清華大學
Tsinghua University

程序设计基础

Fundamental of Programming

清华大学软件学院

刘玉身

liuyushen@tsinghua.edu.cn

Lecture 2: Outline

- **Variables, data types, and arithmetic operators**
- **变量，数据类型，算术运算符**

Variables, Types and Operators

- **Review**
- **Variables and data types**
- **Arithmetic Operators**

Variables, Types and Operators

- **Review**
- **Variables and data types**
- **Arithmetic Operators**

Review: Basics

- **Variable declarations:**
  - `int i; float f;`
- **Initialization:**
  - `char c='A'; int x=y=10;`
- **Arithmetic Operators:**
  - `+,-,*,/,%`
- **Expressions:**
  - `int x,y,z; x=y*2+z*3;`
- **Function:**
  - `int factorial (int n);` **/*function takes int , returns int */**

Variables, Types and Operators

- Review
- **Variables and data types**
- Arithmetic Operators

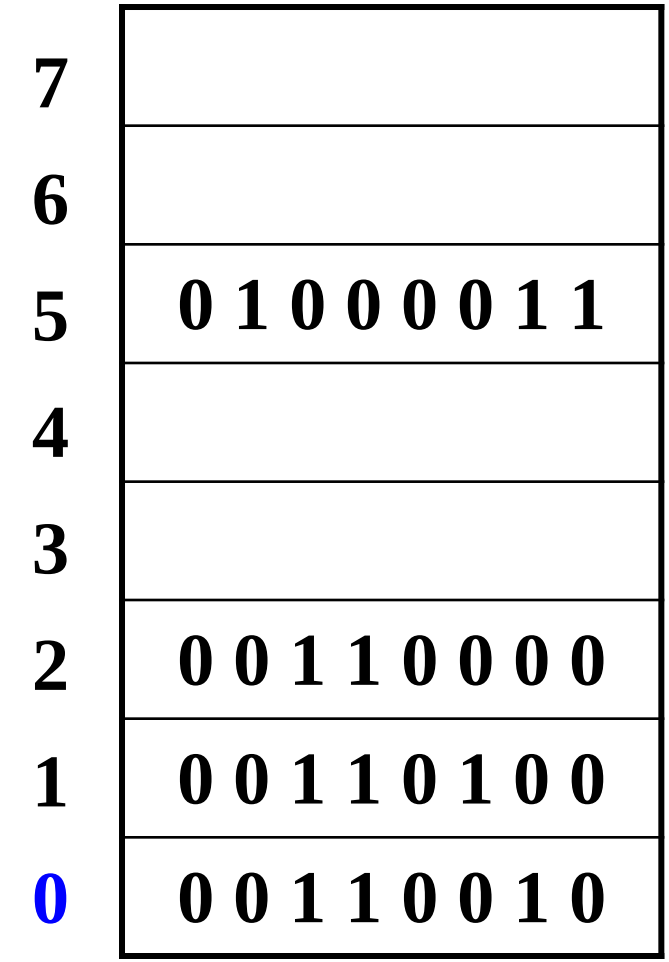
1. 什么是“变量”？
2. 为什么需要“变量”？
3. 如何访问一个“变量”？

