

第4章 选择结构程序设计（1）

S. IST@XMU

复习回顾

Ø 上次课的内容：

u printf与scanf

l 格式控制符

u putchar与getchar

u C语言的流程

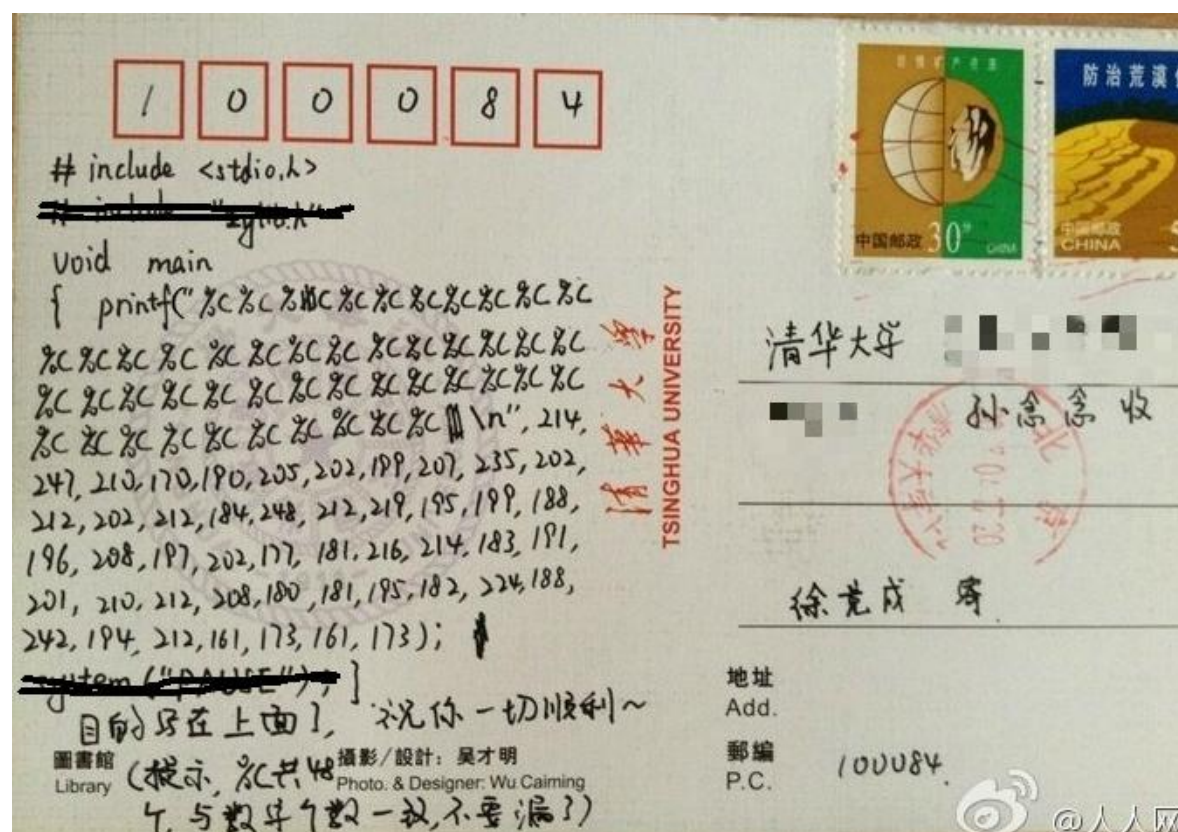
l 流程图

l 顺序、选择、循环

u 顺序结构程序设计

l 四个步骤

u 挑战一下：试试解密一封比较装13的明信片.....



