



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

编程实现嵌套的选择分支

1、选择结构的嵌套

3.4.3 选择结构的嵌套

- ▶ 在if语句和switch语句中，分支的子语句可以是任意的控制语句，当这些子语句是if语句或switch语句时，就构成了选择结构的嵌套。
- ▶ 对于复杂程序来说，有着复杂的分支形式，因而需要选择结构的嵌套来实现。

3.4.3 选择结构的嵌套

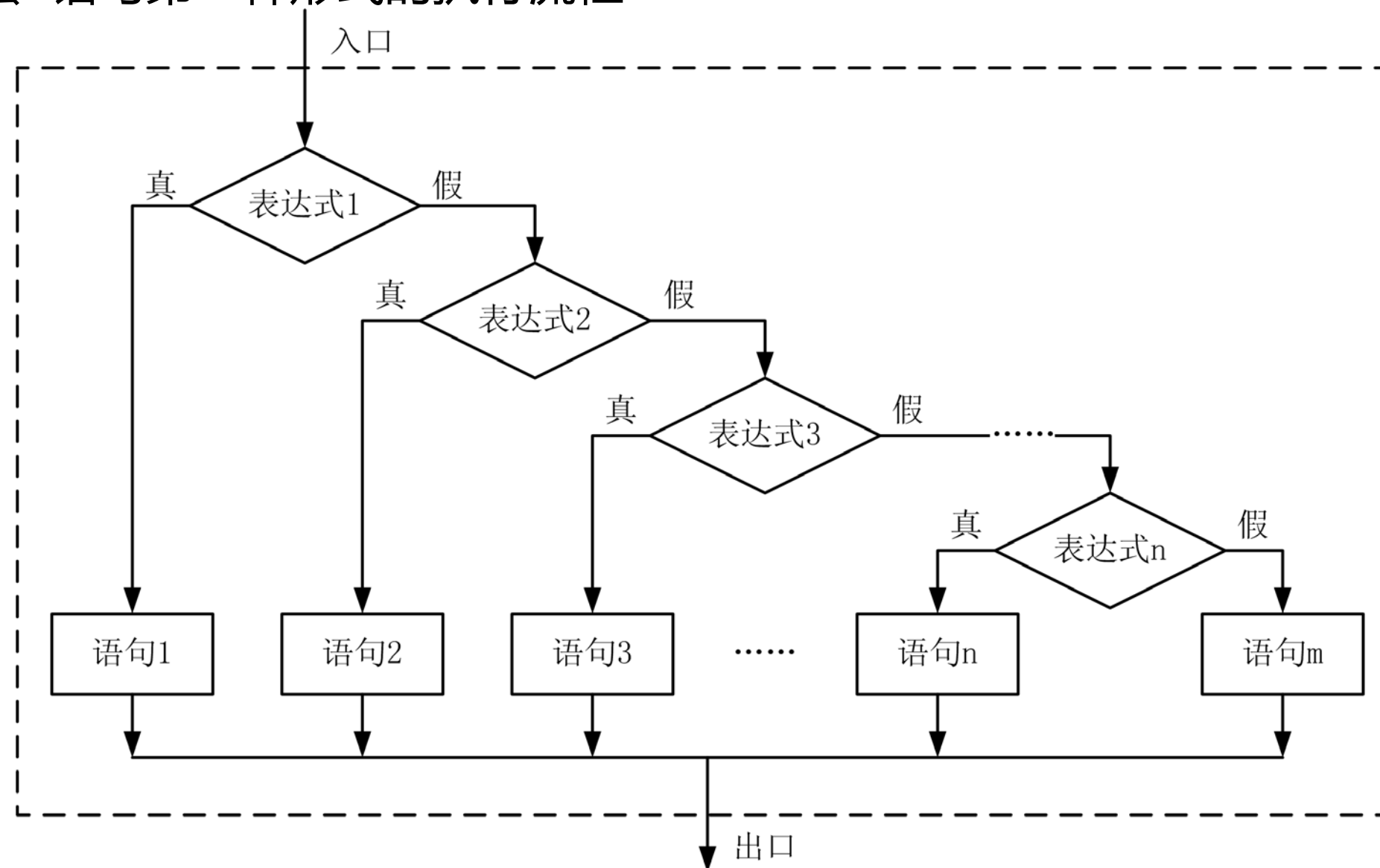
▶ 1. if语句的嵌套

- ▶ (1) 第一种形式，在else分支上嵌套if语句，语法形式为：

```
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n) 语句n  
else 语句m
```

