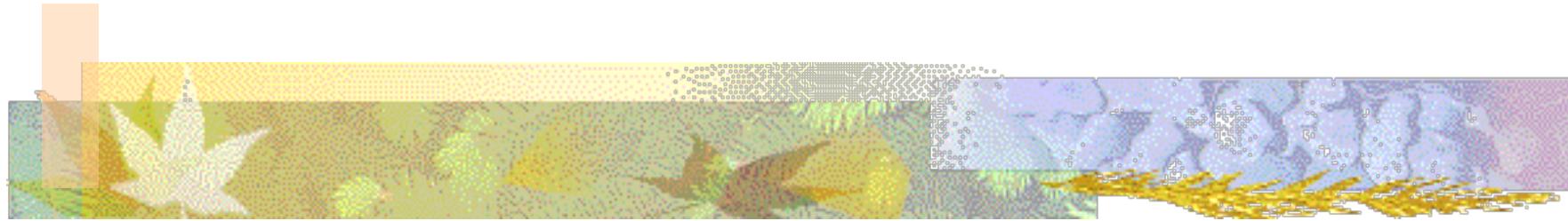




规格严格 功夫到家



# 第5章 选择控制结构



哈尔滨工业大学

计算机科学与技术学院

苏小红

[sxh@hit.edu.cn](mailto:sxh@hit.edu.cn)

# 本章学习内容

- ❖ 算法的描述方法
- ❖ 用于单分支控制的**if**语句
- ❖ 用于双分支控制的**if-else**语句
- ❖ 用于多路选择的**switch**语句
- ❖ **break**语句在**switch**语句中的作用
- ❖ 关系运算符
- ❖ 条件运算符
- ❖ 逻辑运算符
- ❖ 程序测试

# 5.1 生活中与计算机中的问题求解 (*Problem Solving Process*)

生活中的问题求解：

**Problem:** 烤蛋糕 (Baking a Cake)

**How to solve:**

1. Start
2. 将烤箱预热
3. 准备一个盘子
4. 在盘子上抹上一些黄油
5. 将面粉、鸡蛋、糖和香精混合在一起搅拌均匀
6. 将搅拌好的面粉团放在盘子上
7. 将盘子放到烤箱内
8. End



# 分治策略 ("Divide and Conquer" Strategy )

**Problem:** 准备早餐 ( Prepare a Breakfast)

1. Start
2. 准备早餐
3. End



# 分治策略 ("Divide and Conquer" Strategy )

1. Start

2. 准备早餐

    2.1 准备一个金枪鱼三明治

    2.2 准备一些薯条

    2.3 冲一杯咖啡

3. End



# 分治策略 ("Divide and Conquer" Strategy )

1. Start

2. 准备早餐

    2.1 准备一个金枪鱼三明治

        2.1.1 拿来两片面包

        2.1.2 准备一些金枪鱼酱

    2.2 准备一些薯片

    2.3 冲一杯咖啡

3. End



# 分治策略 ("Divide and Conquer" Strategy )

1. Start

2. 准备早餐

    2.1 准备一个金枪鱼三明治

        2.1.1 拿来两片面包

        2.1.2 准备一些金枪鱼酱

    2.2 准备一些薯片

        2.2.1 将土豆切成片

        2.2.2 油炸这些土豆片

    2.3 冲一杯咖啡

3. End

# 分治策略

## ( "*Divide and Conquer*" Strategy )

1. Start

2. 准备早餐

    2.1 准备一个金枪鱼三明治

        2.1.1 拿来两片面包

        2.1.2 准备一些金枪鱼酱

    2.2 准备一些薯片

        2.2.1 将土豆切成片

        2.2.2 油炸这些土豆片

    2.3 冲一杯咖啡

        2.3.1 烧些开水放入杯中

        2.3.2 在水杯中加入一些咖啡和糖

2024/3/20 3. End

8/56

## 5.2 算法的概念及其描述方法

- 面向对象程序 = 对象 + 消息
- 面向过程的程序 = 数据结构 + 算法
- 计算机中的算法（Algorithm）
  - 为解决一个具体问题而采取的、确定的、有限的操作步骤，仅指计算机能执行的算法
  - A specific and step-by-step set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the procedure terminate at some point

# 5.2 算法的概念及其描述方法

## ■ 算法的特性

- 有穷性
  - 在合理的时间内完成
- 确定性，无歧义
  - 如果 $x \geq 0$ ，则输出Yes；如果 $x \leq 0$ ，则输出No
- 有效性
  - 能有效执行
    - 负数开平方
  - 没有输入或有多个输入
  - 有一个或多个输出

# 5.2 算法的概念及其描述方法

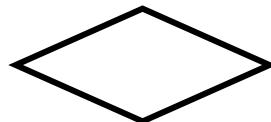
## ■ 算法的描述方法

- 自然语言描述
- 传统流程图 (Flowchart)
  - 在1966年, Bohra 与 Jacopini 提出
- N-S结构化流程图
  - 1973年, 美国学者I.Nassi 和 B.Shneiderman 提出
- 伪码 (Pseudocode) 表示

# 流程图 (*Flowchart*)

■ Flowchart represents algorithm graphically.

## Symbol



## Semantic

Start/End

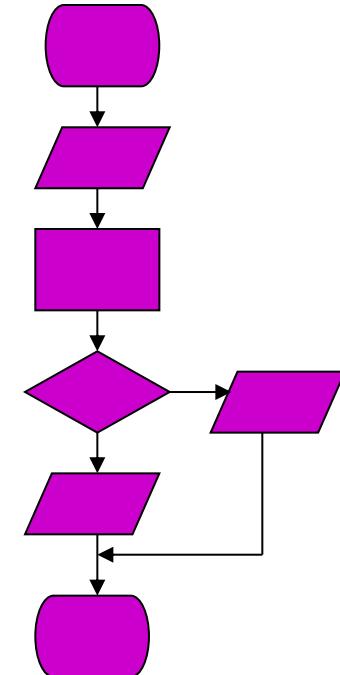
Process

Input/Output

Test

Connector

Flow of activities



# 计算机中的问题求解过程

Example : 买苹果，计算价钱

Calculate and display the price of a number of apples  
if the quantity in kg and price per kg are given.



- quantity
  - pricePerkg
- $\text{price} = \text{quantity} * \text{pricePerkg}$
- price

First identify the input and output of the problem.