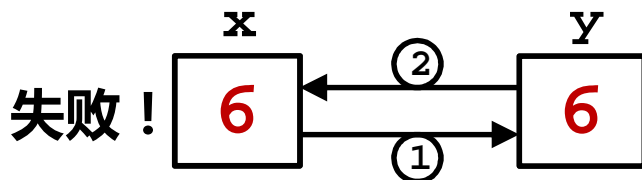
顺序结构程序举例：交换1

Ø 输入两个整数x和y，交换二者的值，然后输出。

└ 样例输入：6 8

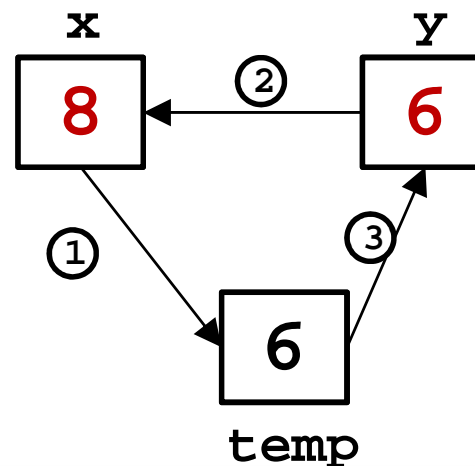
└ 样例输出：8 6

【分析】首先将输入存入整型变量x和y，然后进行交换。最经典的交换方法是**三变量法**：



【友提】是不是类似于加一个空瓶子，把一瓶油和一瓶醋对调的过程？

成功！



顺序结构程序举例：交换2

Ø 输入两个整数x和y，交换二者的值，然后输出。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x, y, temp;
5.     scanf("%d%d", &x, &y);
6.     temp = x;    //将x的初值赋予变量temp
7.     x = y;       //仅改变变量x的值，y的值不变
8.     y = temp;    //变量y被赋予新的值，原值被覆盖
9.     printf("%d %d\n", x, y);
10.    return 0;
11. }
```

【思考】如果仅定义两个变量，真的无法实现交换吗？

6	8	✓
8	6	

顺序结构程序举例：交换3

Ø 输入两个整数x和y，不用定义其他变量，实现交换二者的值，然后输出。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x, y;
5.     scanf("%d%d", &x, &y);
6.     x = x+y;
7.     y = x-y;
8.     x = x-y;
9.     printf("%d %d\n", x, y);
10.    return 0;
11. }
```

【启示】 同一个问题往往有多种解决办法，正确的不代表就是最优的。

源代码选美

```
1. #include <stdio.h>
2. int main(){char c1,c2;
3. c1=getchar();c2=c1+32;
4. putchar(c2);putchar( '\n' );
5. return 0;
6. }
```



```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     char c1,c2;
6.
7.     c1 = getchar();
8.     c2 = c1+32;
9.     putchar(c2);
10.    putchar( '\n' );
11.
12.    return 0;
13.}
```

初窥良好的编程风格（一）

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     char c1, c2;
6.
7.     c1 = getchar();
8.     c2 = c1+32;
9.     putchar(c2);
10.    putchar( '\n' );
11.
12.    return 0;
13.}
```

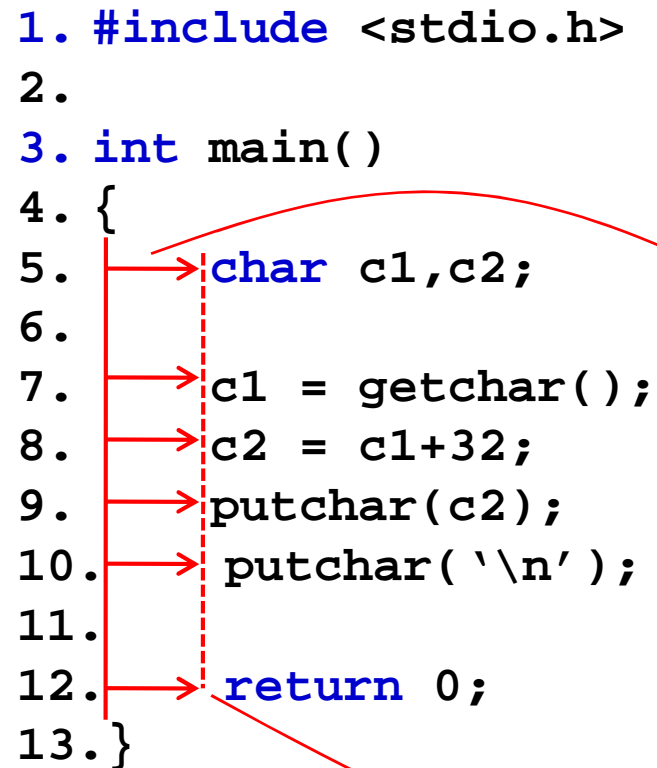
Ø 大括号

┌ 大括号单独成行

└ 成对的大括号应对齐

初窥良好的编程风格（二）

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     → char c1, c2;
6.
7.     → c1 = getchar();
8.     → c2 = c1+32;
9.     → putchar(c2);
10.    → putchar( '\n' );
11.
12.    → return 0;
13. }
```



