

# 程序设计基础

## Fundamental of Programming

清华大学软件学院

刘玉身

[liuyushen@tsinghua.edu.cn](mailto:liuyushen@tsinghua.edu.cn)

# Lecture 3 : Outline

---

- Statements and Compound Statements (语句和复合语句)
- Making Decisions (选择结构)
  - Relational Operators (关系运算符)
  - Logical Operators (逻辑运算符)
  - The **if** Statement (**if**语句)
    - The **if-else** Construct
    - **Nested if** Statements (嵌套**if**语句)
    - The **else-if** Construct
  - The **switch** Statement (**switch**语句)
  - The Conditional Operator (条件运算符)

# Lecture 4 : Outline

---

- **Program Looping (循环结构)**
  - The **for** Statement
    - Relational Operators
    - **Nested for** Loops
    - Increment Operator
    - Program Input
    - **for** Loop Variants
  - The **while** Statement
  - The **do-while** Statement
  - The **break** and **continue** Statement

# Lecture 3

---

- **Statements and Compound Statements**
- **Making Decisions**
  - **Relational Operators**
  - **Logical Operators**
  - **The `if` Statement**
    - The `if-else` Construct
    - **Nested `if` Statements**
    - The `else-if` Construct
  - **The `switch` Statement**
  - **The Conditional Operator**

# Lecture 3

---

- **Statements and Compound Statements**
- **Making Decisions**
  - Relational Operators
  - Logical Operators
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The `switch` Statement
  - The Conditional Operator

# Statements and Compound Statements

---

- **语句 (statement)**

- 以“**;**”作分隔符，用来完成一定操作任务

- `x = 0;`

- **复合语句 (compound statement/block)**

- 用一对花括号 **{...}** 把一组声明和语句括在一起就构成了一个**复合语句**（也叫程序块）

# Compound Statements（复合语句）

---

- 语法上等价于单条语句
- 变量可以在内部声明

```
{  
    int temp = x+y;  
    z = foo(temp);  
}
```

- 复合语句可以为空 { }
- 括号外没有分号

# 顺序结构程序设计

---

## 顺序结构程序设计

- ☺ **语句一条接一条地执行，没有分支、跳转等结构**
- ☺ **最简单的一种程序结构**





# Example 1: 公式计算

---

## 问题描述:

输入一个角度  $\alpha$ ，计算下列  $y$  的值:

$$y = \sqrt{\frac{1 - \cos \alpha}{2}}$$

---

## 问题分析：

- 需要哪些数据？
  - 定义两个`double`类型的变量：  
**alpha 和 y**
- 数学函数调用：余弦、平方根函数
  - `#include <math.h>`
  - `cos()`, `sqrt()`

---

```
double cos ( double x ) ;
```

- 功能：计算  $x$  的余弦值
- 说明： $x$  是弧度值。

$$\text{弧度} = \text{角度} * \pi / 180^\circ$$

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

```
int main( )
{
```

```
    double  alpha, y;
```

```
    printf("请输入一个角度: ");
```

```
    scanf("%lf", &alpha);
```

```
    y = sqrt((1 - cos(alpha / 180 * 3.14159)) / 2.0);
```

```
    printf("y = %.2f\n", y);
```

```
    return 0;
```

```
}
```

请输入一个角度: 30  
y = 0.26

- 对于float类型的变量, printf()中的说明符可以用%f或%lf, 而scanf()中的说明符则只能用%f
- 对于double类型的变量, printf()中的说明符可以用%f或%lf, 而scanf()中的说明符则只能用%lf
- 对于long double类型的变量, printf()中的说明符可以用%Lf, 而scanf()中的说明符则只能用%Lf

