num = 10;
```

```
    total = num * PRICE;
```

```
    printf( "total = %d\n", total);
```

```
    return 0;
```

```
}
```

- #define是宏定义命令
- 在本程序中，用PRICE来替代30这个数字
- 宏定义不是语句，结尾没有;

小测验

- 挑出下面程序中的问题

```
# include <stdio.h>
int main() {
    float g; h;
    float tax, rate;
    g = e21;
    tax = rate * g;
    printf("%d", tax);
}
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



02. 格式输入输出

输入输出函数scanf & printf

- 输入输出函数的作用是什么?
 - 与程序交流 (Communicate with a Program)
- 函数printf和scanf是否属于C语言本身?
 - 不! 它们不是C语言的关键字, 只是库函数的名字 (C library)
- 为什么C语言本身不提供输入输出?
 - C语言将输入输出的实现留给编译程序, 以更好适应指定机器的输入输出情况
- 你怎么来设计printf和scanf?

设计 printf 的基本思想 (1)

- 需要考虑哪些基本问题
 - What - 要输出的是什么?
 - How - 要怎样进行输出?



设计 printf 的基本思想 (2)

■ 函数 printf 需要解决哪些实际问题？

- 输入：一个或多个变量
 - 以二进制编码
- 输出：人可以识别的数据
 - 不同类型的变量的输出格式不同
 - 希望将变量嵌入到希望输出的“文字”中
 - 希望输出变量的不同精度
 - 希望对数据输出做些简单的“排版”

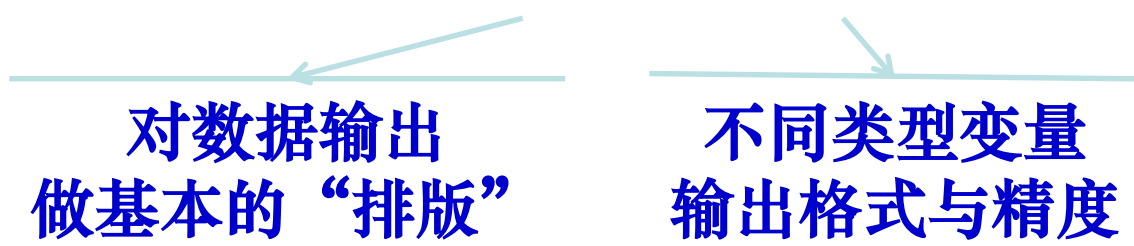
输出函数 printf 的使用 (1)

- “格式控制” 的使用
- 示例: “You look great in %s\n”
 - 双引号
 - 普通字符, 如You look great in和\n
 - 格式声明, 如%s
- 思考: “格式声明” 应该怎么用?

输出函数 printf 的使用 (2)

■ “格式声明” 的使用

➤ 格式声明 = % + 附加字符 + 格式字符



对数据输出
做基本的“排版”

不同类型变量
输出格式与精度

➤ 常用格式字符（详见教材）

- 输出整数：%4d (%i); %u, %o, %x, %X
- 输出实数：%f, %-n.mf, %e, %E
- 输出单个字符：%c
- 输出字符串：%s

输出函数 printf 的使用 (3)

■ “格式声明” 的使用

➤ 格式声明 = % + 附加字符 + 格式字符

对数据输出
做基本的“排版”

不同类型变量
输出格式与精度

➤ 附加字符

- 输出数据的最小宽度：一个整数n
- 靠左对齐与靠右对齐：是否加符号-

输出函数 printf 的使用 (4)

- 指出下面语句的错误:

```
printf("The score was Squids %d, Slugs %d.\n",  
score1);
```

- 注意:

- 格式声明的个数与变量的个数保持一致
- 使用准确的格式声明

小测验

- 声明一个整型变量age，将变量赋值为你实际的年龄，然后使用三次printf()函数
 - 使用一个printf()函数，将你的姓名和age，打印在一行
 - 使用一个printf()函数，将你的姓名和age，打印在两行
 - 使用两个printf()函数，将你的姓名和age，打印在一行

设计 scanf 的基本思想

- 需要考虑哪些基本问题
 - What - 要输入的是什么?
 - How - 要怎样进行输入?
- 参考 printf 的例子, scanf也应包含
 - 格式控制, 如" %d %f"
 - 变量列表, 如&num, &score
- 字符串怎么输入?
 - 示例: scanf ("%s" , name)