Ø语句

u 一条语句占一行

u 语句要有层次感

l 语句起始位置相对于大括号的位置应缩进（用tab或空格）

l 同层次的语句应对齐

初窥良好的编程风格（三）

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     char c1,c2;
6.
7.     c1 = getchar();
8.     c2 = c1+32;
9.     putchar(c2);
10.    putchar( '\n' );
11.
12.    return 0;
13.}
```

Ø空行

u 预处理命令结束后
留有空行

u 声明部分和执行部
分之间有空行

u 算法不同部分之间
有空行

吐槽：教材上的代码真心不漂亮

Ø 比如，教材P66，例3.5，“漂亮”的代码应该写成这样

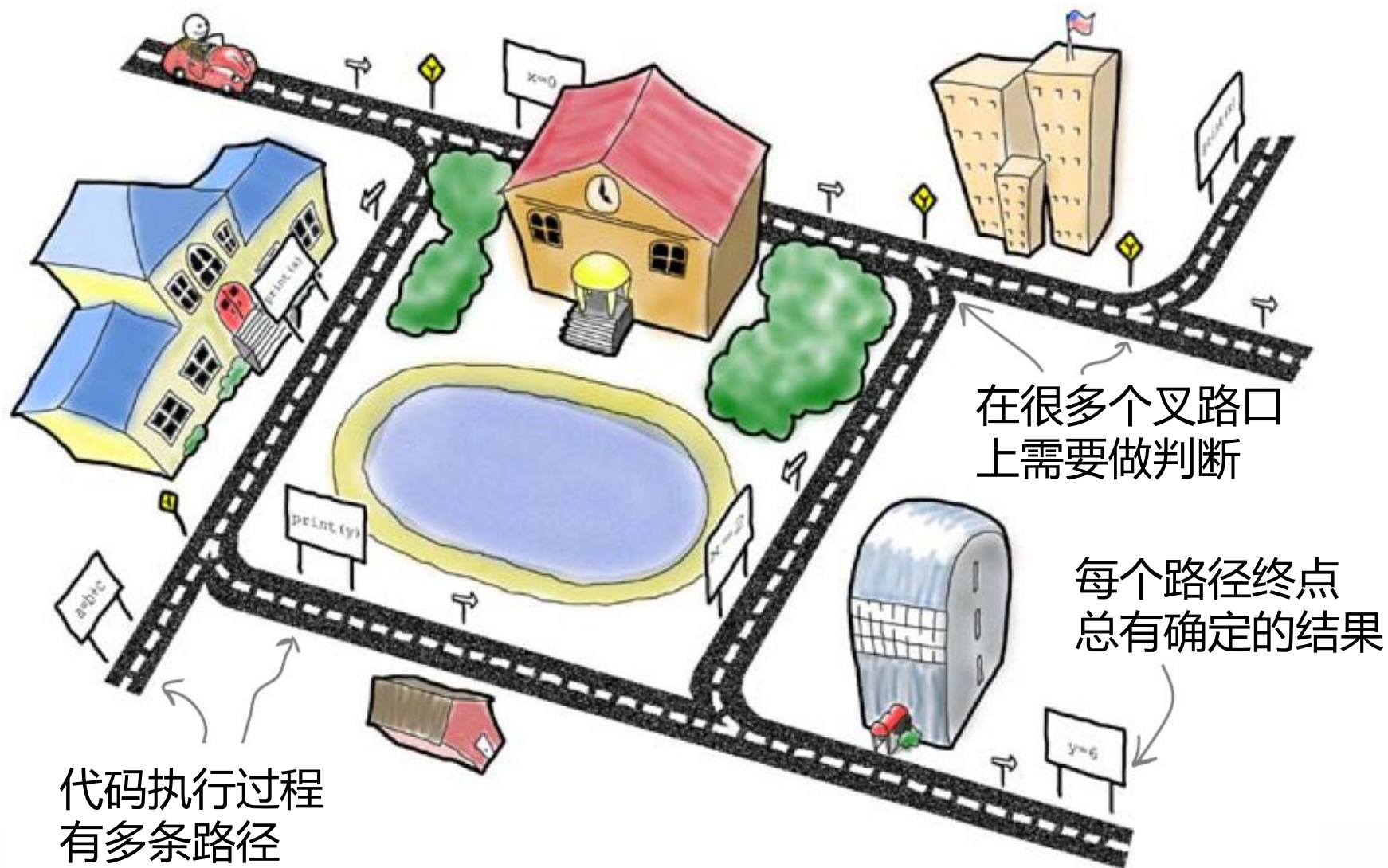
```
1. #include <stdio.h>
2. #include <math.h>
3.
4. int main()
5. {
6.     double a,b,c,disc,x1,x2,p,q;           // 变量定义
7.