# 顺序结构 (*Sequence Structure*)

## ■ 给变量赋值

- 赋值表达式语句

    赋值表达式；

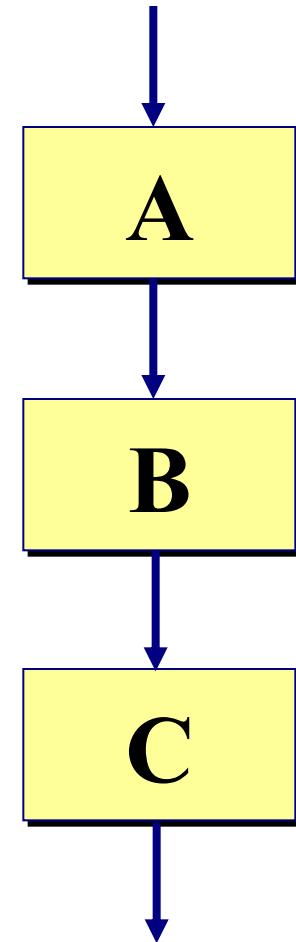
```
    price = quantity*pricePerkg;
```

## ■ 输入输出数据

- 标准库函数调用语句

```
    scanf("%d", &pricePerkg);
```

```
    printf("%d", price);
```



# 【例5.1】计算两整数的最大值



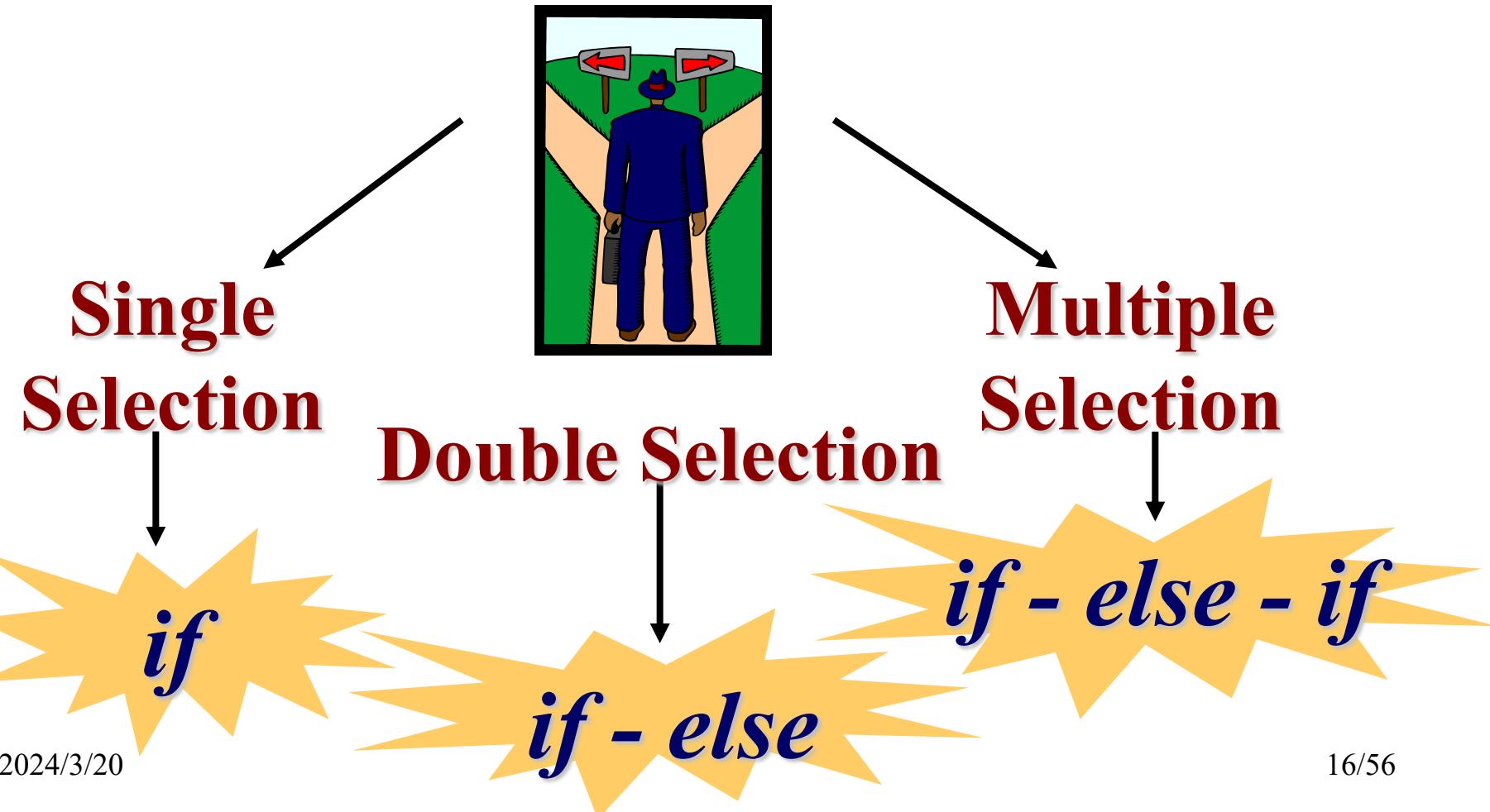
- num1
- num2

????

