西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

# C程序设计

# Programming in C

**1011014**

主讲：姜学锋，计算机学院

# 编程实现选择分支（1）

1、程序顺序结构

2、程序选择结构

## 3.3 程序顺序结构

- 通常情况下，语句以其出现的顺序执行

- 一个语句执行完会自动转到下一个语句开始执行，这样的执行称为顺序执行。

## 3.3.1 顺序执行

- 顺序执行的次序是很重要

图3.1 求圆面积的执行次序

## 3.3.1 顺序执行

- 顺序执行有一种特殊的情形就是函数执行。程序执行到函数时，会暂停当前的执行流程，进入到函数中开始一段新的执行流程，从函数返回后再继续当前的执行流程。

## 3.3.2 跳转执行

- 从问题求解的一般过程来看，还需要跳转执行。

- ①选择语句：if语句、switch语句；
- ②循环语句：while语句、do语句、for语句；
- ③跳转语句：goto语句、break语句、continue语句、return语句。

## 3.3.2 跳转执行

- goto语句的作用是使程序无条件跳转到别的位置，语法形式为：

```
goto 标号;
```

- 这里的标号是一个自定义的标识符，标号语句形式为：

```
标号: 语句序列.....;
```

- 当程序执行到goto语句时，就直接跳转到标号语句的位置继续运行。

## 3.3.2 跳转执行

- 例如：

```
 1 goto L1;
   ...... //语句序列

10 L1: x=a+b;
       ...... //语句序列
```

## 3.3.2 跳转执行

- C语言规定，goto语句只能在函数内部跳转，不能跳转到别的函数中。

- 标号语句的标号，一般用大写，后跟冒号（：），后面可以是任意形式的C语句。如果要跳转到的位置是语句块的结束，右大括号（}）的前面，需要在标号后面使用空语句。

## 3.3.2 跳转执行

- 由于goto语句无条件的跳转破坏了程序的结构化，导致可阅读性降低，所以少用或不用goto语句是编程的好习惯。

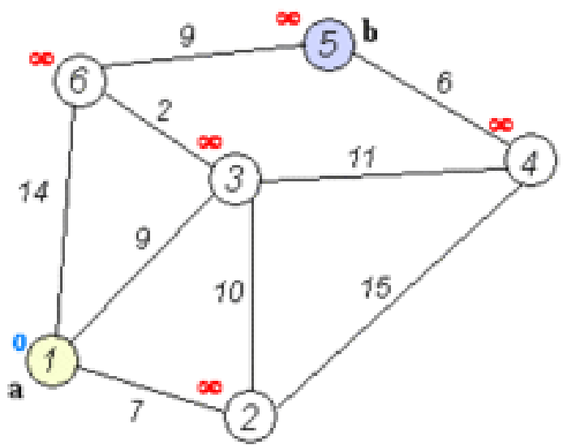

Go To Statement Considered Harmful

Edsger W. Dijkstra, Reprinted from Communications of the ACM, Vol. 11, No. 3, March 1968, pp. 147-148.