内存的工作原理

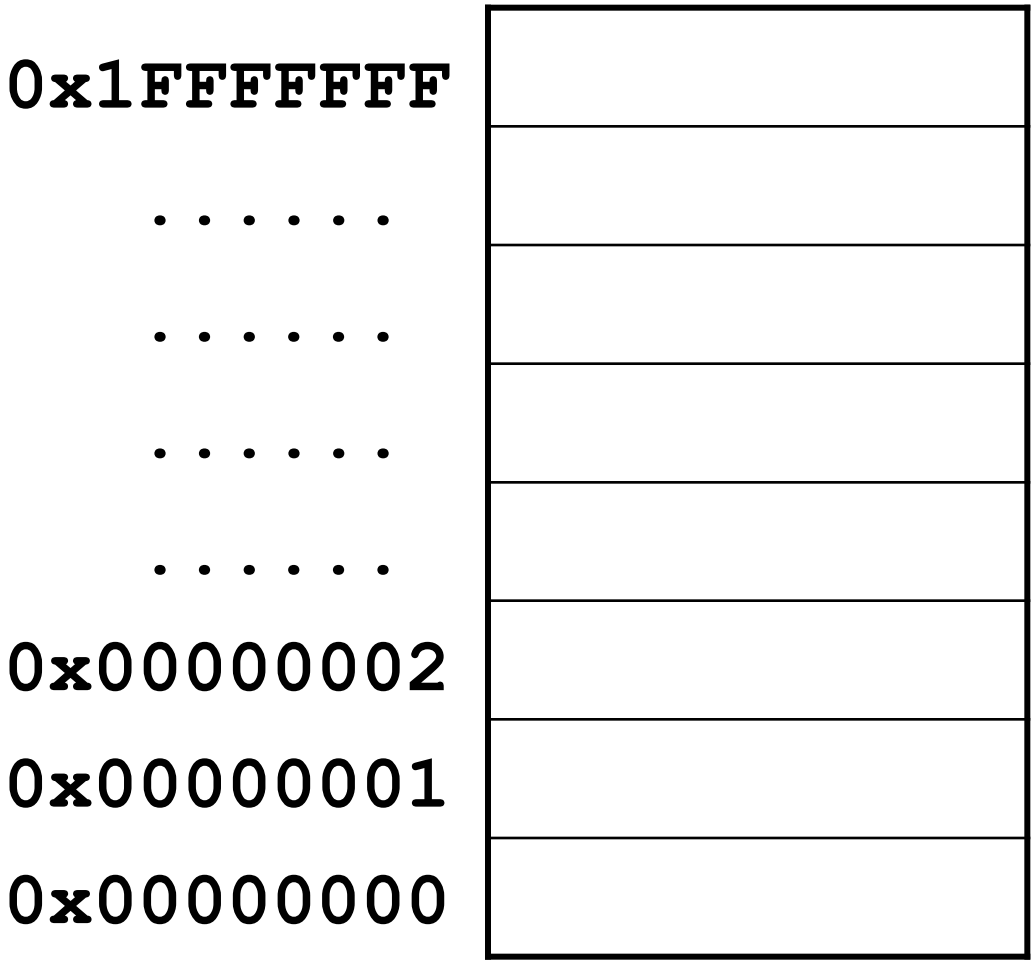
- **一个内存中包含有许多存储单元，每个单元可以存放一个适当单位的信息**
  - **bit (binary digit, 位/比特): 0 or 1**
  - **byte (字节) = 8 bits (1B)**

KB (kilobyte) = 1024 byte (2^{10} B)
MB (megabyte) = 1024 KB (2^{20} B)
GB (gigabyte) = 1024 MB (2^{30} B)
TB (terabyte) = 1024 GB (2^{40} B)
PB (petabyte) = 1024 TB (2^{50} B)

- **全部存储单元按一定顺序编号，这种编号称为存储器的地址。对各个存储单元的读写操作就是通过它们的地址来进行的**



Example: 512MByte



不同的数据，可能需要不同长度的存储空间，怎么办?

如：1、300、70000

Data types —— 数据类型

- 把所有的数据归纳为有限的几种类型；
- 同一种类型的数据具有相同的长度，占用相同大小的内存空间；
- 每一种类型的数据依然是以**二进制**的形式存放在内存当中；
- 在访问一个数据时，根据它在内存的**起始地址**和**类型**来确定它所占用的存储单元。