# Better Style

---

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159
```

```
int main()
{
    double alpha, y;

    printf("请输入一个角度: ");
    scanf("%lf", &alpha);
    y = sqrt((1 - cos(alpha / 180 * PI)) / 2.0);
    printf("y = %.2f\n", y);

    return 0;
}
```

**Macro Substitution (宏替换):**

**#define** *name replacement-text*

Please refer to P.89 [K&R]

## Example 2: 进制转换

---

### 问题描述:

输入一个十进制整数，然后以字符形式输出它的十六进制形式。说明：假设该整数所对应的十六进制形式有两位数字，且每位数字的值在'A' ~ 'F'之间。

例如：假设输入为186，则相应的输出为：十六进制：0xBA。

# 字符类型

---

## ◆ 字符类型

- ☺ 一个字符型数据只占用一个字节的空間。
- ☺ 双重属性：**整数属性**和**字符属性**。

## ◆ 整数属性

- ☺ 字符类型即**单字节的整数类型**。

## ◆ 字符属性

- ☺ 数据值即为相应字符的 **Ascii 码**。

ASCII 字符代码表 一

高四位   低四位		ASCII非打印控制字符										ASCII 打印字符												
		0000					0001					0010	0011		0100	0101		0110		0111				
		0					1					2	3		4	5		6		7				
		十进制	字符	ctrl	代码	字符解释	十进制	字符	ctrl	代码	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	ctrl		
0000	0	0	BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☹	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOF	传输结束	20	¶	^T	DC4	设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ	查询	21	♫	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	●	^G	BEL	震铃	23	↕	^W	ETB	传输块结束	39	'	55	7	71	G	87	w	103	g	119	w	
1000	8	8	◻	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◼	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	垂直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	换页/新页	28	└	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	🎵	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	Back space

注：表中的ASCII字符可以用:ALT + “小键盘上的数字键”输入



---

◆ 存放方式？ 存放内容？

☺ 二进制、ASCII值 (如'A', 65)

◆ 使用方式？

☺ 当成整数或字符

# 字符常量

---

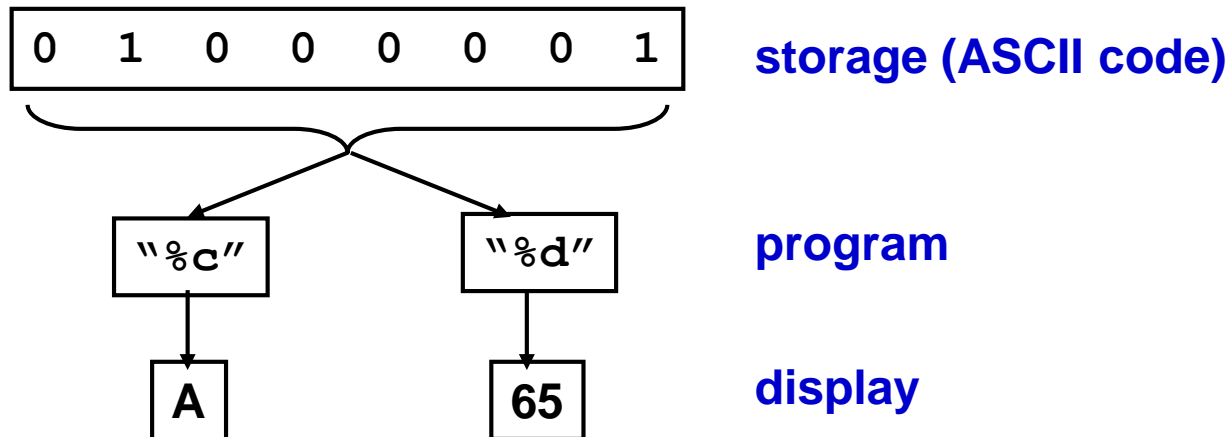
用单引号括起来的一个字符，如 'a'、  
'A'、'#'。

`('F' - 'A') + 'd' = 'i'`

# Data display vs data storage