小测验

■ 挑出下面程序的错误!

```
define B booboo
define X 10
main(int)
{
    int age;
    char name;
    printf("Please enter your first name.");
    scanf("%s", name);
    printf("All right, %c, what's your age?\n", name);
    scanf("%f", age);
    xp = age + X;
    printf("That's a %s! You must be at least %d.\n", B, xp);
    rerun 0;
}
```

getchar与gets

- gets读一行，getchar()读一个字符

```
1  # include <stdio.h>
2
3  int main() {
4      char ch1, ch2;
5      ch1 = getchar();
6      ch2 = getchar();
7      printf("%d %d\n", ch1, ch2);
8      return 0;
9  }
```

```
1  # include <stdio.h>
2  int main()
3  {
4      char str1[20], str2[20];
5      gets(str1);
6      printf("%s\n", str1);
7      gets(str2);
8      printf("%s\n", str2);
9      return 0;
10 }
```

C++的输入输出流

```
#include<iostream>
using namespace std;
```

```
int main () {
    int n1,n2;
    float f1, f2;
    double d1, d2;
    char c1, c2;

    cin >> n1 >> n2;
    cin >> f1 >> f2;
    cin >> d1 >> d2;
    cin >> c1 >> c2;

    cout << n1 << " " << n2 << endl;
    cout << f1 << " " << f2 << endl;
    cout << d1 << " " << d2 << endl;
    cout << c1 << " " << c2 << endl;
    return 0;
}
```

What is new?

- 建立C++项目与.cpp为扩展名的源文件
 - C++兼容C语言（语句、头文件.....）
- 预处理：引入输入输出流的头文件

```
#include<iostream>  
using namespace std;
```

- 输入流cin
 - 从键盘提取变量信息（>>）
- 输出流cout
 - 将变量插入到显示设备中（<<）

输出流的基本操作

■ cout语句的一般格式为

➤ `cout<<表达式1<<表达式2<<.....<<表达式n;`

■ 一个cout语句可以分成若干行写

```
cout << n1 << " " << n2 << endl << f1 << " " << f2 << endl;
```



等价

```
cout << n1 << " " << n2 << endl;
```

```
cout << f1 << " " << f2 << endl;
```

```
1  cout<<"This is " //注意行末尾无分号
2  <<"a C++ "
3  <<"program."
4  <<endl; //语句最后有分号
```

等价

```
1  cout<<"This is "; //语句末尾有分号
2  cout <<"a C++ ";
3  cout <<"program.";
4  cout<<endl;
```

输出流的基本操作

- 在用cout输出时，用户不必通知计算机按何种类型输出，系统会自动判别输出数据的类型，使输出的数据按相应的类型输出。
 - 符号endl表示换行符
 - 为什么不用\n了？

1	cout<<a,b,c; //错误,不能一次插入多项
2	cout<<a+b+c; //正确,这是一个表达式,作为一项

输入流的基本格式

- cin语句的一般格式为

- cin>>变量1>>变量2>>.....>>变量n;

- 一个cin语句可以分成若干行写

```
cin >> n1 >> n2 >> f1 >> f2  
>> d1 >> d2 >> c1 >> c2;
```

等价



```
cin >> n1 >> n2;  
cin >> f1 >> f2;  
cin >> d1 >> d2;  
cin >> c1 >> c2;
```

- 在用cin输入时,系统也会根据变量的类型从输入流中提取相应长度的字节

- **不能**用cin语句把**空格字符**和**回车换行符**作为字符输给字符变量, 它们被**跳过**

用输入流接收字符串

```
1  #include <iostream>
2  using namespace std;
3  main ()
4  {
5      char a[20]; //此处也可以是char* a;
6      cin>>a;
7      cout<<a<<endl;
8  }
```

- 接收字符串时，遇“空格”、“TAB”、“回车”都结束。

如何接收一行字符串

- 一行字符串可能包含空白

```
1  # include <iostream>
2  using namespace std;
3
4  □ int main() {
5      |     char m[20];
6      |     cin.getline(m, 3);
7      |     cout << m << endl;
8      | }

```

- **cin.getline** (接受字符串的数组m, 接受个数5, 结束字符)
- cin 读取的字符数将比该数字少一个, 为 null 终止符留出空间
 - 当第三个参数省略时, 系统默认为 ' \0'