max



# 选择结构（分支结构） (Selection Structure)



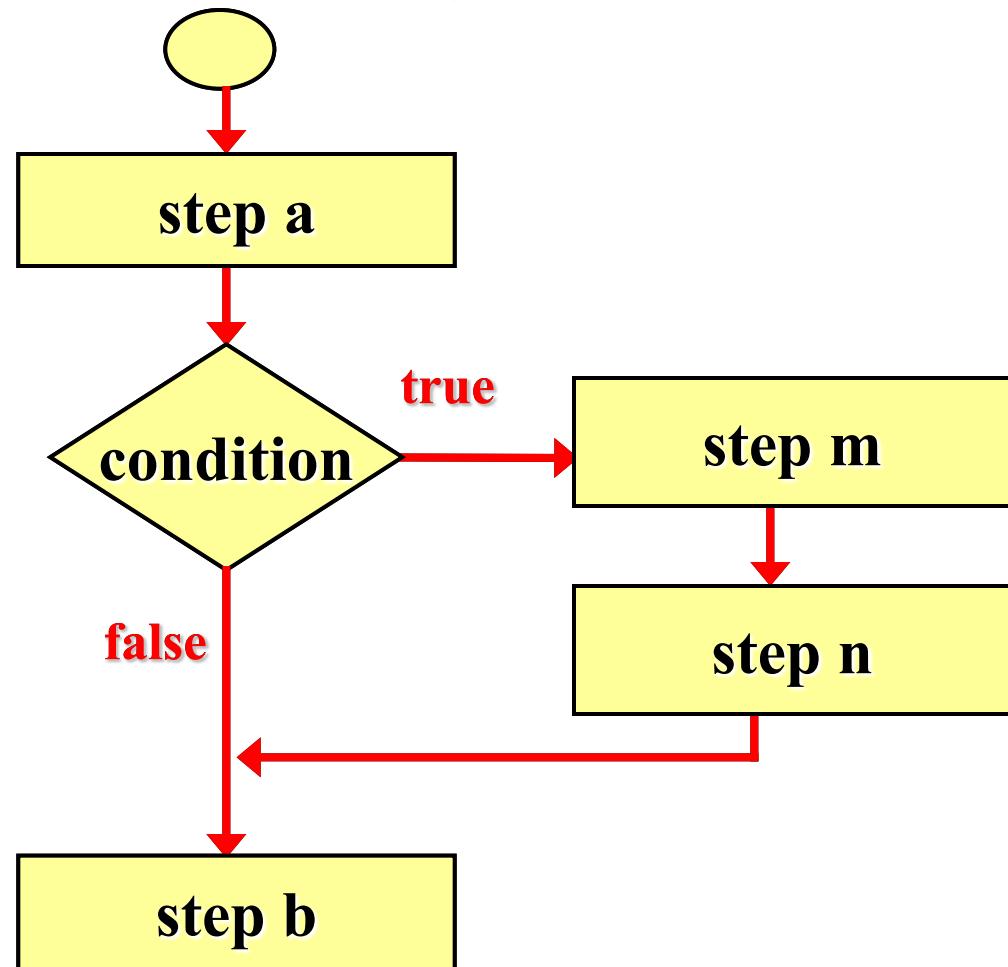
# 5.3 关系运算符与关系表达式

Relational Operation	Description	Examples of Expression	Value
<	Less than	$6 < 9$	1 (true)
$\leq$	Less than or equal to	$5 \leq 5$	1 (true)
>	Greater than	$2 > 6$	0 (false)
$\geq$	Greater than or equal to	$9 \geq 5$	1 (true)
$\equiv$	Equal to	$7 \equiv 5$	0 (false)
$\neq$	Not equal to	$6 \neq 5$	1 (true)

## 5.4 用于单分支控制的条件语句 (Single Selection)

### Pseudocode Structure

```
step a
if <condition is true>
start
    step m
    step n
end_if
step b
```



# *if Statement*

The structure is 表达式非0为真 Selection (flowchart)

Syntax:

```
if (expression)  
    statement;
```

or

```
if (expression)  
{  
    statement1;  
    statement2;  
}
```

复合语句  
compound statement  
被当作一条语句看待

# *if Statement*

The structure is similar to single selection (flowchart)

Syntax:

```
if (expression)  
    statement;
```

**Don't forget the  
parentheses !!**

or

```
{  
if (expression)  
    statement1;  
    statement2;  
}
```

**Don't forget the  
braces !!**

# 【例5.1】计算两整数的最大值

```
#include <stdio.h>
main()
{
    int a, b, max;
    printf("Input a,b:");
    scanf("%d,%d", &a, &b);
    if (a > b)    max = a;
    if (a <= b)   max = b;
    printf("max = %d\n", max);
}
```

Input a,b: 20 15

max = 20

# 5.5 用于双分支控制的条件语句 (Double Selection)

## Pseudocode Structure

Step a

if <condition is true>

start

Step m

Step n

end if

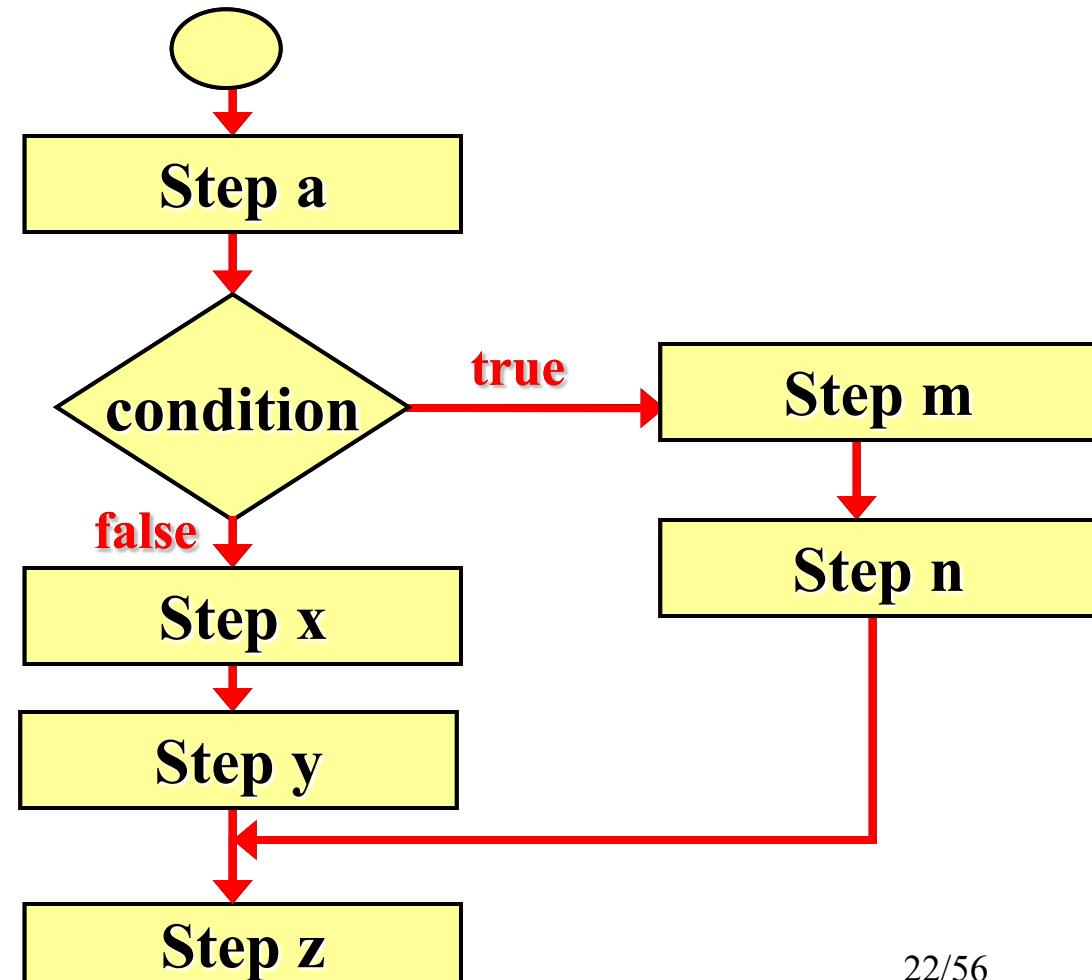
else start

Step x

Step y

end else

Step z



# ***if - else Statement***

The structure is similar to double selection (flowchart)