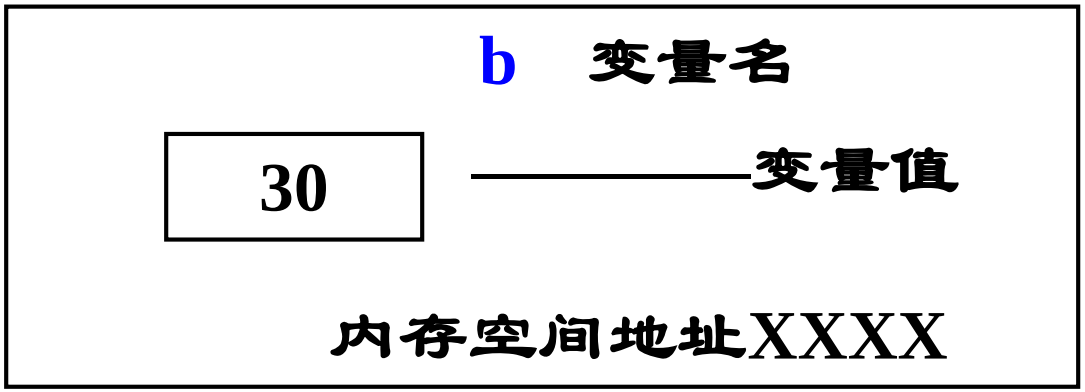
占用空间大于实际需要？

- C语言的四种基本类型：
  - 字符类型：用 **char** 来表示
  - 整数类型：用 **int** 来表示
  - 单精度浮点类型：用 **float** 来表示
  - 双精度浮点类型：用 **double** 来表示
- 此外，C语言还有一些类型修饰符：**short**、**long**、**signed**、**unsigned**

变量的基本概念

A variable is as named link/reference to a value stored in the system's memory or an expression that can be evaluated.

变量：其值可变的量



通过 b 可以找到相应的存储空间地址XXXX，

从而对该变量的值进行访问和修改

一道面试题： Pass by value vs. Pass by reference

- **传值 (pass by value)**
- **引用 (pass by reference)**

变量的命名规则

- 仅包含字母、数字和下划线（‘_’）；
- 第一个字符必须为字母或下划线；
- 不能使用C语言保留的“关键字”来作为变量名，如`int, float, if, else`等；
- 变量名是大小写有关(case sensitive)的，例如：sum和SUM是两个不同的变量名。

合法的变量名：

sum, average, _total, Class, Stu_name, LI

是否合法？

float&variable, Main, M.John, _int

12a, if, a>b, average_samples_div_count

张三

合法的名字 != 好名字

变量的定义

数据类型　变量1, 变量2, …, 变量n;

例如：

```
int nA, nB, nC;
double totalCourses, totalPoints, gpa;
```

整数类型

整数类型可分为：基本型、短整型和长整型三种。

1. 基本型：`int`（4字节）；
2. 短整型：`short int`，或`short`（2字节）；
3. 长整型：`long int`，或`long`（4字节）；

无符号整数类型：

`unsigned int`，`unsigned short`和`unsigned long`

整型数据的长度及取值范围 (通常在32位系统)

数据类型	字节数	比特数	取值范围
int	**4**	**32**	**$-2^{31} \sim (2^{31} - 1)$**
short	**2**	**16**	**$-2^{15} \sim (2^{15} - 1)$**
long	**4**	**32**	**$-2^{31} \sim (2^{31} - 1)$**
unsigned int	**4**	**32**	**$0 \sim (2^{32} - 1)$**
unsigned short	**2**	**16**	**$0 \sim (2^{16} - 1)$**
unsigned long	**4**	**32**	**$0 \sim (2^{32} - 1)$**

Write a program to print the sizes of data types in your machine.

Sizes of Data types

- C 标准并没有定义具体的整数类型的宽度

- The individual sizes are machine/compiler dependent. However, the following is guaranteed:
  - sizeof(char)<sizeof(short)<=sizeof(int)<=sizeof(long) and sizeof(char)<sizeof(short)<=sizeof(float)<=sizeof(double)

- `sizeof()`
  - Returns the number of bytes in variable or type.