8.     scanf("%lf%lf%lf", &a, &b, &c);        // 输入参数
9.
10.    disc = b*b-4*a*c;                       // 计算判别式disc
11.    p = -b/(2.0*a);
12.    q = sqrt(disc)/(2.0*a);
13.    x1 = p+q;                               // 计算方程的根x1,x2
14.    x2 = p-q;
15.
16.    printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);   // 输出结果
17.
18.    return 0;
19. }
```

【友提】适当的注释可以使代码可读性大大提高。

程序并不总是如此简单

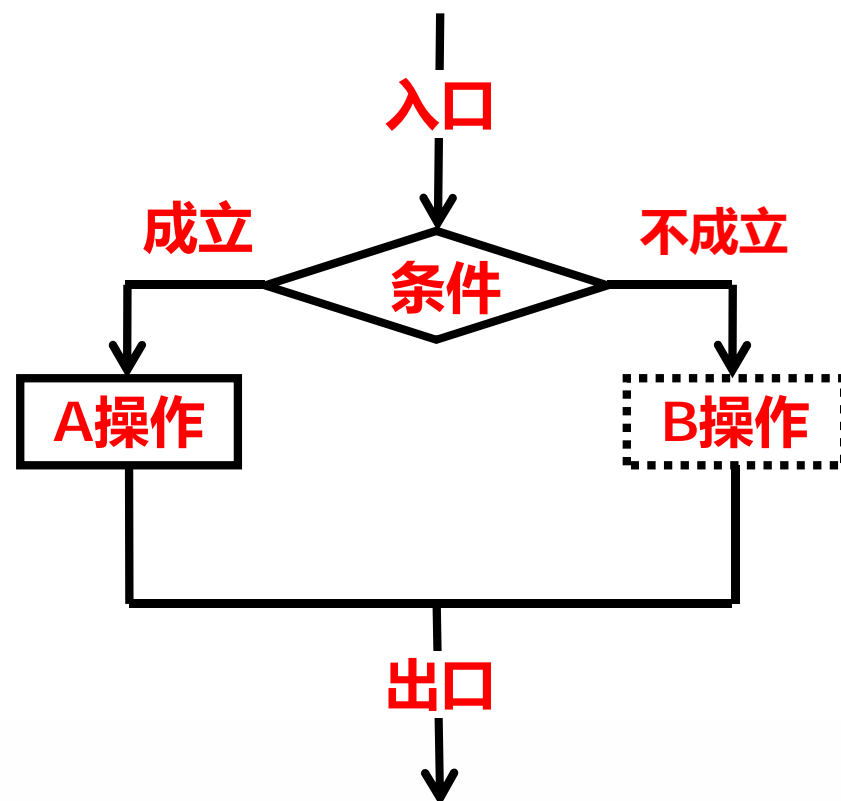


程序通常像复杂的路网



什么是选择结构

Ø 又叫分支结构，就是通过进行一个判断，在若干个可选的语句序列之间进行选择执行。



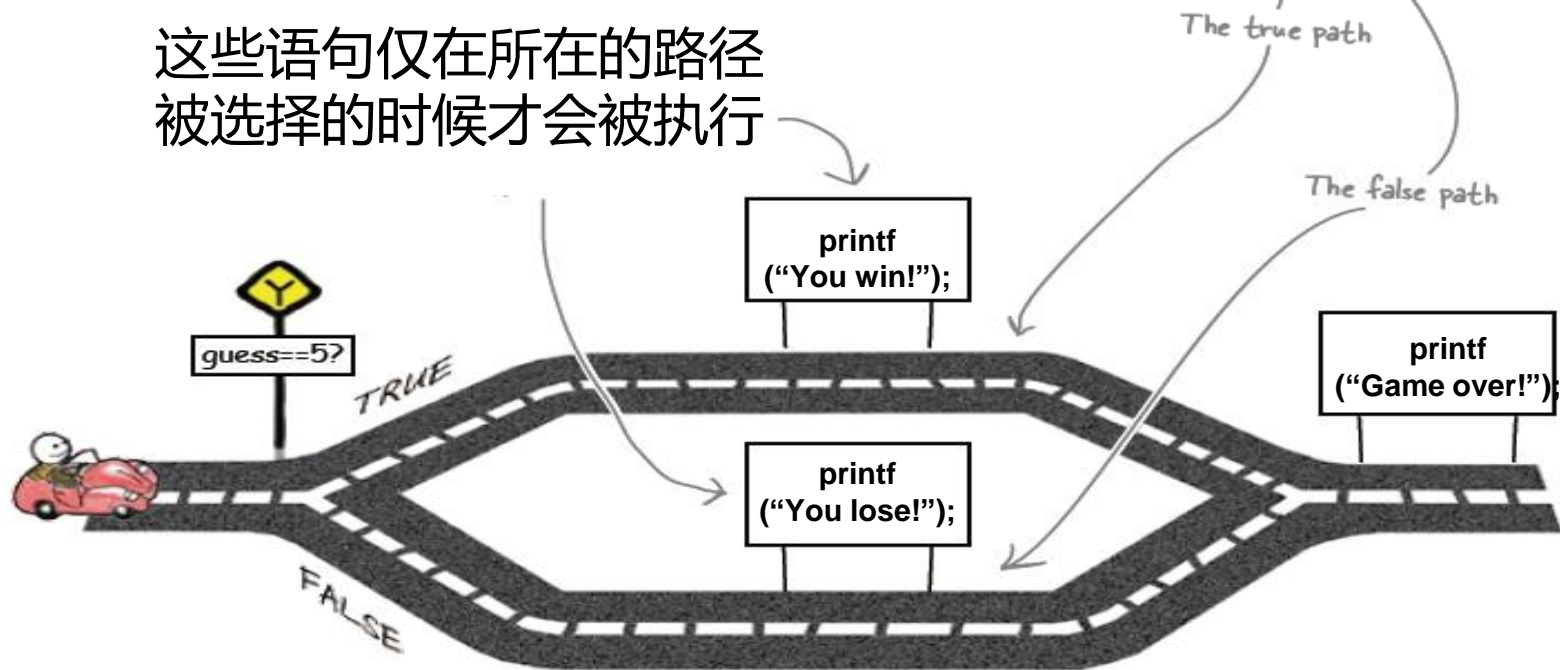
简单实例：一个猜数字游戏