```
/* displays ASCII code for a character */  
#include <stdio.h>  
int main()  
{  
    char ch;  
    ch='A';  
    printf("The code for %c is %d.\n", ch, ch);  
}
```

The code for A is 65.



# 转义字符（escape sequences）

符号	ASCII值	含义
<code>\a</code>	<b>007</b>	响铃
<code>\b</code>	<b>008</b>	退格
<code>\n</code>	<b>010</b>	换行
<code>\r</code>	<b>013</b>	回车
<code>\t</code>	<b>009</b>	水平Tab键
<code>\v</code>	<b>011</b>	竖直Tab键
<code>\'</code>	<b>039</b>	单引号 '
<code>\''</code>	<b>034</b>	双引号 "
<code>\\</code>	<b>092</b>	反斜杆 \
<code>\ooo</code>	—	八进制表示的字符
<code>\xhh</code>	—	十六进制表示的字符

Why 转义字符?

如:

`'\x41'`

---

转义字符通常用于代表难于表达的或无法键入的字符

```
printf("\a\a");  
printf("你\t好\n你有");  
printf("\\"计算机语言与程序设计\\"这本书吗?");
```

你 好  
你有"计算机语言与程序设计"这本书吗?

---

输入一个十进制整数，然后以字符形式输出它的十六进制形式（两位数字）。

```
printf("请输入一个整数:");  
scanf("%d", &value);  
  
printf("%x", value);
```

严打

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

```
int main()
```

```
{
```

```
    int value;
```

```
    int v1, v2;
```

```
    char c1, c2;
```

```
    printf("请输入一个整数:");
```

```
    scanf("%d", &value);
```

```
    v1 = value / 16;
```

```
    v2 = value % 16;
```

```
    c1 = v1 - 10 + 'A';
```

```
    c2 = v2 - 10 + 'A';
```

```
    printf("0x%c%c\n", c1, c2);
```

```
    return 0;
```

```
}
```

请输入一个整数:186

0xBA

# Lecture 3

---

- Statements and Compound Statements
- **Making Decisions**
  - Relational Operators
  - Logical Operators
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The `switch` Statement
  - The Conditional Operator



# Making Decisions (选择结构)

---



**Where to go?!**

# 假币问题

有五枚硬币，但其中有一枚是假币。已知假币比真币要轻。现有一架天平，请问：**最多**需要称几次就能把假币找出来？



---

答案：二次即可。

任取4枚硬币，放在天平上，一边2枚：

- *if* 左边的2枚比较轻，则把它们再称一次，轻者即为假币；
- 否则，*if* 右边的2枚比较轻，则把它们再称一次即可；
- 否则，剩下的那枚硬币是假币。

# Lecture 3

---

- Statements and Compound Statements
- Making Decisions
  - **Relational Operators**
  - Logical Operators
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The `switch` Statement
  - The Conditional Operator

# 关系运算符和关系表达式

所谓的“关系运算”实际上是“比较运算”。

C 语言提供了 6 种关系运算符：

- |        |         |   |           |
|--------|---------|---|-----------|
| (1) <  | (小于)    | } | 优先级相同 (高) |
| (2) <= | (小于或等于) |   |           |
| (3) >  | (大于)    |   |           |
| (4) >= | (大于或等于) |   |           |
| (5) == | (等于)    | } | 优先级相同 (低) |
| (6) != | (不等于)   |   |           |

例如:  $a > b == c$   
 $a == b < c$

等价于  $(a > b) == c$   
等价于  $a == (b < c)$

---

优先级高	↑	算术运算符: +、-、*、/、%
		关系运算符: >、<、==、>=、<=、!=
优先级低		赋值运算符: =

例如: **c > a + b**

等价于 **c > (a + b)**

**a = b > c**

等价于 **a = (b > c)**

---

## ◆ 关系表达式

☺ 用关系运算符将两个操作数连接起来的式子

## ◆ 关系表达式的值

☺ 逻辑值“真”或“假”。但 C 语言 (C89) 没有单独的逻辑型数据，而是以整数 1 来代表“真”、以 0 来代表“假”。

---