Example: Integer overflow

- **What happens if an integer tries to get a value too big for its type (out of range)?**

```
#include <stdio.h>
int main()
{
   int i = 2147483647;
   printf("%i %i %i\n", i, i+1, i+2);
   return 0;
}
```

如何表示大整数？

```
2147483647 -2147483648 -2147483647
```

On this computer, `int` is stored on 32 bits: the first bit represents the sign (符号), the rest of 31 bits represent the value.
Biggest positive `int` value here: $2^{31}-1 = 2147483647$

- C99 为 C 语言扩展了新的整数类型 long long，通常被定义成 64 位宽 (8 byte)
- C99: long long int
- 格式输出： "%lld"

```
#include <stdio.h>
int main()
{
   long long int i = 2147483647;
   printf("%lld %lld %lld\n", i, i+1, i+2);
   return 0;
}
```

```
2147483647 2147483648 2147483649
```

课后思考：“大整数求和”如何实现？

回顾：C 的缺点

- **Shortcomings?**
  - **Exceptions (缺乏异常处理)**
  - **Range-checking (缺乏取值范围检查)**
  - **Garbage collection (缺乏垃圾收集机制)**
  - **Object-oriented programming (缺乏面向对象编程)**
  - **C语言不提供直接处理诸如字符串、集合、列表或数组等复合对象的操作；**
  - **C语言不直接提供多线程、并行操作、同步和协同的操作；**

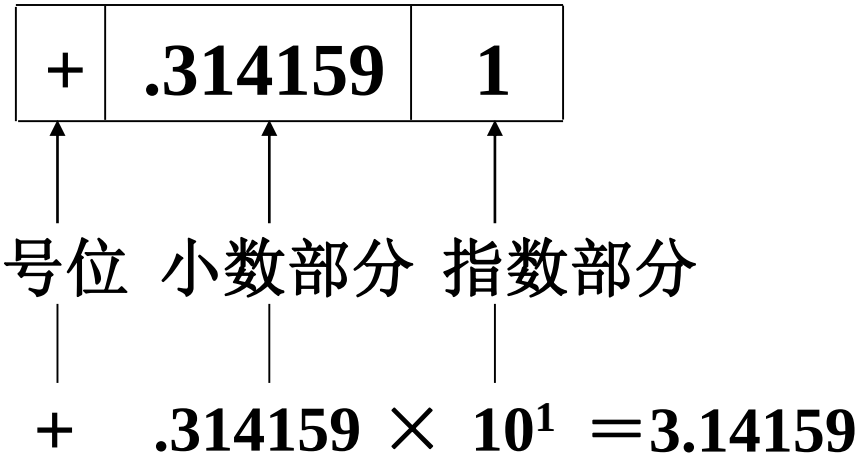
实数类型

实数类型（浮点类型）：分为单精度浮点类型（**float**）、双精度浮点类型（**double**）和长双精度浮点类型（**long double**）三种。

Examples: 3.，125.8，-.0001，1.7e4

实型数据的存放形式：小数部分＋指数部分

实数3.14159：



小数部分占的位数越多，数据的有效数字越多，精度越高；指数部分占的位数越多，则能表示的数值范围越大。

各种实型数据

类型	字节数	有效数字	数值范围
float	**4**	**6~7**	$10^{-38} \sim 10^{38}$
double	**8**	**15~16**	$10^{-308} \sim 10^{308}$
long double	**16**	**18~19**	$10^{-4932} \sim 10^{4932}$

浮点运算误差

- 浮点数要转换为二进制数值
- 计算机所使用二进制代码无法准确表示某些带小数位的十进制数据

十进制数值转换为二进制数值
- 整数部分："除2取余法“
- 小数部分："乘2取整法"

```
#include <stdio.h>
int main()
{
        float a = 0.65f;
        float b = 0.6f;
        float c = a - b;
        printf("%.10f\n", c);
        return 0;
}
```

```
0.0499999523
```

$(0.65)_{10} = (0.10100110011001100110011001100110011001100110011.....)_2$

$(0.6\ )_{10} = (0.10011001100110011001100110011001100110011001100.....)_2$

Knowing actual ranges for types

- **Defined in the include files `<limits.h>` and `<float.h>`**
- **`<limits.h>` contains system-dependent values that specify the sizes of various character and integer data types:**
  - **the maximum size of an `int` is `INT_MAX`**
  - **the maximum size of an `unsigned long int` is `ULONG_MAX`**
- **`<float.h>` gives floating-point data types.**
  - **`FLT_MAX` specifies the maximum floating-point number,**

```
…
#define SHRT_MIN    (-32768)        /* minimum (signed) short value */
#define SHRT_MAX      32767         /* maximum (signed) short value */
#define USHRT_MAX     0xffff        /* maximum unsigned short value */
#define INT_MIN     (-2147483647 - 1) /* minimum (signed) int value */
#define INT_MAX       2147483647    /* maximum (signed) int value */
…
```