接收单个字符

```
1 # include <iostream>
2 using namespace std;
3
4 int main() {
5     char ch;
6     cin.get(ch);
7     // ch= cin.get();
8     cout << ch << endl;
9 }
```

- **cin.get(字符变量名)**
 - 可以用来接收字符，可以接收空格、TAB和回车。

cout保留小数点

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    float a;
    cin >> a;
    cout << fixed << setprecision(5) << a;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    float a;
    cin >> a;
    cout << fixed << setprecision(5) << a;
}
```

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    double PI=3.141592654;
```

```
    cout<<PI<<endl;
```

```
    cout<<setprecision(2)<<PI<<endl;
```

```
    cout<<fixed<<setprecision(2)<<PI<<endl;
```

```
    cout<<setfill('*')<<setw(20)<<setprecision(10)<<PI<<endl;
```

```
    cout<<setfill('*')<<setw(20)<<setprecision(10)<<left<<PI<<endl;
```

```
    cout<<scientific<<setprecision(10)<<PI<<endl;
```

```
    cout<<scientific<<uppercase<<setprecision(10)<<PI<<endl;
```

```
    return 0 ;
```

```
}
```

输出结果如下：

3.141592654

3.1

3.14

*****3.1415926540

3.1415926540*****

3.1415926540e+000

3.1415926540E+000



中國人民大學
RENMIN UNIVERSITY OF CHINA



03. 运算符与表达式

算术运算符

- 基本的算术运算符

- 正(负)号: $+$ ($-$)

- 加: $+$

- 减: $-$

- 乘: $*$

- 除: $/$

- 求余: $\%$

- 例子: 如求21整除4的余数, 可用下式表示: $21\%4$

自增、自减运算符

- $++i, --i$
 - 在使用之前, 先使 i 的值加(减)1
- $i++, i--$
 - 使用之后, 使 i 的值加(减)1
- 要先彻底理解, 再进行使用

运算符

- 关系运算符

- $>, \geq, <, \leq, ==, !=$

- 逻辑运算符

- $!, \&\&, ||$

- 赋值运算符

- $=$

运算符的优先次序（参考）

- | | |
|--|--------------------------|
| 1. ., →, [], () | 8. & |
| 2. ++, --, !, ~,
(DataType), sizeof | 9. ^ |
| 3. *, /, % | 10. |
| 4. +, - | 11. && |
| 5. >>, << | 12. |
| 6. >, <, >=, <= | 13. ?: |
| 7. ==, != | 14. =, +=, -=, *=, |

不用死记，复杂表达式中多用“()”运算符

表达式

- **表达式** (expression) 是由运算符 (操作符, operator) 和运算对象 (操作数, operand) 组成。
 - 表达式的计算结果称为表达式的值
 - 一个常量或一个变量名字是最简单的表达式, 其值即该常量或变量的值
 - 运算符不能单独使用, 必须有操作数与之共同构成表达式
 - 表达式的值还可以用作其他运算的操作数

表达式的类型

- C语言的表达式有类型区别，常见的：
 - 算术表达式
 - 用算术运算符和括号将操作数连接起来
 - 关系表达式
 - 逻辑表达式
 - 赋值表达式
- 表达式类型由操作符和运算结果决定。

整数算术表达式

- 如果在算术运算中只包含整数，任何结果的小数部分均被截去，即使结果赋给浮点型变量也是这样。
- 如果希望保留计算的小数部分，就必须强制编译程序把操作对象之一变成float型，方法可以是：
 - 操作对象直接定义为float型
 - 操作对象前加强制float类型转换，如 $2 \rightarrow (\text{float})2$
 - 操作对象表示为实数，如 $2 \rightarrow 2.0$

关系表达式

- 用**关系运算符**将两个表达式连接起来的式子，称关系表达式
- 关系表达式的值是一个逻辑值，即“真”或“假”

运算符	说明
>	大于
>=	大于等于
<	小于
<=	小于等于
=	等于
!=	不等于

} 优先级相同（高）

} 优先级相同（低）

逻辑运算

- 逻辑运算符如下：

运算符	说明
!	逻辑非 (NOT)
&&	逻辑与 (AND)
	逻辑或 (OR)

