



南京大學
NANJING UNIVERSITY



C语言中的整数

南京大学

计算机科学与技术系

袁春风

email: cfyuan@nju.edu.cn



2015.6

C语言支持的基本数据类型

| C语言声明 | 操作数类型 | 存储长度（位） |
|--------------------------|---------|---------|
| (unsigned) char | 整数 / 字节 | 8 |
| (unsigned) short | 整数 / 字 | 16 |
| (unsigned) int | 整数 / 双字 | 32 |
| (unsigned) long int | 整数 / 双字 | 32 |
| (unsigned) long long int | - | 2×32 |
| char * | 整数 / 双字 | 32 |
| float | 单精度浮点数 | 32 |
| double | 双精度浮点数 | 64 |
| long double | 扩展精度浮点数 | 80 / 96 |

整数类型分：无符号整数和带符号整数

无符号整数 (Unsigned integer)

- 机器中字的位排列顺序有两种方式：（例：32位字: $0\dots01011_2$ ）
 - 高到低位从左到右：**0000 0000 0000 0000 0000 0000 0000 1011**  **LSB**
 - 高到低位从右到左：**1101 0000 0000 0000 0000 0000 0000 0000**  **MSB**
 - Leftmost 和 rightmost 这两个词有歧义，故用 **LSB(Least Significant Bit)**来表示最低有效位，用**MSB**来表示最高有效位
 - 高位到低位多采用从左往右排列
- 一般在全部是正数运算且不出现负值结果的情况下，可使用无符号数表示。例如，地址运算，编号表示，等等
- 无符号整数的编码中**没有符号位**
- 能表示的最大值大于位数相同的带符号整数的最大值（Why？）
 - 例如，8位无符号整数最大是255（1111 1111）
而8位带符号整数最大为127（0111 1111）
- 总是整数，所以很多时候就**简称为“无符号数”**

带符号整数 (Signed integer)



- 计算机必须能处理正数(positive) 和负数(negative) , 用MSB表示数符 (0--正数 , 1--负数)
- 有三种定点编码方式
 - Signed and magnitude (原码)
定点小数 , 用来表示浮点数的尾数
 - Excess (biased) notion (移码)
定点整数 , 用于表示浮点数的阶 (指数)
 - Two's complement (补码)
50年代以来 , 所有计算机都用补码来表示带符号整数
- 为什么用补码表示带符号整数 ?
 - 补码运算系统是模运算系统 , 加、减运算统一
 - 数0的表示唯一 , 方便使用
 - 比原码多表示一个最小负数

C语言程序中的整数

无符号数：unsigned int (short / long) ; 带符号整数：int (short / long)

常在一个数的后面加一个“u”或“U”表示无符号数

若同时有无符号和带符号整数，则C编译器将带符号整数强制转换为无符号数

假定以下关系表达式在32位用补码表示的机器上执行，结果是什么？

| 关系表达式 | 运算类型 | 结果 | 说明 |
|--------------------------------|------|----|----|
| 0 == 0U | | | |
| -1 < 0 | | | |
| -1 < 0U | | | |
| 2147483647 > -2147483647-1 | | | |
| 2147483647U > -2147483647-1 | | | |
| 2147483647 > (int) 2147483648U | | | |
| -1 > -2 | | | |
| (unsigned) -1 > -2 | | | |

C语言程序中的整数

| 关系 表达式 | 类型 | 结果 | 说明 |
|----------------------------------|----|----|--|
| $0 == 0U$ | 无 | 1 | $00...0B = 00...0B$ |
| $-1 < 0$ | 带 | 1 | $11...1B (-1) < 00...0B (0)$ |
| $-1 < 0U$ | 无 | 0* | $11...1B (2^{32}-1) > 00...0B(0)$ |
| $2147483647 > -2147483647 - 1$ | 带 | 1 | $011...1B (2^{31}-1) > 100...0B (-2^{31})$ |
| $2147483647U > -2147483647 - 1$ | 无 | 0* | $011...1B (2^{31}-1) < 100...0B(2^{31})$ |
| $2147483647 > (int) 2147483648U$ | 带 | 1* | $011...1B (2^{31}-1) > 100...0B (-2^{31})$ |
| $-1 > -2$ | 带 | 1 | $11...1B (-1) > 11...10B (-2)$ |
| $(unsigned) -1 > -2$ | 无 | 1 | $11...1B (2^{32}-1) > 11...10B (2^{32}-2)$ |