Example: Using data types

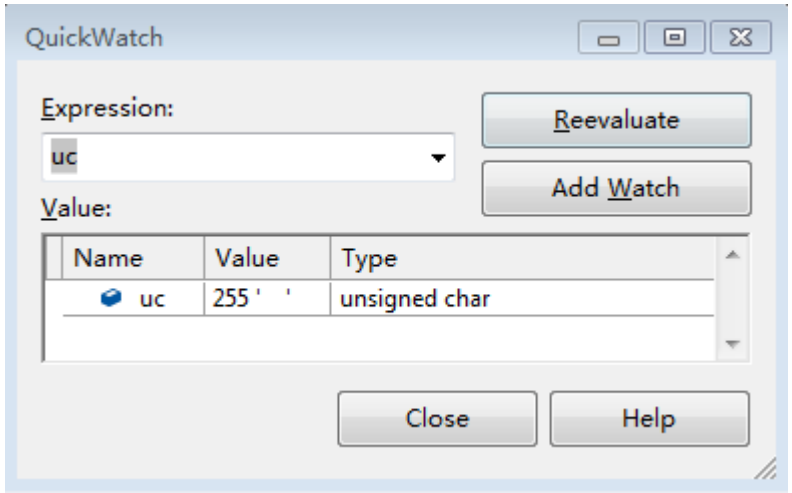
```
#include <stdio.h>
int main()
{
        int integerVar = 100;
        float floatingVar = 331.79;
        double doubleVar = 8.44e+11;
        char charVar = 'W';

        printf ("integerVar = %i\n", integerVar);
        printf ("floatingVar = %f\n", floatingVar);
        printf ("doubleVar = %e\n", doubleVar);
        printf ("doubleVar = %g\n", doubleVar);
        printf ("charVar = %c\n", charVar);
        return 0;
}
```

```
integerVar = 100
floatingVar = 331.790009
doubleVar = 8.440000e+011
doubleVar = 8.44e+011
charVar = W
```

Pop quiz I

- int x=017; int y=16; /*is x>y?*/
- short int s=0xFFFF12; /*correct?*/
- char c = -1; unsigned char uc = -1; /*correct?*/
- char c = 65 ; char c = "A" ; char c = 'A'; /*correct?*/
- float f = 12.5f; /*correct?*/



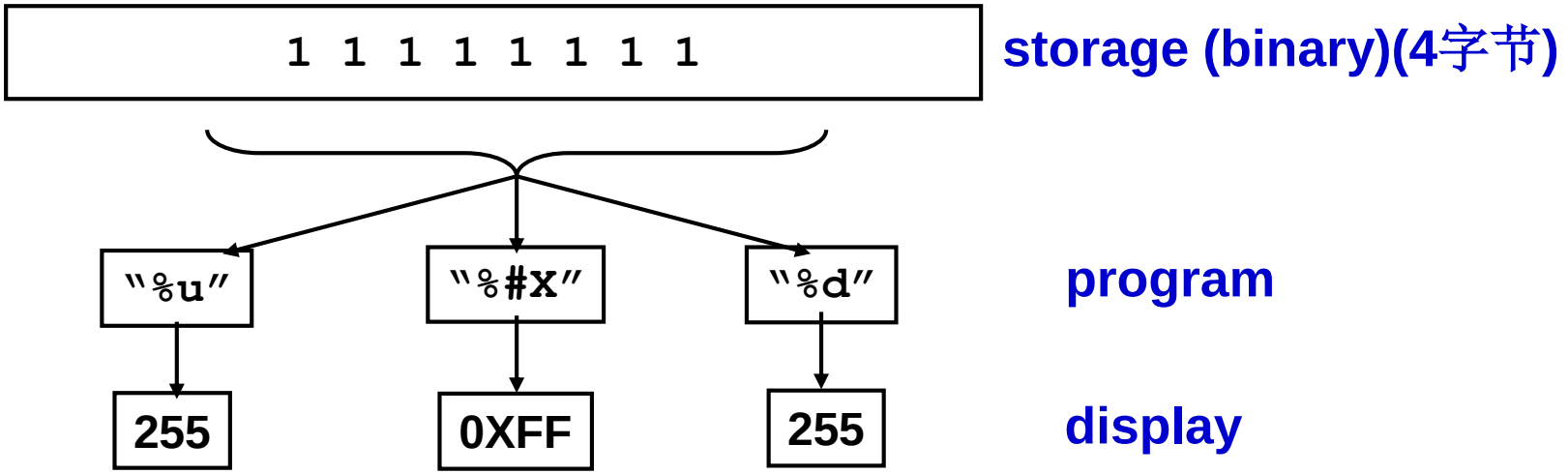
1. 负数的补码等于它的绝对值的二进制形式，按位取反再加1
2. 有符号整数的最高位被用作符号位。
   符号位: 0代表正数; 1代表负数
   Ref: P.21, 《计算机语言与程序设计》谌卫军