```
int x, y = 5;           // y初始化为5
x = (y < 4);            // x等于0
x = (y >= 5);           // x等于1
```

```
int flower;
int kong = 0, se = 1;
x = (flower != flower); // x等于0
x = (kong == se);       // x等于0
```



# 补充：布尔类型（C99）

---

- C++里有专门的**bool**类型，用来表示真或假
- C99标准中增加了**bool**类型
  - `#include <stdbool.h>`
  - 例如：`bool b = 1;`
- C99标准更新：
  - <http://www.comeaucomputing.com/techtalk/c99/#bool>
- 但vc中的C编译器还不完全支持C99
  - 没有`<stdbool.h>`

# Lecture 3

---

- Statements and Compound Statements
- Making Decisions
  - Relational Operators
  - **Logical Operators**
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The `switch` Statement
  - The Conditional Operator

# 逻辑运算符和逻辑表达式

---

用逻辑运算符将关系表达式或逻辑量连接起来的式子，就是**逻辑表达式**。

C 语言提供了 3 种逻辑运算符，包括：

- (1)    **&&**                      (逻辑与 (AND), 双目运算符)
- (2)    **||**                        (逻辑或 (OR), 双目运算符)
- (3)    **!**                         (逻辑非 (NOT), 单目运算符)

# 逻辑运算符

---

Operator	Symbol	Meaning
AND	&&	<b>x &amp;&amp; y</b> is true if BOTH <b>x</b> and <b>y</b> are true
OR		<b>x    y</b> is true if at least one of <b>x</b> and <b>y</b> is true
NOT	!	<b>!x</b> is true if <b>x</b> is false

## 1. 逻辑运算的判断:

**true = non-zero, false=zero**

## 2. 表达式的结果:

**true = 1, false=zero**

---

**a && b**

<b>a \ b</b>	真	假
真	真	假
假	假	假

**a || b**

<b>a \ b</b>	真	假
真	真	真
假	真	假

**!a**

<b>a</b>	真	假
<b>!a</b>	假	真

# 运算符优先级

优先级高

优先级低

**Precedence**

!, ++, --, (type)

\*, /, %

+, -

<, <=, >, >=, ==, !=

&&

||

=

例如:

**$a > b \ \&\& \ b > c \ || \ b > d$**

等价于:

**$((a > b) \ \&\& \ (b > c)) \ || \ (b > d)$**

**加括号! 加括号! 加括号!**

# 两个需要注意的问题

---

(1) 非 0 即为真。在逻辑表达式中，运算对象的值如果为 0，则视为“假”；否则视为“真”。

例如:

<code>y = -4 &amp;&amp; 5;</code>	<code>// y = 1</code>
<code>y = (! 'c')    0;</code>	<code>// y = 0</code>
<code>y = 'c' &amp;&amp; 5.2;</code>	<code>// y = 1</code>

(2) 在逻辑表达式的求解中，并非所有逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符。

```
y = (3==3) || ((c=getchar())=='y');
```

```
y = (0) && ((x=x+1)>0);
```

# Lecture 3

---

- Statements and Compound Statements
- Making Decisions
  - Relational Operators
  - Logical Operators
  - The **if** Statement
    - The **if-else** Construct
    - Nested **if** Statements
    - The **else-if** Construct
  - The **switch** Statement
  - The Conditional Operator



# if 语句

---

**if 语句**是一种分支语句，它用来判定所给出的条件是否满足，然后根据判定的结果（真或假）来决定执行相应的操作。

# if 语句的形式之一

---

**if (表达式) 语句1;      // 语句1, 只一句**

**如果表达式为真, 执行语句1; 否则什么都不做**

例如:

**if (x > y)    printf(“最大值是 %d”, x);**

**if (temperature > 38)**

**printf("You have a fever. \n");**

**printf("Go see the doctor \n");**

有点问题吧?

---

**if (表达式)**

**{**

**语句1;**

**语句2;**

**.....**

**}**

**引入复合语句：若表达式为真，执行复合语句当中的每一条语句；否则什么都不做；**