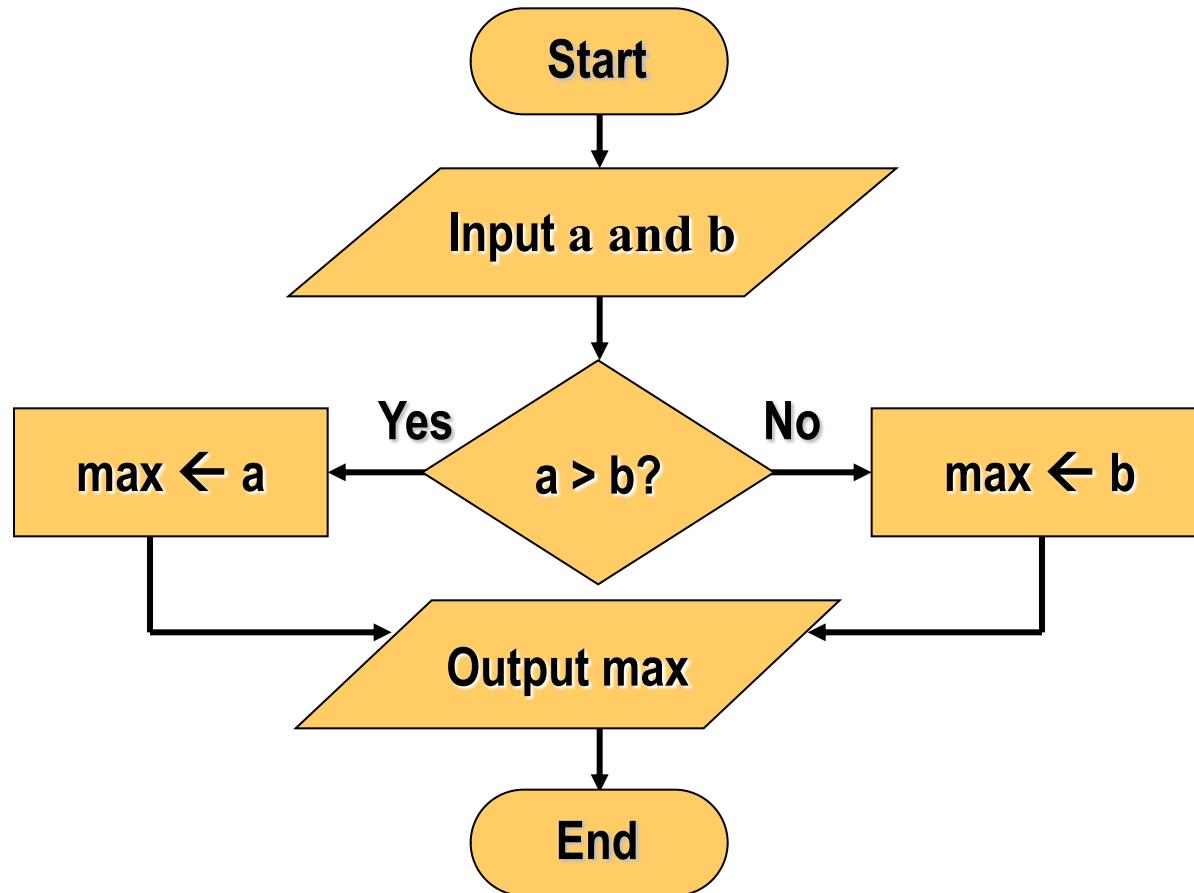
Syntax:

```
if (expression)  
    statement1;  
else  
    statement2;
```

or

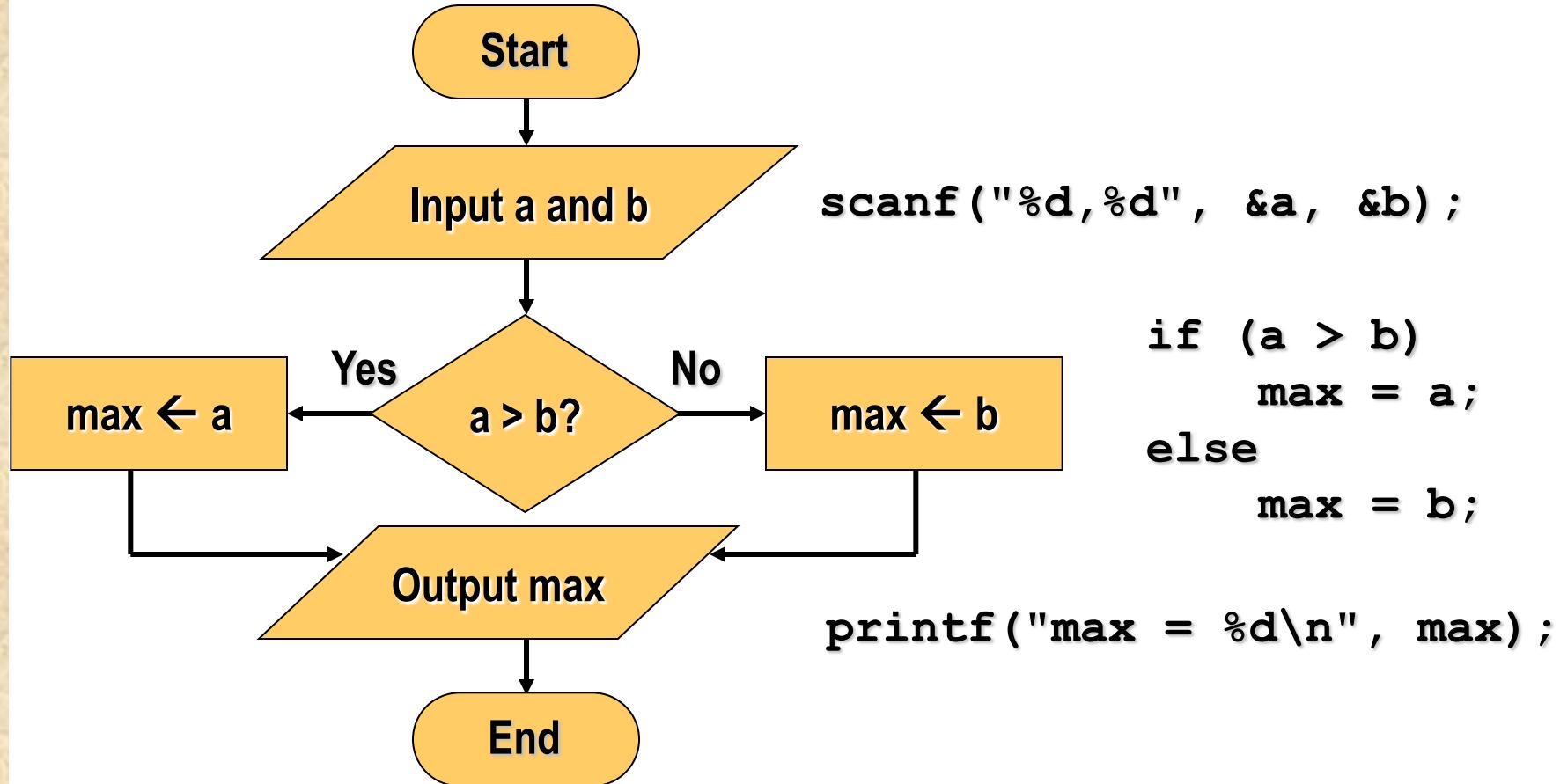
```
if (expression)  
{  
    statement1;  
    statement3;  
}  
else  
{  
    statement2;  
    statement4;  
}
```