补充：unsigned负数转换成整数

- 用unsigned把负数转正并不改变变量的内存形态（即：二进制编码不会改变）；只是因为读取出来的时候，赋予不同的意义。

```
unsigned char seed = -1;
printf("unsigned int: %u\n", seed);
printf("int: %d\n", seed);
printf("int: %#X\n", seed);
```

```
unsigned int: 255
int: 255
int: 0XFF
```



Variables, Types and Operators

- Review
- Variables and data types
- **Arithmetic Operators**

基本算术运算符

+：加法运算符(addition)，如 3 + 5；

-：减法运算符(subtraction)，如 5 – 2；

*：乘法运算符(multiplication)，如 3 * 5；

/ ：除法运算符(division)。如 5 / 3。**两个整数相除，结果为整数，小数部分被舍去**；

%：模运算符(modulus)，或求余运算符(remainder)，**%两侧均为整型数据**，如 7 % 4。

- In C there are many more operators, we will learn later!
- Operators can be:
  unary (e.g., --, ++), binary (e.g., +, -, *, /), ternary (?:)

Example

```
#include  <stdio.h>
int   main( )
{
        int capital, earn;              // 原始资金、盈利
        double rate;                    // 利润率
        capital = 200;
        earn = 15;                                    7.5% ?
        rate = earn * 100 / capital;
        printf("%.1f%%", rate);         // 结果? 7.0%
        return 0;
}
```

自增和自减运算符

- **Increment (自增) and decrement (自减) operations**
- **Postfix (后缀) :**
  - **x++; /* x=x+1*/** **x−− ; /*x=x−1*/**
- **Prefix (前缀) :**
  - **++x; /* x=x+1*/** **−−x; /*x=x−1*/**
- **Often confusing:**
  - **y=x++; /*y=x; x=x+1*/**
  - **y=++x; /*x=x+1; y=x; */**
- **Example (x = 1)**
  - **y=x++; /* y=1; x=2;*/**
  - **y=++x; /*y=2; x=2; */**

```
int x;
for ( x = 1; x <= 10; x++ ){...}
```

```
a[i] = i++; /* correct? */
```

Example: 一名同学的疑问

```
#include <stdio.h>
int main()
{
   int n = 10;
   printf("n: %d n--: %d n: %d", n, n--, n);
   return 0;
}
```

```
n: 9 n--: 10 n: 9
```

我想让它出来的结果是：“10 10 9”，可为什么第一个是9？
是因为printf( ) 中 n-- 而导致的吗？
要是想输出“10 10 9”，只能做三个printf函数吗？