```
if (temperature > 38)
```

```
{
```

```
    printf("You have a fever. \n");
```

```
    printf("Go see the doctor \n");
```

```
}
```

# if 语句的形式之二

---

```
if (表达式) 语句1;  
else 语句2;
```

**如果表达式为真，执行语句1；否则执行语句2**

其中，语句1和语句2可以是单条语句，也可以是复合语句。

# 计算绝对值

---

## 问题描述:

计算 $x$ 的绝对值 $|x|$ ，把结果保存在`abs`变量中。

### 方案1

```
if (x >= 0) abs = x;  
if (x < 0 ) abs = -x;
```

### 方案2

```
abs = x;  
if (x < 0) abs = -x;
```

### 方案3

```
if (x >= 0) abs = x;  
else abs = -x;
```

**哪一个正确?**

# if 语句的形式之三

---

```
if (表达式1) 语句1;  
else if (表达式2) 语句2;  
else if (表达式3) 语句3;  
.....  
else if (表达式m) 语句m;  
else 语句m+1;
```

层叠的 if  
语句

如果表达式1为真，执行语句1；否则如果表达式2为真，执行语句2；否则如果表达式3为真，执行语句3；否则，...，如果表达式m为真，执行语句m，否则执行语句m+1

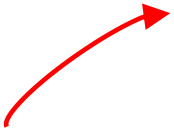
# 寻找死代码

```
1  if (speed >= 0)
2  {
3      printf("You don't go backward.\n");
4      if (speed == -1) {
5          printf("Wait! I was wrong!");
6      }
7  }
8  else if (speed > 0) {
9      printf("You go forward.\n");
10 }
11 else if (speed < 0) {
12     printf("You go backward.\n");
13 }
14 else {
15     printf("What did you do?!\n");
16 }
```

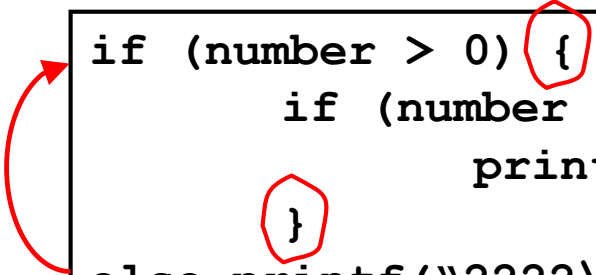


# Bug 1

```
if (number > 0)
    if (number < 10)
        printf("1111\n");
else printf("2222\n");
```



```
if (number > 0) {
    if (number < 10)
        printf("1111\n");
}
else printf("2222\n");
```



1. **if-else**配对原则：缺省{ }时，**else**总是和它上面离它最近的未配对的**if**配对；
2. 实现**if-else**正确配对方法：加{ }



# Bug 2

---

## 世上倒数第二个C Bug

```
if ( 0 <= x <= 10 )  
    printf ( "x is between 0 and 10. \n " ) ;
```

**YES**

```
if (x >= 0 && x <= 10)
    printf ( "x is between 0 and 10. \n " ) ;
```

**NO!**

```
if (0 <= x <= 10)
    printf ( "x is between 0 and 10. \n " ) ;
```

**Syntactically correct, but semantically an error !!!**  
**语法正确，但语义错误!!!**

**等价于：  $(0 \leq x) \leq 10$**

**其中：  $(0 \leq x)$  等于0或1，因此  $(0 \leq x \leq 10)$  永远为1**

# Bug 3

---

## 世上最后一个C Bug