# 【例5.2】计算两整数的最大值



**Flowchart: Calculate the Maximum**

# 【例5.2】计算两整数的最大值



*Turn Flowchart to C Program*

# 【例5.2】计算两整数的最大值

```
#include <stdio.h>
main()
{
    int a, b, max;

    printf("Input a, b:");
    scanf("%d,%d", &a, &b);

    if (a > b)
        max = a;
    else
        max = b;

    printf("max = %d", max);
}
```

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

```
if (a > b)
    max = a;
if (a <= b)
    max = b;
```

# 5.6 条件运算符和条件表达式

```
#include <stdio.h>
main()
{
    int a, b, max;
```

【例5.3】

表达式1 ? 表达式2 : 表达式3

```
printf("Input a, b:");
scanf("%d,%d", &a, &b);
```

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

```
max = a > b ? a : b;
```

```
printf("max = %d", max);
```

# 5.7 用于多分支控制的条件语句 (Multiple Selection)

## Multi-way if

Step a

**if** (expression1)

{

Step m

}

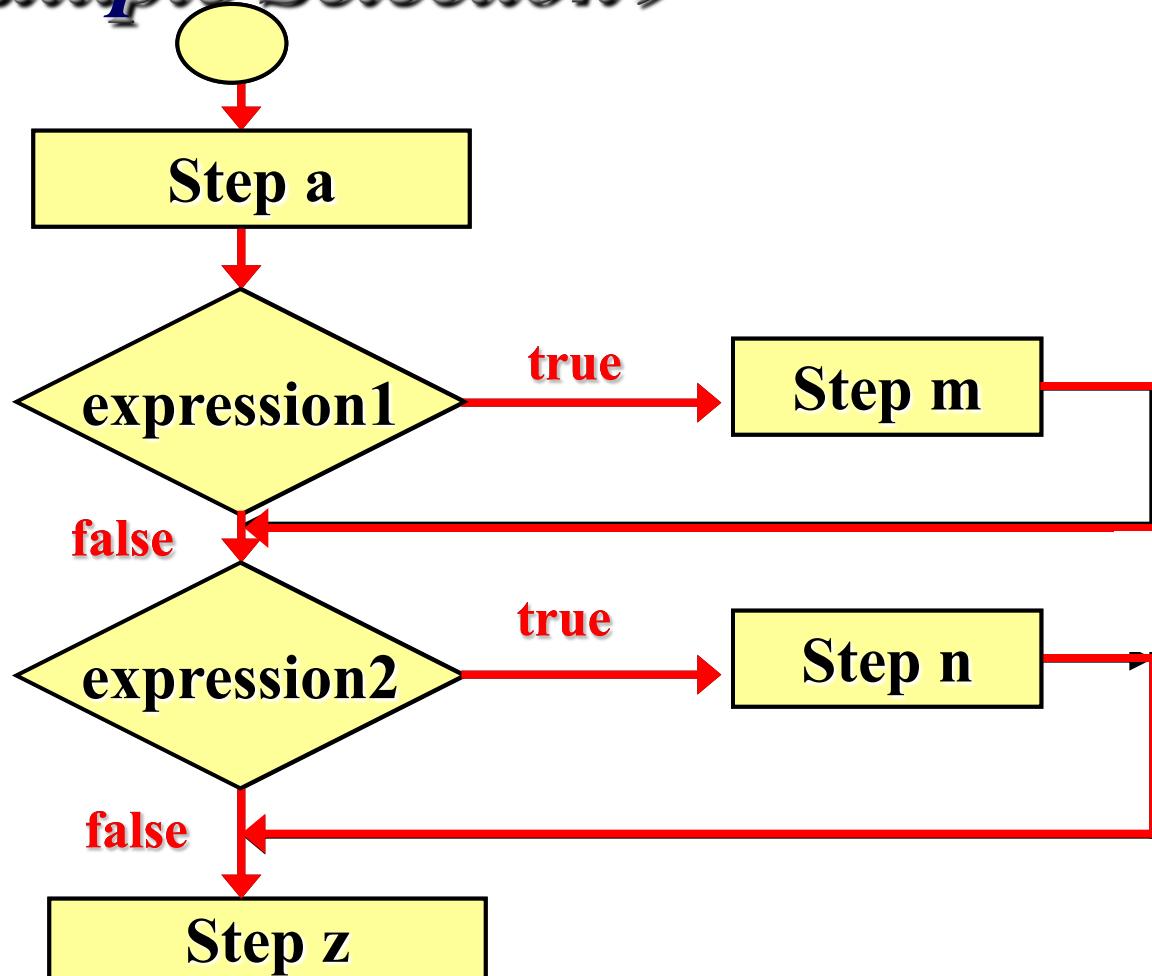
**if** (expression2)

{

Step n

}

Step z



## 5.7 用于多分支控制的条件语句 (Multiple Selection)

### Cascaded if

Step a

```
if (expression1)
```

```
{
```

```
    Step m
```

```
}
```

```
else if (expression2)
```

```
{
```

```
    Step n
```

```
}
```