```
1. #include <stdio.h>
2. int main()
3. {
4.     int guess;
5.     printf("Guess the number:");
6.     scanf("%d",&guess);
7.     if (guess == 5)
8.         printf("You win!\n");
9.     else
10.        printf("You lose!\n");
11.    printf("Game over!\n");
12.    return 0;
13.}
```

“猜数字” 游戏代码图解

```
1. if (guess == 5)
2.     printf("You win!\n");
3. else
4.     printf("You lose!\n");
```

这些语句仅在所在的路径
被选择的时候才会被执行



再做句子组合练习

Ø 用“如果...”把下列分句组合成一个完整的句子

└ 人的正常体温是 36°C - 37°C 。

└ 体温在 37.3°C - 38°C 之间叫低烧。

└ 体温在 38.1°C - 42°C 之间叫高烧。

└ 体温在 42°C 以上叫... 燃烧

Ø 组合：如果人的体温在 $\times\times\times\times\times$ ，那么状态是 $\times\times$ 。

└ 注意：

┆ 必有一个判断条件（如，体温）

┆ 分句之间没有先后顺序。

Ø C语言中的选择结构就是对先某种条件进行判断，然后按照是否满足条件去执行顺序相同的若干个语句中的一个。

条件判断不一定带“如果”

Ø碰到下列描述应该能自动抽取出判断条件

- └ 周末我们去郊游。（需要判断是否周末）

- └ 1.4 以下儿童免车票。（需判断身高年龄）

- └ 衣冠不整不得入内。（需判断着装）

Ø另外，还有些“夸张”的标语，比如：

- └ 放火烧山，牢底坐穿.....

- └ 一人超生，全村结扎.....

选择结构是靠选择语句实现的

ØC语言有两种选择语句：

(1) **if语句**，一个语句可实现两个分支的选择结构

(2) **switch语句**，一个语句可实现多分支的选择结构

关于if语句的解释

Ø **if语句**用来判断所给定的条件是否满足，根据判定结果的真或假来决定执行给出的两种操作中的某一种。

Ø 判断结果**真或假**是看条件表达式的值是非0值还是0值。**非0值为真，0值为假！**

Ø **三种形式**

- u 基本形式：if...

- u if...else...形式

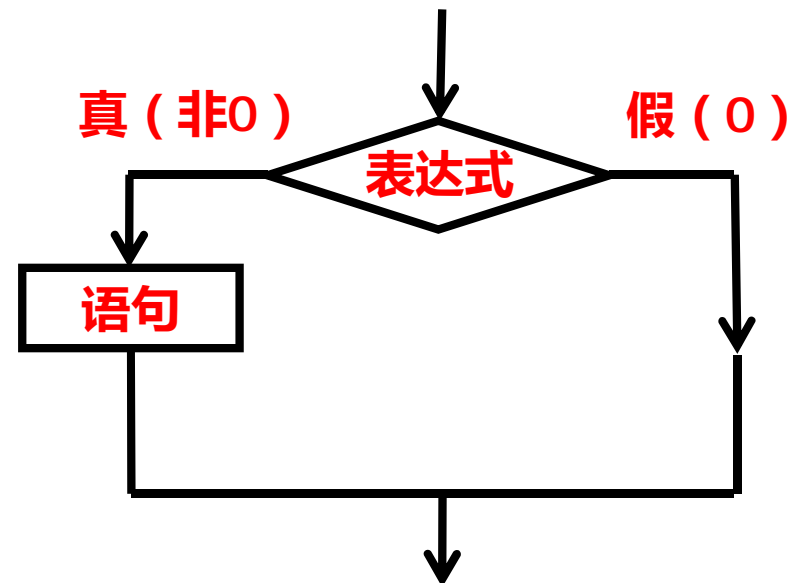
- u else if形式

if...语句

Ø 一般形式：**if(表达式) 语句;**

Ø 语义：选择是否执行某个动作。如果表达式的值为真（非0），则执行其后的语句；如果表达式的值为假（0），则不执行该语句。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int n;
5.     scanf("%d",&n);
6.     if (n != 0) //后接简单语句
7.         printf("n不等于0\n");
8.     if (n > 0) //后接复合语句
9.     {
10.        n = 0;
11.        printf("n一开始大于0\n");
12.        printf("n现在等于0\n");
13.    }
14.    return 0;
15. }
```