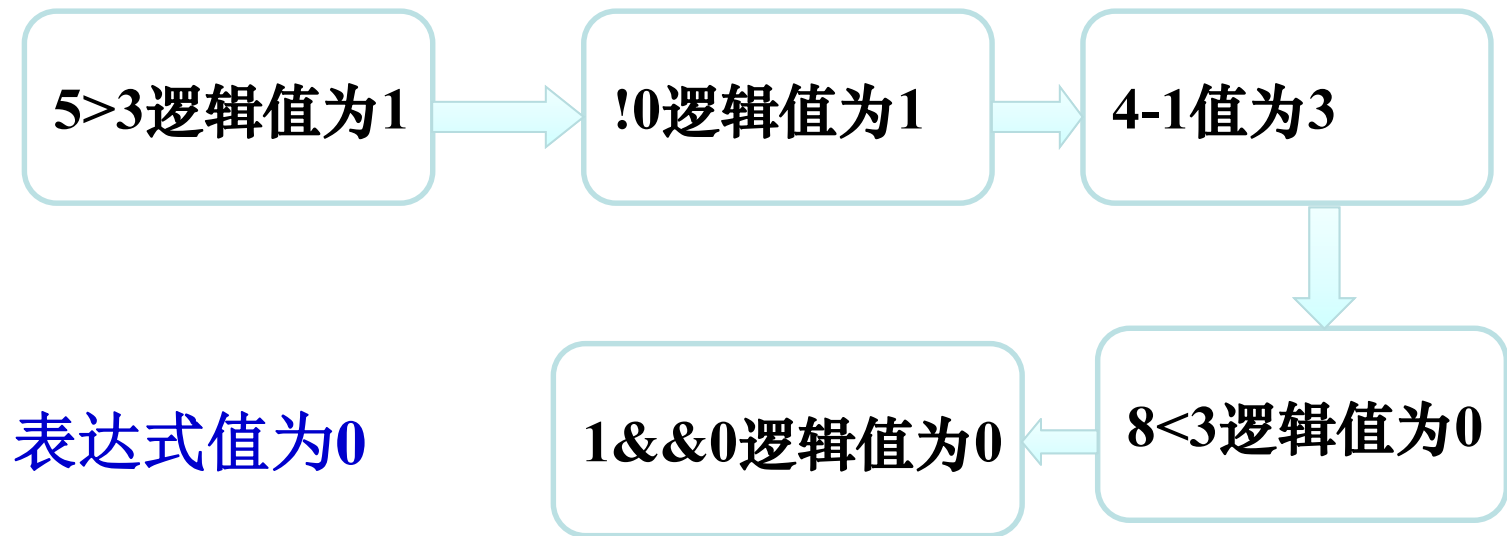
- !为单目运算符，&&和||为双目运算符，结合方向为从右至左。

逻辑运算运算的优先级

- $! > \&\& > ||$, 即 “!”为三者中最高。
- “&&”和 “||”**低于**关系运算符。
- “!”**高于**算术运算符。

逻辑表达式 (1)

- 用逻辑运算符将关系表达式或逻辑量连接起来的式子
- 求解： $5 > 3 \&\& 8 < 4 - !0$ 的值



逻辑表达式 (2)

- 在逻辑表达式的求解中，并不是所有逻辑运算符都要被执行。
 - $a \& \& b \& \& c$ 只有a为真时，才需要判断b的值，只有a和b都为真时，才需要判断c的值。
 - $a || b || c$ 只要a为真，就不必判断b和c的值，只有a为假，才判断b。a和b都为假才判断c
 - 例如： $(m = a > b) \& \& (n = c > d)$
 - $a=1, b=2, c=3, d=4, m$ 和 n 的原值为1时，由于“ $a > b$ ”的值为0，因此 $m=0$ ，而“ $n = c > d$ ”不被执行，因此 n 的值不是0而仍保持原值1。

赋值表达式

- $\langle \text{变量} \rangle \langle \text{赋值运算符} \rangle \langle \text{表达式} \rangle$
- 先求赋值表达式右侧的表达式的值，然后赋值给运算符左侧的变量
- 赋值表达式的值是赋值号左边变量被赋值后的值