带*的结果与常规预想的相反！

C语言程序中的整数

例如，考虑以下C代码：

```
1 int x = -1;
2 unsigned u = 2147483648;
3
4 printf ( "x = %u = %d\n" , x, x);
5 printf ( "u = %u = %d\n" , u, u);
```

在32位机器上运行上述代码时，它的输出结果是什么？为什么？


$x = 4294967295 = -1$

$u = 2147483648 = -2147483648$

- ◆ 因为-1的补码整数表示为“11...1”，作为32位无符号数解释时，其值为 $2^{32}-1 = 4\ 294\ 967\ 296-1 = 4\ 294\ 967\ 295$ 。
- ◆ 2^{31} 的无符号数表示为“100...0”，被解释为32位带符号整数时，其值为最小负数： $-2^{32-1} = -2^{31} = -2\ 147\ 483\ 648$ 。

编译器处理常量时默认的类型


- C90



| 范围 | 类型 |
|------------------------|--------------------|
| $0 \sim 2^{31}-1$ | int |
| $2^{31} \sim 2^{32}-1$ | unsigned int |
| $2^{32} \sim 2^{63}-1$ | long long |
| $2^{63} \sim 2^{64}-1$ | unsigned long long |

$2147483648 = 2^{31}$

- C99



| 范围 | 类型 |
|------------------------|--------------------|
| $0 \sim 2^{31}-1$ | int |
| $2^{31} \sim 2^{63}-1$ | long long |
| $2^{63} \sim 2^{64}-1$ | unsigned long long |

C语言程序中的整数

- 1) 在有些32位系统上，C表达式 $-2147483648 < 2147483647$ 的执行结果为false。Why？
- 2) 若定义变量“`int i=-2147483648;`”，则“`i < 2147483647`”的执行结果为true。Why？
- 3) 如果将表达式写成“ $-2147483647-1 < 2147483647$ ”，则结果会怎样呢？Why？

1) 在ISO C90标准下，2147483648为unsigned int型，因此

“ $-2147483648 < 2147483647$ ”按无符号数比较，
10.....0B比01.....1B大，结果为false。

在ISO C99标准下，2147483648为long long型，因此

“ $-2147483648 < 2147483647$ ”按带符号整数比较，
10.....0B比01.....1B小，结果为true。

由C语言中的
“Integer
Promotion”
规则决定的。

2) $i < 2147483647$ 按int型数比较，结果为true。

3) $-2147483647-1 < 2147483647$ 按int型比较，结果为true。

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

```
void main()
```

```
{
```

```
    int x=-1;
```

```
    unsigned u=2147483648;
```

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

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

```
    if(-2147483648 < 2147483647)
```

```
        printf("-2147483648 < 2147483647 is true\n");
```

```
    else
```

```
        printf("-2147483648 < 2147483647 is false\n");
```

```
    if(-2147483648-1 < 2147483647)
```

```
        printf("-2147483648-1 < 2147483647\n");
```

```
    else if(-2147483648-1 == 2147483647)
```

```
        printf("-2147483648-1 == 2147483647\n");
```

```
    else
```

```
        printf("-2147483648-1 > 2147483647\n");
```

```
}
```

请大家试试在C99上的运行结果。

C90上的运行结果是什么？

```
x = 4294967295 = -1
```

```
u = 2147483648 = -2147483648
```

```
-2147483648 < 2147483647 is false
```

```
-2147483648-1 == 2147483647
```