## 3.3.2 跳转执行

- Edsger W. Dijkstra

Dijkstra's algorithm



## 3.4 程序选择结构

- 1. 单分支选择：if语句
- 2. 多分支选择：switch语句

## 3.4.1 if语句

- if语句的作用是计算给定的表达式，根据结果选择执行相应的语句，语句形式有两种：
- ①if形式：

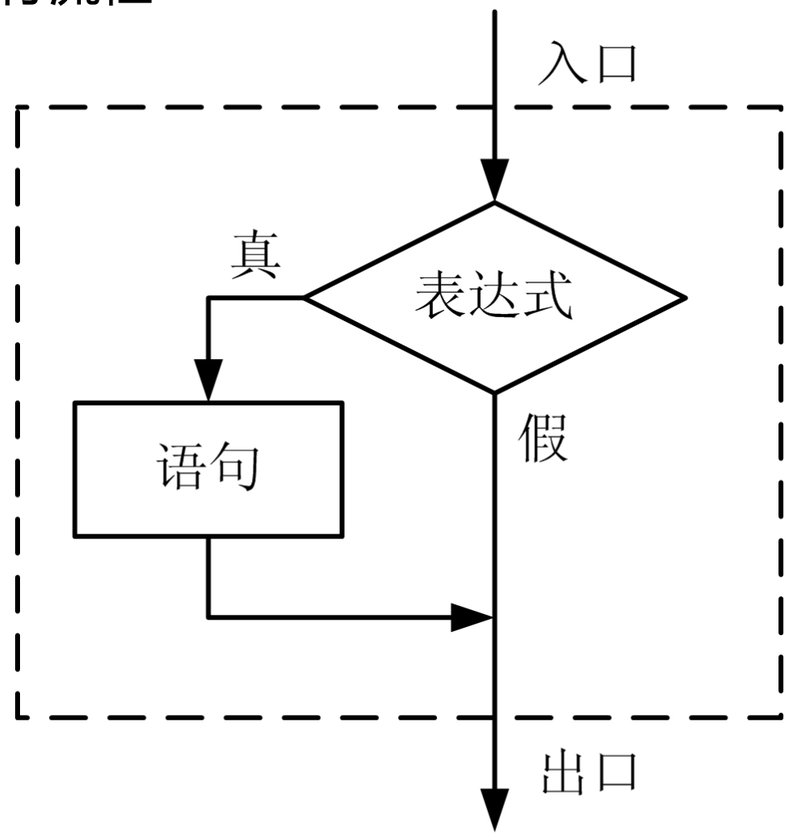
```
if (表达式) 语句1;
```

- ②if-else形式：

```
if (表达式) 语句1; else 语句2;
```

## 3.4.1 if语句

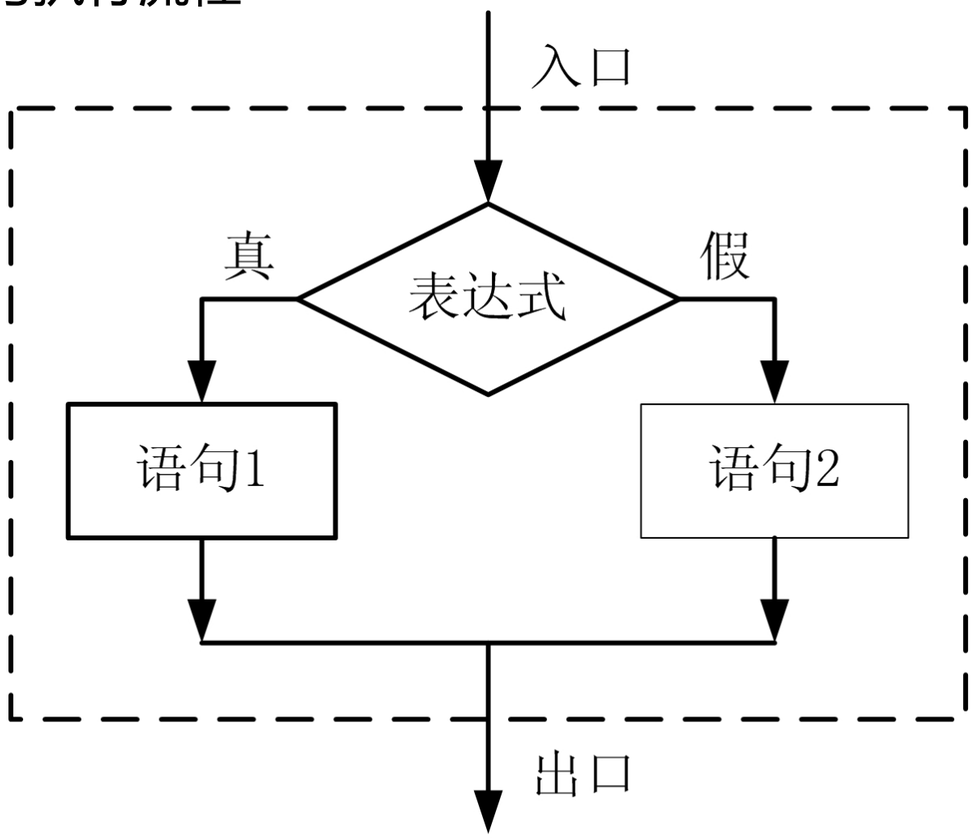
图3.2 两种形式的if语句执行流程




（a）

## 3.4.1 if语句

图3.2 两种形式的if语句执行流程




（b）

## 3.4.1 if语句

### 用程序实现自己的“意图”

当a > b时，执行t=a,a=b,b=t运算，即a和b相互交换，若a小于等于b则什么也不做；总而言之，无论a和b之前是什么数，执行这段程序后，a肯定小于等于b。

```
if ( a>b ) t=a, a=b, b=t;
```

## 3.4.1 if语句

- （1）if语句中的子语句既可以是简单语句，又可以是复合语句或控制语句，但必须是“一个语句”的语法形式。例如：

```
1 if (a>b)
2    x=a+b; y=a-b;
3 else
4    x=a-b; y=a+b;
```