中國人民大學
RENMIN UNIVERSITY OF CHINA



04. 数据存储与类型转换

数据类型

■ C程序员可以使用的基本数据类型

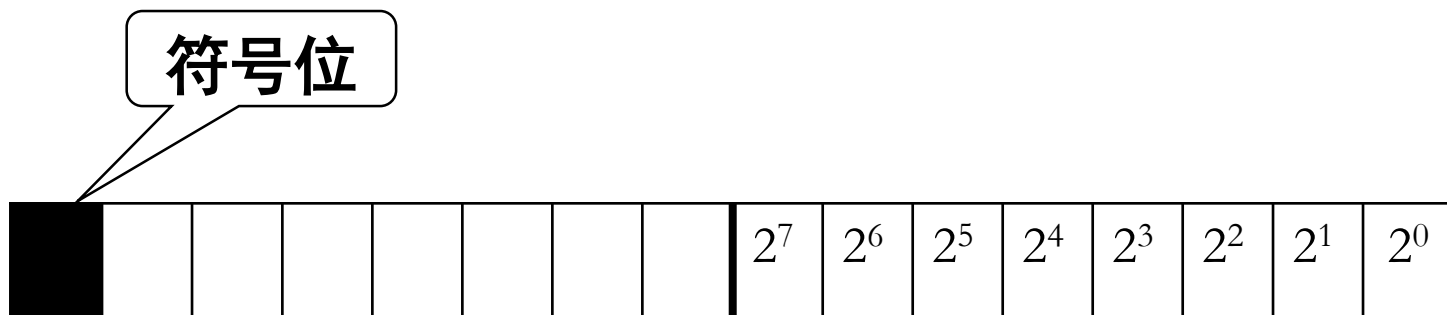
	signed	unsigned
short	short int x; short y;	unsigned short x; unsigned short int y;
default	int x;	unsigned int x;
long	long x;	unsigned long x;
float	float x;	N/A
double	double x;	N/A
char	char x; signed char x;	unsigned char x;

- 有符号(signed)与无符号(unsigned)的区分
- 表示整数使用int, short, long三种类型
- 表示实数使用float, double两种类型
- 表示字符使用char类型

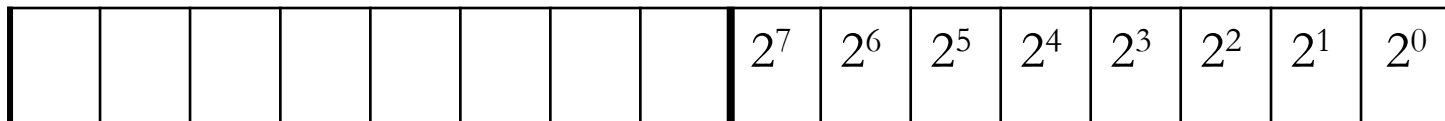
整数类型 (1)

- 在计算机中如何表示和存储整数?

16位有符号整数



16位无符号整数

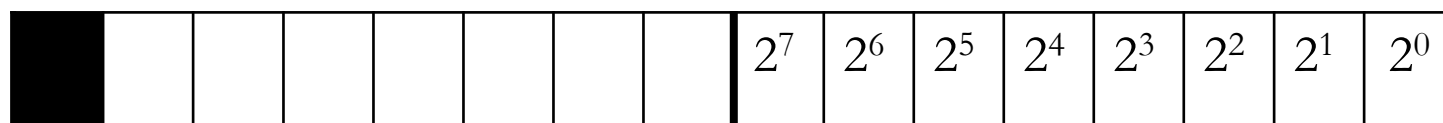


整数类型 (2)

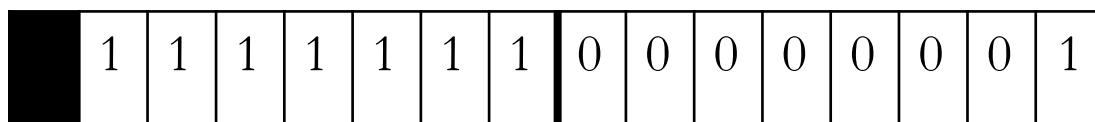
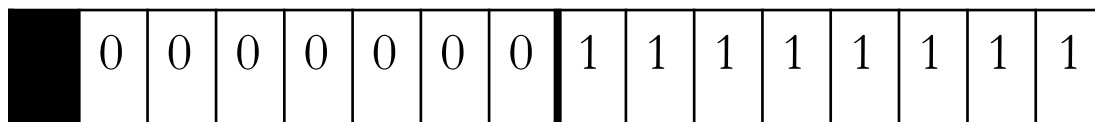
- 有符号(Signed)整数的表示方法