不建议在参数中使用 -- / ++，尤其对初学者来说，很容易犯错误，分不清谁先谁后。

赋值运算符

- **Assignment operators: var = var op expr**
  - **x=x+1**
  - **x=x∗10**
  - **x=x/2**
- **C provides compact assignment operators instead: var op= expr**
  - **x+=1 /∗is the same as x=x+1∗/**
  - **x−=1 /∗is the same as x=x−1∗/**
  - **x∗=10 /∗is the same as x=x∗10 ∗/**
  - **x/=2 /∗ is the same as x=x/2 ∗/**
  - **x%=2 /∗is the same as x=x%2 ∗/**

一些基本概念

- 常量 (Constants)：程序运行过程中其值不能被改变的量。
  - 整型常量：十进制形式（如200），八进制形式（如0200），十六进制形式（如0x200），二进制形式?
  - 实型常量：十进制小数形式（如7.0），指数形式（如321.54e6 ＝ 321.54×10^6）
  - 字符型常量：如'F'
- 表达式 (Expression)：有“值”的式子，通常由一些变量、常量、函数调用和运算符组合。

- 算术表达式：用算术运算符和括号将运算对象（也称操作数）连接起来的式子。
  - 运算符的优先级：在表达式求值时，先按运算符的优先级的高低次序执行，如先乘除后加减；
  - 运算符的结合性：若一个运算对象两侧的运算符的优先级别相等，则按规定的“结合方向”处理。算术运算符的结合方向为“从左到右”，即“左结合性”；
  - 分不清优先级和结合性：**加括号！加括号！加括号！**
- 赋值运算符 (Assignment Operators)：用赋值运算符 “=”把一个数据赋给一个变量。如a = 3。

- 类型转换 (Type Conversions)：把一种类型数据转换成另一种类型
  - 赋值转换（系统自动进行）：赋值运算符两侧的类型不一致，如：**rate = earn * 100 / capital**；
  - 运算转换（系统自动进行）：运算符带有不同数据类型的运算对象，如：**rate = earn * 100.0 / capital**；
  - 强制转换（程序员指定）：程序员使用强制类型转换运算符，如：**rate = (double)earn * 100 / capital**。

R进制 for integers

典型R进制：二进制 (Binary)、八进制 (Octal) 和十六进制 (Hexadecimal)，其共同之处都是进位计数制。

如果某种数制只采用R个基本符号，则称为基R数制（R进制），R称为数制的“基数”(Radix)，而数制中每一固定位置所对应的单位值称为“权”。

进位计数制的编码符合“逢R进一，借一当R”的规则,各个位的权是以R为底的幂，一个数可按照权展开成多项式。如一个十进制数$(2564)_{10}$可按权展开为：

$$2564 = 2\times10^3 + 5\times10^2 + 6\times10^1 + 4\times10^0$$

几种常用的进位数制

二进制	R＝2	基本符号	0, 1
八进制	R＝8	基本符号	0, 1, 2, 3, 4, 5, 6, 7
十进制	R＝10	基本符号	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
十六进制	R＝16	基本符号	0 – 9, A, B, C, D, E, F