【思考】第6行改成if(n)可以吗？

if...else语句

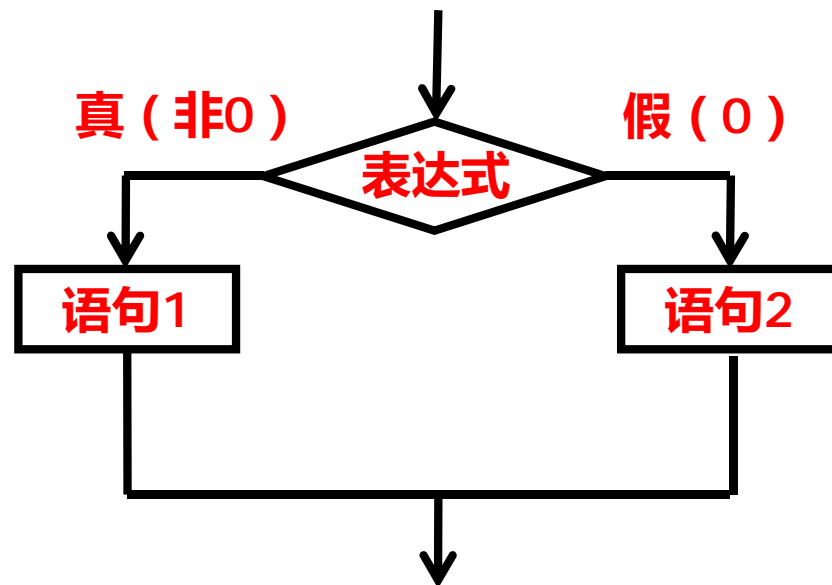
Ø 一般形式：if (表达式)

语句1;

else

语句2;

Ø 语义：在两个动作间直接进行选择。如果表达式的值为真（非0）则执行语句1；如果值为假（0），则执行跟在else后面的语句2。

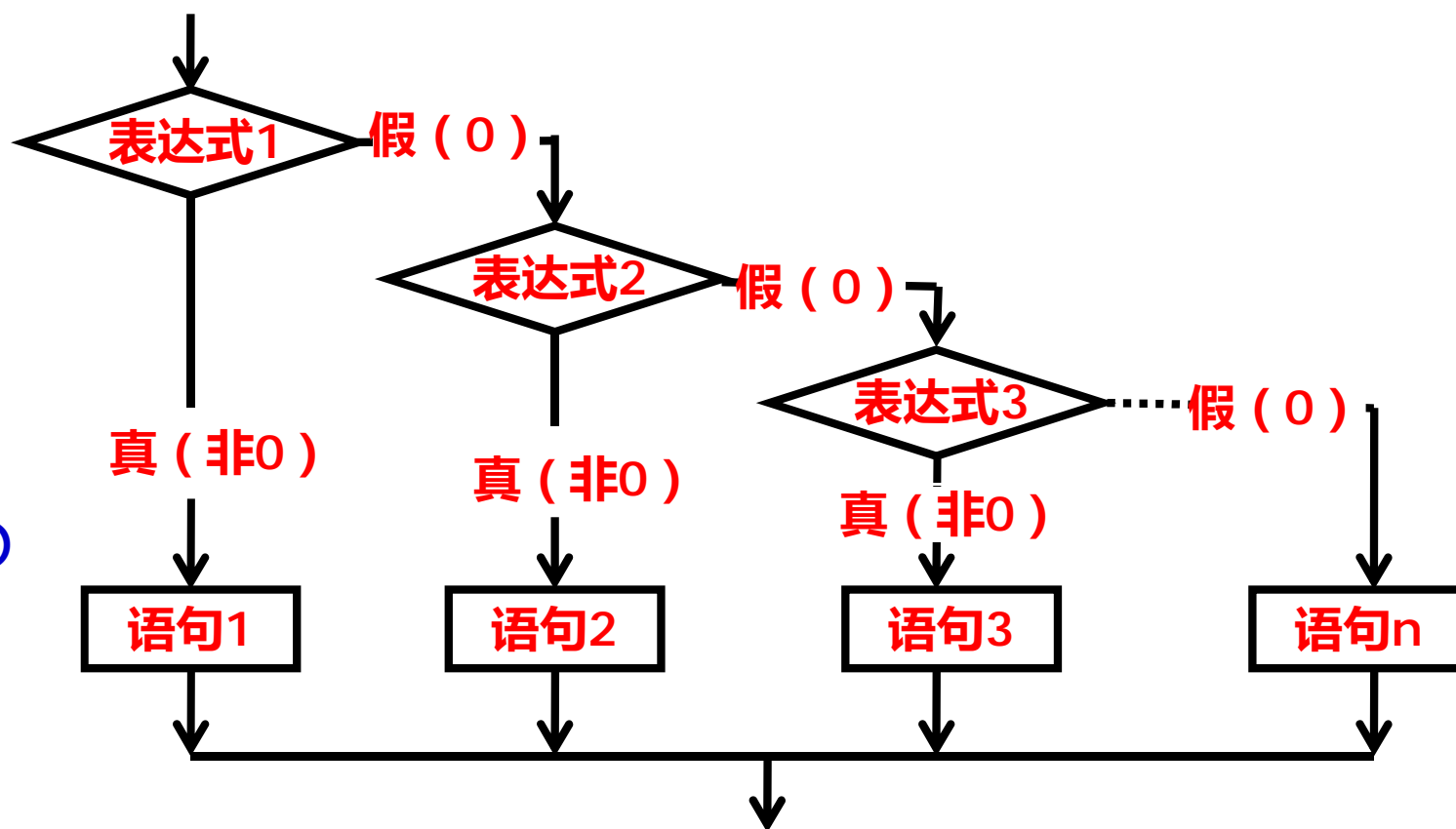


```
1. #include <stdio.h>
2. int main()
3. {
4.     int n;
5.     scanf("%d",&n);
6.     if (n >= 0)
7.         printf("n不小于0\n");
8.     else
9.     {
10.        printf("n一开始小于0\n");
11.        n = 0;
12.        printf("n现在等于0\n");
13.    }
14.    return 0;
15. }
```