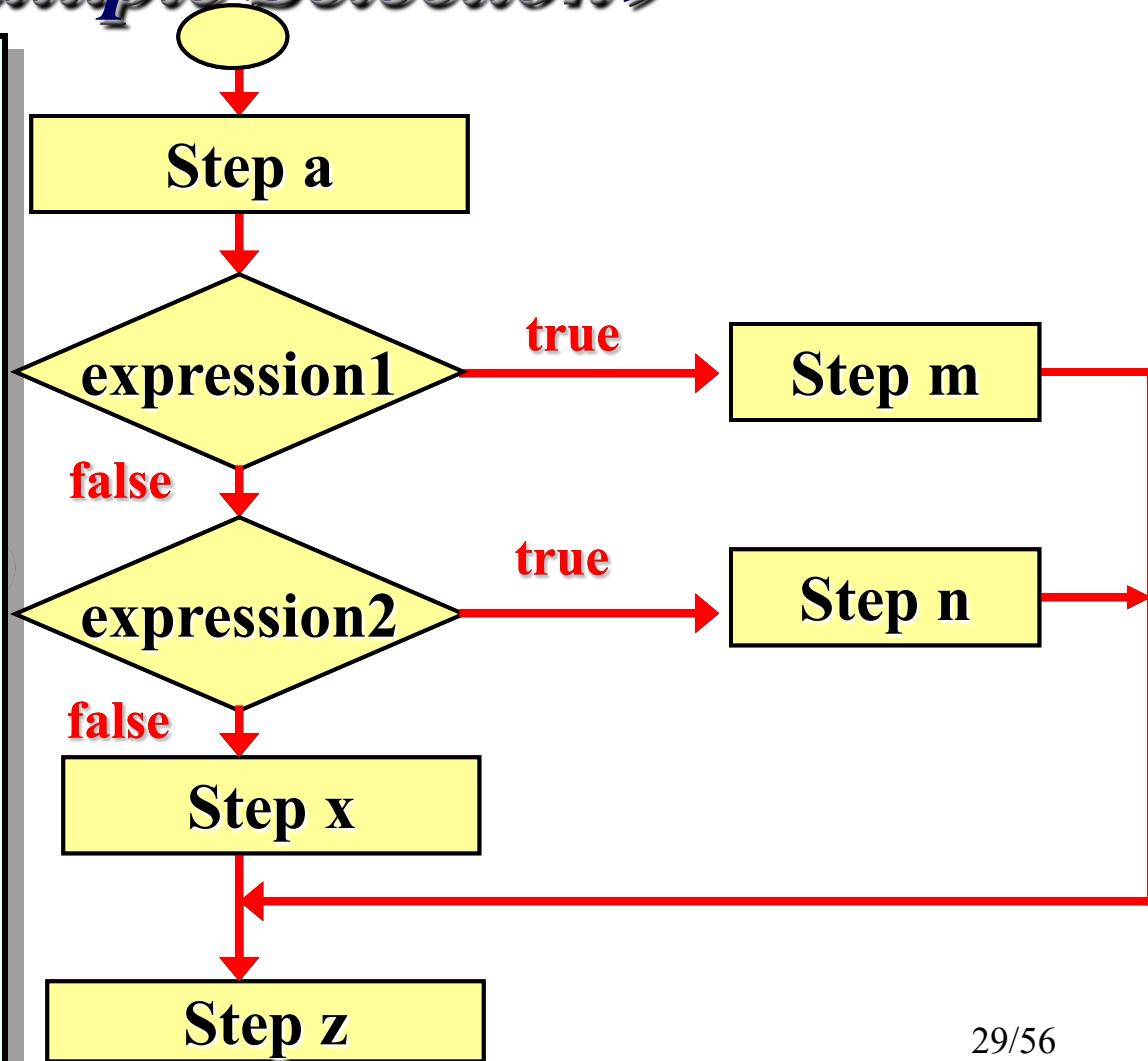
```
else
```

```
{
```

```
    Step x
```

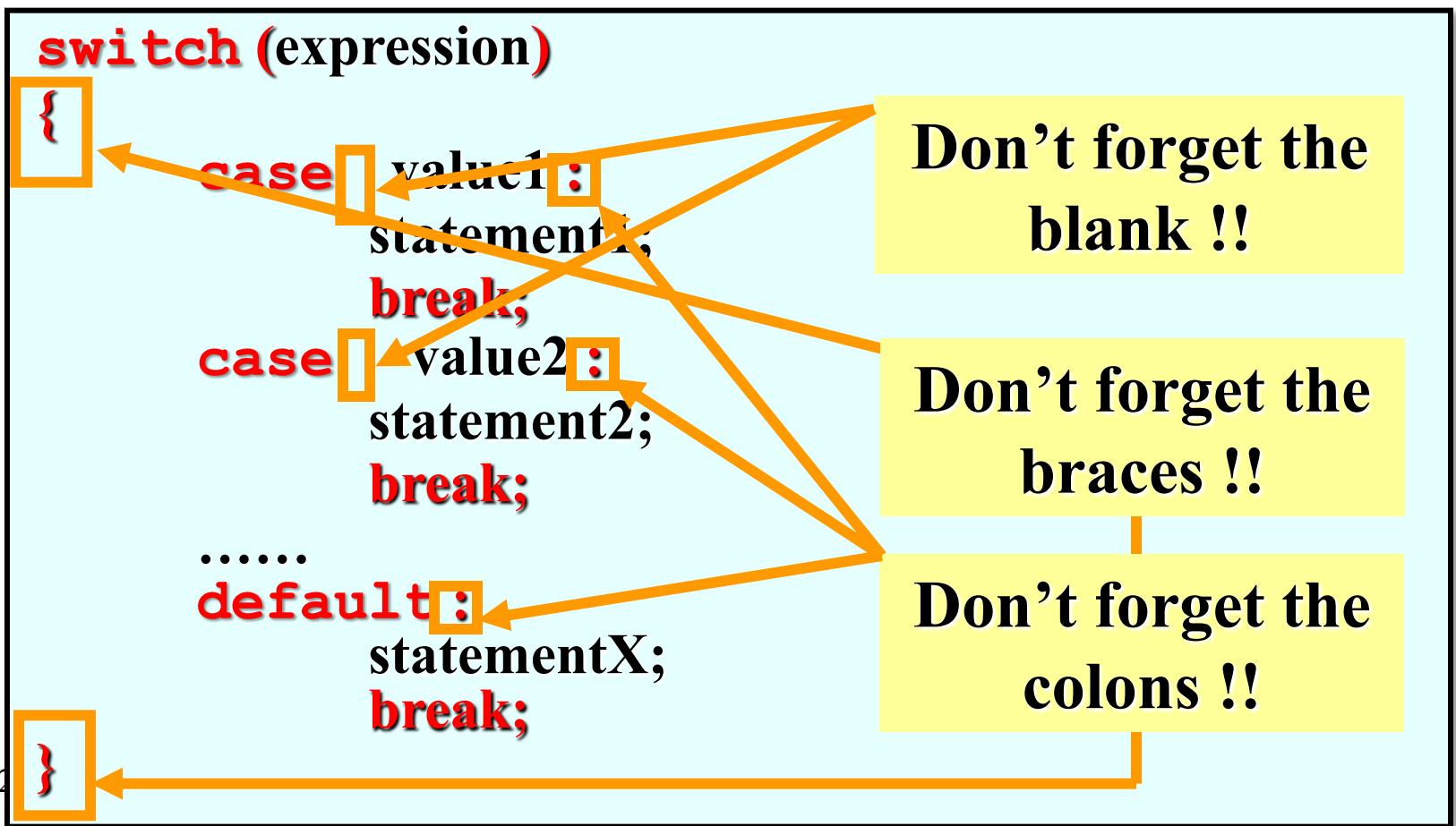
```
}
```

```
Step z
```



# 5.8 用于多路选择的switch语句

The structure is similar to multiple selection (flowchart)



# 5.8 用于多路选择的switch语句

Important Rule !

```
switch(expression)
{
    case value1 :
        statement1;
        break;
    case value2 :
        statement2;
        break;
    .....
    default:
        statementX;
        break;
}
```

Must be  
int or char!



## 5.8 用于多路选择的switch语句

```
Example: switch (month) {  
    case 1:  
        printf("Janua...");  
        break;  
    case 2:  
        printf("Februa...");  
        break;  
    case 3:  
        printf("March...");  
        break;  
    default:  
        printf("Others\n");  
        break;  
}  
printf("End");
```

Assume month = 1,

so ...

this step will be  
...case is terminated  
here. Jump to ...