符号位

例：16位有符号整数



➤ 存储负数时，使用补码



补码
Complement

整数类型 (3)

- 三种整数类型short, int, long的区别是用多少个字节存储
(不同的存储空间)
- 每种类型的空间大小与机器相关, 但大部分PC机如下

类型	符号	字节	位数	最小值	最大值
short int	Signed unsigned	2	16	-32768 0	32767 65535
int	Signed unsigned	4	32	-2,147,483,648 0	2,147,483,647 4,294,967,295
long int	Signed unsigned	4	32	-2,147,483,648 0	2,147,483,647 4,294,967,295

整数类型 (4): sizeof函数

```
1  #include <stdio.h>
2
3  int main() {
4      printf( "char: %d bytes \n", sizeof(char) );
5      printf( "short: %d bytes \n", sizeof(short) );
6      printf( "int: %d bytes \n", sizeof(int) );
7      printf( "long: %d bytes \n", sizeof(long) );
8      printf( "float: %d bytes \n", sizeof(float) );
9      printf( "double: %d bytes \n", sizeof(double) );
10     return 0;
11 }
```

char: 1 bytes
short: 2 bytes
int: 4 bytes
long: 4 bytes
float: 4 bytes
double: 8 bytes

实型数据

■ 在计算机中如何表示和存储实数

➤ IEEE 754 标准



- float 浮点型用4字节，表示范围： $-3.4 * 10^{38} - 3.4 * 10^{38}$
- double 双精度8字节，表示范围： $-1.7 * 10^{308} - 1.7 * 10^{308}$

扩展阅读: <http://liyanrui.is-programmer.com/posts/3806.html>

字符型数据

- 占用1个字节字符类型的数据
- 在内存中以相应的ASCII码存放。
- 例：'a'的ASCII码为97，内存中的存储

0	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

66	B	82	R	98	b	114	r	
67	C	83	S	99	c	115	s	
68	D	84	T	100	d	116	t	
69	E	85	U	101	e	117	u	
70	F	86	V	102	f	118	v	
71	G	87	W	103	g	119	w	
72	H	88	X	104	h	120	x	
73	I	89	Y	105	i	121	y	
74	J	90	Z	106	j	122	z	
75	K	91	[107	k	123	{	
76	L	92	\	108	l	124		
77	M	93]	109	m	125	}	
78	N	94	^	110	n	126	~	
79	O	95	_	111	o	127	␣	^Backspace 代码：DEL

ASCII码表: <http://asciima.com/>

符号常量的使用

```
#include <stdio.h>
```

```
#define PRICE 30
```

```
int main( )
```

```
{
```

```
    int num, total;
```

```
    num = 10;
```

```
    total = num * PRICE;
```

```
    printf( "total = %d\n", total);
```

```
    return 0;
```

```
}
```

- #define是宏定义命令
- 在本程序中，用PRICE来替代30这个数字
- 宏定义不是语句，结尾**没有**;

截去小数与四舍五入

- 当一个实数（浮点数）转换为整数时，实数的小数部分全部舍去，并按整型存放。但实数的整数部分不要超过整数允许的最大范围，否则数据出错。例：

3276.85转换成整数为3276

- 当由double型转换为float型，去掉多余的有效数字，按四舍五入处理。

丢失精度

- 下例情况可能丢失精度：
 - 四舍五入会丢失一些精度；
 - 截去小数也会丢失一些精度；
 - 由long型转换为float或double型时，有可能在存储时不能准确地表示该长整数的有效数字，精度也会损失。

结果不确定与截去高位

- 浮点数降格时，即double转换为float或long，float转换为long、int或short型。数值超过目标类型的取值范围时，所得的结果将是不确定的。
- 当较长的整数转换为较短的整数时，要将高位截去只将低位字节送过去。这会产生很大的误差。

不同类型数据的隐式转换

- C语言中的数据类型转换分为两种：
 - 隐式转换
 - 显示转换
- 隐式转换发生在以下4种情况：
 - 运算转换，即在不同类型数据混合运算时出现；
 - 赋值类型
 - 输出转换
 - 函数调用转换
- 下面介绍前三种转换，函数调用转换将在以后讨论。