- 第2行语法错误，因为if分支子语句是两个语句的形式，不符合语法要求；第4行虽然没有语法错误，但有写法歧义性问题，即y=a+b;这个语句并不是else分支子语句，而是这个if语句的后续语句。

## 3.4.1 if语句

- （2）子语句往往会有多条语句，甚至更复杂的情形，这时可以使用复合语句。
- 示例

```
1 if (a>b) {
2   x=a+b; y=a-b;
3 }
4 else {
5   x=a-b; y=a+b;
6 }
```

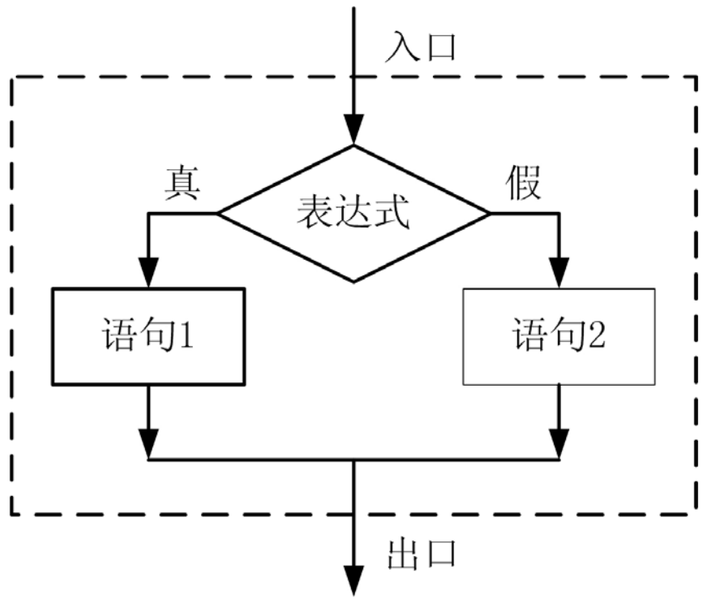
## 3.4.1 if语句

- 总是交换的代码（不好的code风格）：

```
1 if ( a>b ) ;
2 {
3   t=a;
4   a=b;
5   b=t;
6 }
```

```
1 if ( a>b ) {  ;
2   t=a;
3   a=b;
4   b=t;
5 }
```

- （3）if语句都有一个共同的入口和出口，执行流程的不同依赖子语句的不同，这种形式就是程序选择结构。




（b）

## 3.4.1 if语句

- （4）if语句后面的圆括号是语法规定必须有的，表达式可以是C语言的任意表达式；但由于其结果是按逻辑值来处理的，通常情况下，选择条件是关系表达式或逻辑表达式，应该谨慎出现别的表达式。

```
if ( 条件表达式或逻辑表达式 ) 语句1; else 语句2;
```

## 3.4.1 if语句

- 情形一，选择条件是赋值表达式，例如：

```
a=5, b=2;
if ( a = b ) x=a*10;
```

## 3.4.1 if语句

- 表达式是赋值可能还有一个原因就是误将相等比较写成了赋值，即（==）少写了一个等号变成了（=）。例如：

```
a=5, b=2;
if ( a == b ) x=a*10;
```

## 3.4.1 if语句

- 情形二，选择条件是数值、指针值或算术运算，例如：

```
a=5,b=2;
if ( a ) x=a*10;
```

  - 这时表达式直接将a按逻辑值来处理，故选择条件为真，执行子语句。

## 3.4.1 if语句

表3-8 数值按逻辑值处理的结果

| 数值 | 逻辑值 | | | |
|---|---|---|---|---|
| a | a | a!=0 | !a | a==0 |
| 0 | 假 | 假 | 真 | 真 |
| 非0 | 真 | 真 | 假 | 假 |

写法“a”和“a!=0”是等效的，“!a”和“a==0”是等效的。

## 3.4.1 if语句

- 情形三，选择条件是常量，例如：

```
if ( 0 ) {
   x=a*10;
   ......
}
```

- 这时表达式恒为假，子语句永远也不可能得到执行。这种极端写法通常用来调试，即通过安排表达式为假来“屏蔽”尚未完工的子语句。

## 3.4.1 if语句

- （5）if-else语句和条件运算符似乎很像。例如：

```
if (c1>='A' && c1<='Z') c=c1+32;
else c=c1;
```

- 和

```
c = (c1>='A' && c1<='Z') ? c1+32 : c1;
```

## 3.4.1 if语句

- if-else语句是语句，它可以包含任意多的表达式，或任意多的语句组合。而条件运算符则能力有限，它仅局限于表达式。因此if-else语句可以替代条件运算符，反之则不成立。


程序设计