else if 语句

Ø 一般形式：

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



Ø 语义：用于有两个以上选择的情况。依次判断表达式的值，当出现某个值为真时，则执行其对应的语句，然后跳到整个if语句之外继续执行程序。如果所有表达式均为假，则执行语句n，然后继续执行后续程序。

else if语句举例

Ø 编一程序，由键盘输入字符，然后显示该字符是否是一个数字字符、大写字母、小写字母、一个空格或其它字符

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c;
5.     printf("请按任意一个字符键:");
6.     c = getchar();
7.     printf("\n输入的字符 %c 是 ", c);
8.     if (c>='0' && c<='9' )
9.         printf("一个数字\n");
10.    else if (c>='A' && c<='Z' )
11.        printf("一个大写字母\n");
12.    else if (c>='a' && c<='z' )
13.        printf("一个小写字母\n");
14.    else if (c==' ')
15.        printf("一个空格\n");
16.    else
17.        printf("非数字、英文字母和空格的其他字符\n");
18.    return 0;
19. }
```

请按任意一个字符键:c

输入的字符 c 是一个小写字母
Press any key to continue_

// 成立则表示c是一个数字

// 成立则表示c是一个大写字母

// 成立则表示c是一个小写字母

// 成立则表示c是一个空格

// 否则，表示c是其他字符。

应用if语句的注意事项之一

Ø **if**关键字之后的表达式可以是任意表达式！

└ 通常是逻辑表达式或关系表达式（稍后介绍）

└ 但其他表达式也是合法的（通常是误写导致！）

┌ 比如，

```
if (x=100)
    printf("%d",x);
```

└ 也可以是一个变量（或常量）

┌ 比如

```
int y=100;
if (y)
    printf("%d",y);
```

```
int y=0;
if (x=y)
    printf("%d",x);
else
    printf("x=0");
```


应用if语句的注意事项之二

Ø 条件判断表达式**必须用括号包裹起来**，后面**绝对不能加分号**！

```
Uif x==y
```

```
    printf("x和y相等");
```

错

```
Uif (x==y);
```

```
    printf("x和y相等");
```

错

```
Uif (x==y)
```

```
    printf("x和y相等");
```

对

应用if语句的注意事项之三

Ø if结构中的语句应为单个语句。如果想在满足条件时执行一组（多条）语句，则必须把这一组语句用“{ }”括起来组成一个复合语句。但是在“}”之后不能再加分号。

错

```
.....  
if (a>b)  
    t=a;  
    a=b;  
    b=t;  
else  
    .....
```

对

```
.....  
if (a>b)  
{  
    t=a;  
    a=b;  
    b=t;  
}  
else  
    .....
```

强烈推荐

```
.....  
if (x>=0)  
{  
    printf("x是一个非负数");  
}  
else  
{  
    printf("x是一个负数");  
}  
.....
```

强烈推荐：if语句的每个部分都用“{ }”括起来，不管是单条简单语句还是多条语句。

应用if语句的注意事项之四

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int n;
6.
7.     scanf("%d",&n);
8.
9.     if (n >= 0)
10.    {
11.        printf("n不小于0\n");
12.    }
13.    else
14.    {
15.        printf("n一开始小于0\n");
16.        n = 0;
17.        printf("n现在等于0\n");
18.    }
19.
20.    return 0;
21. }
```

Ø if语句正确的编程风格

1. if关键字和条件表达式
括号之间留一个空格
2. 大括号单独成行
3. 成对的大括号垂直对齐
4. 复合语句内的语句相对于
大括号要缩进
5. 复合语句内同层次的语
句要对齐

关系运算符

Ø 条件判断表达式中常见关系运算

Ø **关系运算符**或称为“比较运算符”，用来比较两个值，以判断其比较结果是否符合给定的条件。

u C语言提供6个关系运算符：

l 4个比较两个值大小关系： $>$ 、 $<$ 、 $>=$ 、 $<=$

l 2个判断相等关系： $==$ 和 $!=$

l 前四个优先级同，后两个同，前四个比后两个高；

l 关系运算符优先级低于算术运算符，但高于赋值运算符。如 $c > a + b$ 等价于 $c > (a + b)$ ， $a = b > c$ 等价于 $a = (b > c)$ 。