1、一般算术转换

- 算术运算一般指下列运算：
 - 加(+)、减(-)、乘(*)、除(/)、取余(%）、负号(-)运算
- 算术转换的目的：
 - 将短的数扩展成机器处理的长度
 - 使运算符两端具有共同的类型
- 算术转换原则如下：
 - 将表达式中的char或short全部自动转换为相应的int型；
将float转换为double型。

结果表达式中只剩下五种类型：

char } → int
short }

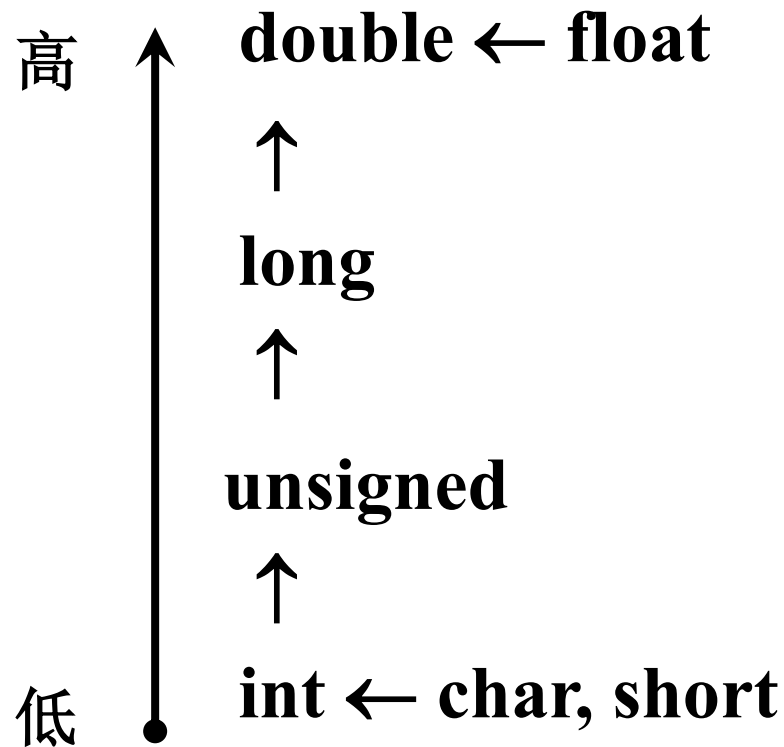
unsigned char } → unsigned int
unsigned short }

float → double

long

unsigned long

当一个运算符两端的运算量类型不一致时，按“向高看齐”的原则对“较低”的类型进行提升。



类型的高低

2、赋值转换

- C语言允许通过赋值使 “=”右边表达式的值的类型自动转换为其左边变量的类型。
- 赋值转换具有强制性，可能是提升，也可能是降格。

3、输出转换

- 在C语言中任何一种类型的数据，都可以转换为与原有类型不同的类型输出。例如：
 - long型数据在printf函数中指定用int型输出（使用%d转换格式）
 - int型数可按无符号方式输出（使用%u转换）。

不同类型数据的显示转换

- C语言提供一种“强制类型转换”运算符，将一种类型的变量强制转换为另一种类型。
- 一般形式为：

(类型标识符) 表达式

例：

`(char)(3 - 3.14159 * x)`

`k=(int)((int)x + (float)i + j)`

`(float)(x = 99)`

`(double)(5 % 3)`

注意：

- 显式转换实际上是一种单目运算符，各种数据类型的标识符都可以用来作显式转换运行符。
- 表达式应该用括号括起来。如果写成：

$(\text{int}) x + y$

则只将x转换成整型，然后与y相加。

- 对一个变量进行显式转换后，得到一个所需类型的中间变量，但原来变量的类型不变。
- C语言提供的数学函数中多数要求参数为 double型，在调用这些函数时可以用显式转换进行类型转换。

作业练习

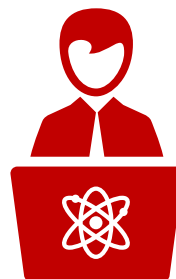
- 作业

- #4. 读入字符串并输出
- #5. 切换字符大小写
- #6. 浮点数求和

- Deadline: 2021-10-7



中國人民大學
RENMIN UNIVERSITY OF CHINA



谢谢大家!