January  
End \_

## 5.8 用于多路选择的switch语句

Example: switch (month) {

case 1:

```
    printf("January\n");  
    break;
```

case 2:

```
    printf("February\n");  
    break;
```

case 3:

```
    printf("Marc  
    break;
```

default:

```
    printf("Others  
    break;
```

}

```
printf("End");
```

March

End -

Assume month = 3,  
so ...

... this stem will be  
... case is terminated  
here. Jump to ...

## 5.8 用于多路选择的switch语句

```
Example: switch (month) {  
    case 1:  
        printf("January\n");  
        break;  
    case 2:  
        printf("February\n");  
        break; // This line is highlighted with an orange box and has an arrow pointing to it.  
    case 3:  
        printf("March\n");  
        break;  
    default:  
        printf("Others\n");  
        break;  
}  
printf("End");
```

Now...what will  
happen if this *break*  
is taken out from the  
program?

## 5.8 用于多路选择的switch语句

```
Example: switch (month) {  
    case 1:  
        printf("January\n");  
        break;  
    case 2:  
        printf("February\n");  
    case 3:  
        printf("March\n");  
        break;  
    default:  
        printf("Others\n");  
        break;  
}  
printf("End");
```

No more !

## 5.8 用于多路选择的switch语句

Example: switch (month)

case 1:

printf("January\n");  
break;

case 2:

printf("February\n");

case 3:

printf("March  
break,

default.

Assume month = 2,  
so ...

End \_

...this step will be  
executed. Later ...

...execution continues. Thus, ...case is terminated  
this step is executed . So ... here. Jump to ...

## 5.8 用于多路选择的switch语句

```
Example: switch (month) {  
    case 1:  
        printf("January\n");  
        break;  
    case 2:  
        printf("February\n");  
    case 3:  
        printf("March\n");  
        break;  
    default:  
        printf("Others\n");  
        break;  
}  
printf("End");
```

And ...  
if month = 34 ?

Now...what will  
happen if these  
breaks are taken out  
from the program?

最好不省略!

# 【例5.5】计算器程序

- 编程设计一个简单的计算器程序，要求用户从键盘输入如下形式的表达式：

操作数1 运算符op 操作数2

然后，计算并输出表达式的值  
指定的运算符为

加 (+)

减 (-)

乘 (\*)

除 (/)



# 【例5.5】

```
main()
{
    int      data1, data2;          /*定义两个操作符*/
    char     op;                  /*定义运算符*/

    printf("Please enter the expression:");
    scanf("%d%c%d", &data1, &op, &data2); /*输入运算表达式*/

    switch (op)
    {
        case '+':                /*处理加法*/
            printf("%d + %d = %d\n", data1, data2, data1 + data2);
            break;
        case '-':                /*处理减法*/
            printf("%d - %d = %d\n", data1, data2, data1 - data2);
            break;
        case '*':                /*处理乘法*/
            printf("%d * %d = %d\n", data1, data2, data1 * data2);
            break;
        case '/':                /*处理除法*/
            if (0 == data2)
                printf("Division by zero!\n");
            else
                printf("%d/%d = %d\n", data1, data2, data1/data2);
            break;
        default:
            printf("Invalid operator! \n");
    }
}
```

注释掉会怎样?

Why?

# 思考题

- 语句 `if (0==data2)` 的必要性——避免"除零错误"
  - 1998年11月，《科学美国人》杂志描述了美国导弹巡洋舰约克敦号上的一起事故，除零错导致军舰推进系统的关闭
- 为什么不用 `if (data2 == 0)` ?
- 如果要求输入的算术表达式中的操作数和运算符之间可以加入任意多个空格符，那么程序如何修



# 【例5.5】

```
main()
{
    int     data1, data2;
    char    op;

    printf("Please enter the expression:");
    scanf("%d %c %d", &data1, &op, &data2); /* %c前有一个空格 */

    switch (op)
    {
        case '+':
            printf("%d + %d = %d\n", data1, data2, data1 + data2);
            break;
        case '-':
            printf("%d - %d = %d\n", data1, data2, data1 - data2);
            break;
        case '*':
            printf("%d * %d = %d\n", data1, data2, data1 * data2);
            break;
        case '/':
            if (0 == data2)
                printf("Division by zero!\n");
            else
                printf("%d/%d = %d\n", data1, data2, data1/data2);
            break;
        default:
            printf("Invalid operator! \n");
    }
}
```

# 思考题

- 如果要求对浮点数进行运算，那么程序如何修改？
- 修改例5.5程序，使其能进行浮点数的算术运算，同时允许使用字符\*、**x**与**X**作为乘号，并且允许输入的算术表达式中的操作数和运算符之间可以加入任意多个空格符。



# 【例5.6】

```
main()
{
    float data1, data2;
    char op;
    printf("Please enter the expression:");
    scanf("%f %c%f", &data1, &op, &data2);
    switch (op)
    {
        case '+':
            printf("%f + %f = %f\n", data1, data2, data1 + data2);
            break;
        case '-':
            printf("%f - %f = %f\n", data1, data2, data1 - data2);
            break;
        case '*':
        case 'x':
        case 'X':
            printf("%f * %f = %f\n", data1, data2, data1 * data2);
            break;
        case '/':
            if fabs(data2) <= 1e-7 /* 实数与0比较 */
                printf("Division by zero!\n");
            else
                printf("%f/%f = %f\n", data1, data2, data1/data2);
            break;
        default:
            printf("Invalid operator! \n");
    }
}
```

取绝对值函数

# 5.9 逻辑运算符和逻辑表达式 (Logical Operators)

## Symbol

**&&**

## Description

与 (AND) 当且仅当两者都为真，则结果为真

**||**

或 (OR) 只要两者中有一个为真，结果就为真

**!**

非 (NOT)

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

高 ! && || 低  
—————>

# 5.9 逻辑运算符和逻辑表达式 (Logical Operators)

■ ch是大写英文字母

`(ch >= 'A') && (ch <= 'Z')`

■ 判断某一年year是否是闰年的条件是满足下列二者之一

- 能被4整除，但不能被100整除；
- 能被400整除；

`year%4==0 && year%100!=0 || year%400==0`

优先级： % == ( != ) && ||

`((year%4==0) && (year%100!=0)) || (year%400==0)`

# 复合表达式 (Compound Expression) 的值

Example:

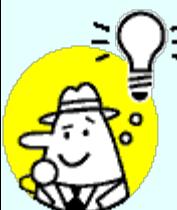
$(a \geq 1) \&\& (b++ == 5)$

$\rightarrow (0 \geq 1) \&\& (b++ == 5)$

$\rightarrow 0 \&\& (b++ == 5)$

$\rightarrow 0$

a	0
b	5
c	15
d	17



尽量使用最少的操作数来确定表达式的值，这就意味着表达式中的某些操作数可能不会被计算

# 5.10 本章扩充内容

## ■ 测试的主要方式

- 给定特定的输入，运行被测软件
- 检查软件的输出是否与预期结果一致

## ■ 测试用例的选取方法

- 尽量覆盖所有分支，减少重复覆盖

## ■ 测试的目的

- 通过运行测试用例找出软件中的Bug
- 成功的测试在于发现迄今为止尚未发现的Bug
- 测试人员的主要任务是站在使用者的角度，通过不断使用和攻击，尽可能多地找出Bug
- 测试的过程就像黑客的攻击过程，专门找软件漏洞

## 5.10 本章扩充内容

- 采用测试用例，通过运行程序查找程序错误的方法
  - 实质是一种抽样检查，彻底的测试是不可能的
  - 彻底的测试不现实，要考虑时间、费用等限制，不允许无休止的测试
- 测试只能证明程序有错，不能证明程序无错  
—— E.W.Dijkstra
- 测试能提高软件质量，但提高软件质量不能依赖于测试



# 软件测试方法的分类

## ■ 白盒测试（结构测试）

- 在完全了解程序的结构和处理过程的情况下，按照程序内部的逻辑测试程序，检验程序中的每条逻辑路径是否都能按预定要求正确工作
- 主要用于测试的早期

## ■ 黑盒测试（功能测试）

- 把系统看成一个黑盒子，不考虑程序内部的逻辑结构和处理过程，只根据需求规格说明书的要求，设计测试用例，检查程序的功能是否符合它的功能说明
- 主要用于测试的后期

通常结合使用选择有限数量的重要路径进行白盒测试，对重要的功能需求进行黑盒测试

# 【例5.7】判断三角形的类型

```
#include <stdio.h>
#include <math.h>
main()
{
    float a, b, c;

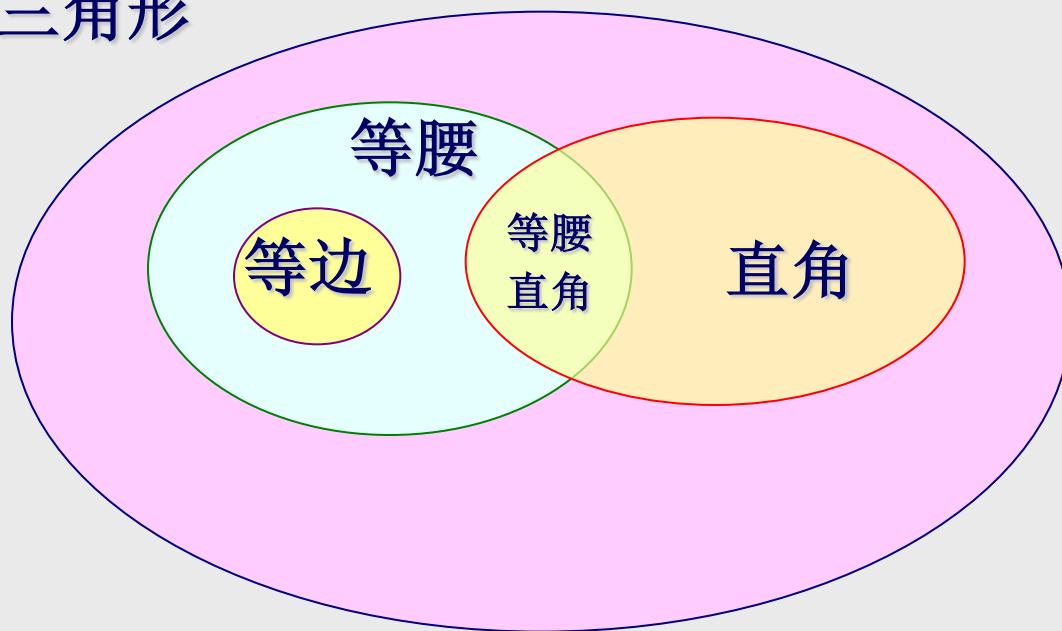
    printf("Input the three edge length:");
    scanf("%f, %f, %f", &a, &b, &c);

    if (a+b>c && b+c>a && a+c>b)          /*三角形的基本条件*/
    {
        if (a==b || b==c || c==a)
            printf("等腰三角形");
        else if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
            printf("直角三角形");
        else
            printf("一般三角形");
    }
    else
    {
        printf("不是三角形\n");
    }
}
```

错在哪里?

# 【例5.7】判断三角形的类型

不是三角形



有交叉关系的用并列的 `if`  
非此即彼的用 `if-else`



# 【例5.7】

```
main()
{
    float a, b, c;
    int flag = 1;
    .....
    if (a+b>c && b+c>a && a+c>b)      /*三角形的基本条件*/
    {
        if (a==b || b==c || c==a)
        {
            printf("等腰");
            flag = 0;
        }
        if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
        {
            printf("直角");
            flag = 0;
        }
        if (flag)
        {
            printf("一般");
        }
        printf("三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
}
```

```

main()
{
    .....
    if (a+b>c && b+c>a && a+c>b) /*三角形的基本条件*/
    {
        if (a==b && b==c && c==a)
        {
            printf("等边");
            flag = 0;
        }
        if (a==b || b==c || c==a)
        {
            printf("等腰");
            flag = 0;
        }

        if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
        {
            printf("直角");
            flag = 0;
        }
        if (flag)
            printf("一般");
        printf("三角形\n");
    }
    else
        printf("不是三角形\n");
}

```

## 【例5.7】

错在哪里?

```
main()
{
    .....
    if (a+b>c && b+c>a && a+c>b) /*三角形的基本条件*/
    {
        if (a==b && b==c && c==a)
        {
            printf("等边");
            flag = 0;
        }
        else if (a==b || b==c || c==a)
        {
            printf("等腰");
            flag = 0;
        }

        if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
        {
            printf("直角");
            flag = 0;
        }
        if (flag)
            printf("一般");
        printf("三角形\n");
    }
    else
        printf("不是三角形\n");
}
```

## 【例5.7】

```
main()
```

```
{
```

```
.....
```

```
if (a+b>c && b+c>a && a+c>b) /*三角形的基本条件*/
```

```
{
```

```
if (a==b || b==c || c==a)
```

```
{
```

```
printf("等腰");
```

```
flag = 0;
```

```
}
```

```
else if (a==b && b==c && c==a)
```

```
{
```

```
printf("等边");
```

```
flag = 0;
```

```
}
```

```
if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
```

```
{
```

```
printf("直角");
```

```
flag = 0;
```

```
}
```

```
if (flag)
```

```
printf("一般");
```

```
printf("三角形\n");
```

```
}
```

```
else
```

```
printf("不是三角形\n");
```

## 【例5.7】

等腰在先  
等边在后  
是否可以?



## ■ Questions and answers