```
status = check_radar ( ) ;  
if (status = 1)  
{  
    launch_nuclear_missiles ( ) ;  
}
```

**July 28, 1962 -- Mariner I space probe.** A bug in the flight software for the Mariner 1 causes the rocket to divert from its intended path on launch. Mission control destroys the rocket over the Atlantic Ocean. The investigation into the accident discovers that a formula written on paper in pencil was improperly transcribed into computer code, causing the computer to miscalculate the rocket's trajectory.

# Lecture 3

---

- Statements and Compound Statements
- Making Decisions
  - Relational Operators
  - Logical Operators
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The **`switch`** Statement
  - The Conditional Operator

# Example: Multiple choices

---

层叠的 if 语句:

```
if (rank == 1){  
    printf("冠军\n");  
    points = 10;  
}  
else if (rank == 2){  
    printf("亚军\n");  
    points = 5;  
}  
else if (rank == 3) {  
    printf("季军\n");  
    points = 2;  
}  
else {  
    printf("鼓励奖" );  
    points = 1;  
}
```

# switch 语句

表达式，其结果为整数

整数常量

switch (rank)

{

case 1:

printf("冠军\n");  
points = 10;  
break;

case 2:

printf("亚军\n");  
points = 5;  
break;

case 3:

printf("季军\n");  
points = 2;  
break;

default:

printf("鼓励奖");  
points = 1;

}

# switch 语句

- 先计算“**表达式**”值，再匹配**case**后的常量；
- “**表达式**”值必须是整数(包括字符) (**No float !**)；
- **case**后面的语句可是多条；
- **case**语句块后面一般应加**break**语句，否则将继续执行；

```
switch (expression)
{
    case value1:
        program statement;
        ...
        break;
    case value2:
        program statement;
        ...
        break;
    ...
    default:
        program statement
        program statement
        ...
        break;
}
```

```

#include <stdio.h>
int main ()
{
    float v1, v2;
    char op;
    printf ("Type in your expression.\n");
    scanf ("%f %c %f", &v1, &op, &v2);
    switch (op) {
        case '+': printf ("%f\n", v1+v2); break;
        case '-': printf ("%f\n", v1-v2); break;
        case '*': printf ("%f\n", v1*v2); break;
        case '/':
            if ( v2 == 0 )
                printf ("Division by zero.\n");
            else
                printf ("%f\n", v1/v2);
            break;
        default: printf ("Unknown operator.\n"); break;
    }
    return 0;
}

```

Type in your expression.

3.0 \* 5.0

15.00

Type in your expression.

4.0 / 0

Division by zero.



# 闰年问题

---

问题描述:

编写一个程序，判断某一年是否是闰年  
(`leap year`) (讨论)

**1. 闰年满足什么规则?**

---

问题分析：

**“四年一闰，百年不闰，四百年再闰”**

1. 能被4整除，但不能被100整除的年份都是闰年，如1996年和2004年；
2. 能被100整除，又能被400整除的年份是闰年，如1600年和2000年；
3. 除此之外的所有年份，都不是闰年。

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

```
int main (void)
```

```
{
```

```
    int year, rem_4, rem_100, rem_400;
```

```
    printf ("Enter the year to be tested: ");
```

```
    scanf ("%i", &year);
```

```
    rem_4 = year % 4;
```

```
    rem_100 = year % 100;
```

```
    rem_400 = year % 400;
```

```
    if ( (rem_4 == 0 && rem_100 != 0) || rem_400 == 0 )
```

```
        printf ("It's a leap year.\n");
```

```
    else
```

```
        printf ("It's not a leap year.\n");
```

```
    return 0;
```

```
}
```

**Enter the year to be tested: 1996**

**It's a leap year.**

**Enter the year to be tested: 2000**

**It's a leap year.**

# 进制转换2

---

## 问题描述:

编写一个程序，输入一个255以内的正整数，然后以字符形式输出它的十六进制形式。例如：假设输入为140，则相应的输出为：十六进制：0x8C。 (讨论)

1. 该十六进制数有几位？
2. 每一位的取值范围？

---

## 问题分析：

1. 该数小于**255**，则相应的十六进制数最多为2位（可能只有1位）；
2. 对于每一位十六进制数，它既可能大于也可能小于10，若大于10，需要转换。

如何变成流程图？

---

# **1. 该十六进制数只有1位**

**1.1 小于10**

**1.2 大于或等于10**

# **2. 该十六进制数有2位**

**2.1 处理第1位 (大于或小于10)**

**2.2 处理第2位 (大于或小于10)**