=与==的区别

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int x = 5;
6.
7.     if (x==8)
8.     {
9.         y = 2*x-3;
10.    }
11.    else
12.    {
13.        y = x*x+1;
14.    }
15.    printf("y=%d\n",y);
16.
17.    return 0;
18.}
```

y=26

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int x = 5;
6.
7.     if (x=8)
8.     {
9.         y = 2*x-3;
10.    }
11.    else
12.    {
13.        y = x*x+1;
14.    }
15.    printf("y=%d\n",y);
16.
17.    return 0;
18.}
```

y=13

关系表达式

Ø 定义：由关系运算符将两个数值表达式连接起来的表达式。

└如， $x > y$ 、 $(x = 5) \leq y$ 、 $x == y$

Ø 计算：如果两个数值表达式之间的关系满足给定的条件，则关系表达式结果为真（非0，一般是1）；不符合，则表达式结果为假（0）。

└例如， $a = 3$ ， $b = 4$ ， $c = 5$ ，则：

└关系表达式 $a + b > 2 * c$ 的值为0（假）

└关系表达式 $'b' != 'B'$ 的值为1（真）

使用关系运算符的注意事项

Ø C语言中的关系表达式和数学上的不同

u 比如： $0 \leq x \leq 3$ ，并不是验证x是否处于0和3之间。

而是从左到右，先计算 $0 \leq x$ ，值只能是0或1，再判断 $0 \leq 3$ 或 $1 \leq 3$ ，所以表达式的结果永远都是1。

Ø 如果两个操作数都是数值型，则按其大小比较；

Ø 如果两个操作数都是字符型，则按字符的ASCII码值比较大小；

Ø 如果两个操作数类型不同，则按算术转换规则进行类型转换。

逻辑运算符与逻辑表达式

Ø 逻辑运算符：

⊃ 逻辑与 **&&** (AND)、逻辑或 **||** (OR)、逻辑非 **!** (NOT)

Ø **逻辑表达式**：由逻辑运算符将关系表达式或逻辑量连接起来的式子，只产生真或假两种结果。

⊃ 例如： $x < y \&\& y < z$ 、 $x || y$ 、 $x == y$ 、 $!(x == y)$

⊃ 数学式子 $0 \leq x \leq 3$ 应表示为 $x \geq 0 \&\& x \leq 3$

Ø **优先级**：“!” 优先级高于 “&&”，“&&” 又高于 “||”，即优先级顺序为非、与、或。

⊃ 难记吗？谐音记法：Not at all (NOT AND OR)

逻辑表达式的值

a	b	a&&b	a b	!a	!b
0	0	0	0	1	1
0	非0	0	1	1	0
非0	0	0	1	0	1
非0	非0	1	1	0	0

Ø 思考：

⊃ 如果a=0，b=5，则!a=?，!b=?，a&&b=?，a || b=?

⊃ 如果a=3，b=8呢？

Ø “&&” 和 “||” 的优先级低于关系运算符

⊃ 如a<b&&c==10 || a!=c等价于((a<b)&&(c==10)) || (a!=c)

⊃ 那么，表达式x=!a<b&&a>!b+c || c<0的运算次序是什么？

&&与||的“短路”运算

Ø 在计算含有&&和||运算符的逻辑表达式时，C语言规定：**如果逻辑表达式的值可以由左侧运算对象单独推导出来，则不会再计算右侧运算对象的值。**

⌌ 表达式1&&表达式2，若计算出表达式1的值为0，就不会对表达式2求值

⌌ 表达式1||表达式2，若计算出表达式1的值不为0，就不会对表达式2求值

```
1. #include <stdio.h>
2. void main( )
3. {
4.     int x,y;
5.     x = 2;
6.     printf("%d ", 0&&(x=6));
7.     printf("x=%d ", x);
8.     printf("%d ", 'A'&&(x=6));
9.     printf("x=%d ", x);
10.    y = 2;
11.    printf("%d ", '0' || y=8);
12.    printf("y=%d ", y);
13.    printf("%d ", 0 || (y=8));
14.    printf("y=%d\n", y);
15. }
```

0□x=2□1□x=6□1□y=2□1□y=8

作业 2016/10/9

Ø 按下列要求编写程序，提交手写源代码（下次课10月11日课间交）

1. 输入3个整数，输出它们的平均值，保留3位小数
2. 输入正整数 n ($n < 360$)，输出 n 度的正弦、余弦函数值。提示：使用教材附录F中的数学函数。
3. 输入1个整数，若能被7整除，输出商的结果（要求使用单分支if...语句结构）
4. 输入2个整数，判断其和是奇数还是偶数，输出判断结果（要求使用if...else...语句结构）
5. 输入3个浮点数，找出其中最大的数，输出该数（要求使用if...else if...语句结构）

Ø 上机练习（不用交）：编译运行本讲义例程