其中，十六进制的符号A～F分别对应于十进制的10～15。

R进制转换为十进制

基数为R的数，只要将其各位数字与相应的权相乘，其积相加，和数就是相应的十进制数。

例：11001010_2

$=1\times2^1 + 1\times2^3 + 1\times2^6 + 1\times2^7$

$=202$

例：3407_8

$= 7\times8^0 + 0\times8^1 + 4\times8^2 + 3\times8^3 = 1799$

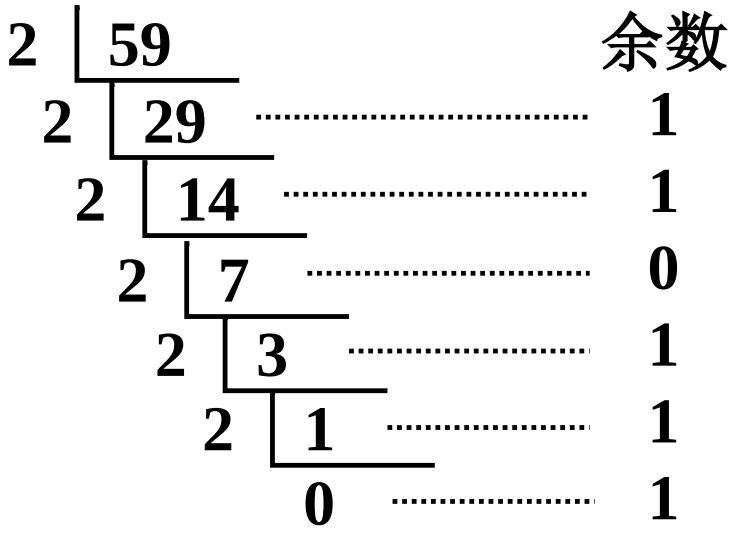
例：$3C_{16}$

$= C\times16^0 + 3\times16^1 = 60$

十进制转换为R进制

用该十进制数连续地除以R，得到的余数即为R系统的各位系数。此方法称为除R取余法。

例如：将59_{10}转换为二进制数：



所以：

$59_{10}=111011_2$

Data display

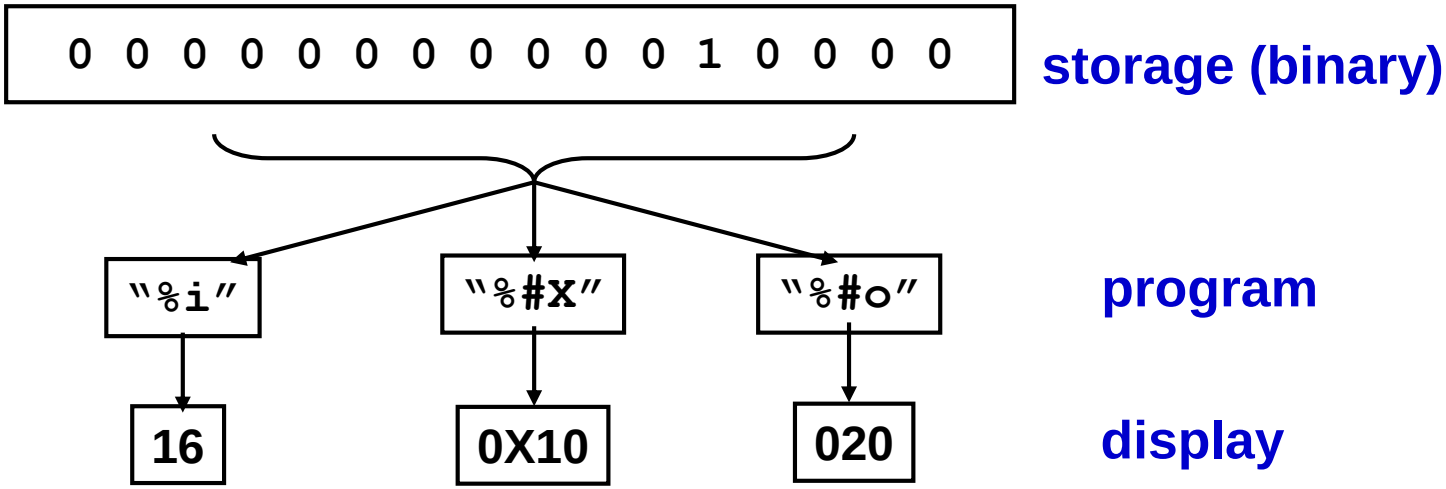
- **Octal format:**
  - `int` x = 016;
  - An integer value can be displayed in octal notation by using the format characters `%o` or `%#o` in the format string of a `printf` statement.
- **Hexadecimal format:**
  - `int` x = 0X16;
  - The format characters `%x`, `%X` , `%#x`, or `%#X` display a value in hexadecimal format
  - `itoa()`

Data storage

- **The option to use decimal, octal or hexadecimal notation doesn't affect how the number is actually stored internally !**

```
int a = 16;
/* Display decimals */
printf("Decimal %d as:\n  Hex: %#X  Octal: %#o \n", a, a, a);
```

```
Decimal 16 as:
  Hex: 0X10  Octal: 020
```



Lecture 2 - Summary

- **Topics covered:**
  - **Variables, data types, and arithmetic operators**
  - **R进制**