```
#include <stdio.h>
int main()
{
    int value, v1, v2;
    char c1, c2;

    printf("请输入一个整数:");
    scanf("%d", &value);

    if(value < 10)
    {
        printf("0x0%c\n", value + '0');
        return 0;
    }
    else if(value < 16)
    {
        printf("0x0%c\n", value - 10 + 'A');
        return 0;
    }
}
```

```
v1 = value / 16;
v2 = value % 16;

if(v1 < 10)      c1 = v1 + '0';
else  c1 = v1 - 10 + 'A';

if(v2 < 10)  c2 = v2 + '0';
else  c2 = v2 - 10 + 'A';

printf("十六进制: 0x%c%c\n", c1, c2);
return 0;
}
```

能否改进…



---

## 几次运行结果

请输入一个整数:9  
0x09

请输入一个整数:12  
0x0C

请输入一个整数:140  
十六进制: 0x8C

---

# Lecture 3

---

- Statements and Compound Statements
- Making Decisions
  - Relational Operators
  - Logical Operators
  - The `if` Statement
    - The `if-else` Construct
    - Nested `if` Statements
    - The `else-if` Construct
  - The `switch` Statement
  - **The Conditional Operator**

# 条件运算符

***condition ? expression1 : expression2***

- 先求解`condition`（条件），
- 如果`condition`为TRUE (nonzero)，则求解`expression1`，其值作为结果；
- 如果`condition`为FALSE (zero)，则求解`expression2`，其值作为结果。

```
maxValue = ( a > b ) ? a : b;
```

等价于：

```
if ( a > b )  
    maxValue = a;  
else  
    maxValue = b;
```

# 进制转换3

```
#include<stdio.h>
int main()
{
    int value;
    int v1,v2;
    char c1,c2;

    printf("请输入一个整数:\n");
    scanf("%d",&value);
    v1=value/16;
    v2=value%16;

    c1 = (v1>=10) ? (v1-10+'A') : (v1+'0');
    c2 = (v2>=10) ? (v2-10+'A') : (v2+'0');

    printf("0x%c%c\n",c1,c2);
    //printf("%#X\n",value);
    return 0;
}
```

---

## 几次运行结果

请输入一个整数:9  
0x09

请输入一个整数:12  
0x0C

请输入一个整数:140  
十六进制: 0x8C

# 思考题

---

- **任意长度十进制整数转换到十六进制?**

```

#include <stdio.h>
int main()
{
    char temp[50];
    int i = 0, k, t;
    long n;
    char c[] = "0123456789ABCDEF";
    printf("put the algorism number:");
    scanf("%ld", &n);
    printf("translate %d to...(2,8,16)?:", n);
    scanf("%d", &k);
    while(n != 0) {
        t = n % k;
        n /= k;
        temp[i++] = c[t];
    }
    printf("the digit transfered is:");
    for(i -= 1; i >= 0; i--)
        putchar(temp[i]);
    putchar('\n');
    return 0;
}

```

put the algorism number:186  
translate 186 to...(2,8,16?):16  
the digit transfered is:BA



# Lecture 3 - Summary

---

- **Topics covered:**
  - **Statements and Compound Statements**
  - **Making Decisions**
    - **Relational Operators**
    - **Logical Operators**
    - **The if Statement**
      - **The if-else Construct**
      - **Nested if Statements**
      - **The else-if Construct**
  - **The switch Statement**
  - **The Conditional Operator**