3.4.3 选择结构的嵌套

图3.5 嵌套if语句第一种形式的执行流程



3.4.3 选择结构的嵌套



【例3.4】

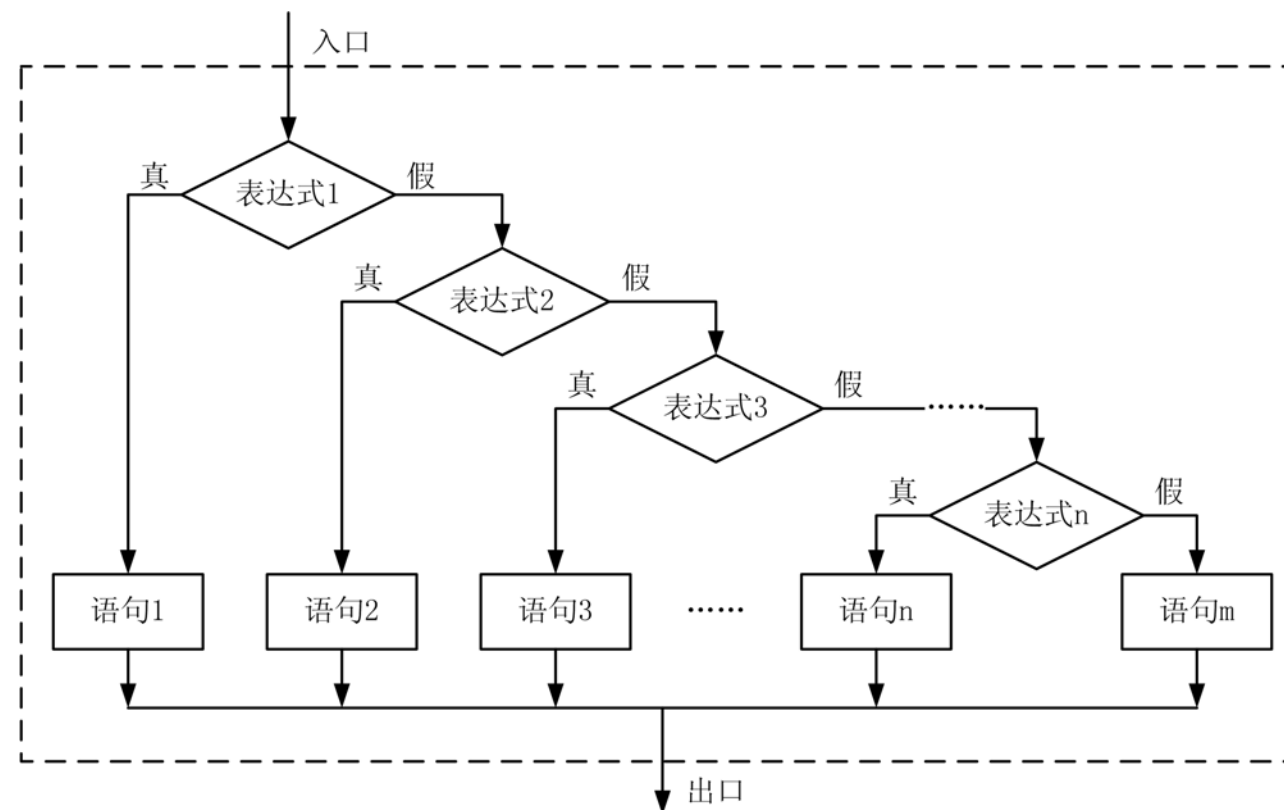
编程输出成绩分类，90以上为A，80～89为B，……，60以下为E。

3.4.3 选择结构的嵌套



例题分析

如图所示。大于90以上为一个分支，80~89为一个分支，依次类推。



3.4.3 选择结构的嵌套

例3.4

```
1 #include <stdio.h>
2 int main()
3 {
4     int score;
5     scanf("%d",&score);
6     if (score >= 90 ) printf("A\n"); //90分以上
7     else if (score >= 80 ) printf("B\n"); //80~89分
8     else if (score >= 70 ) printf("C\n"); //70~79分
9     else if (score >= 60 ) printf("D\n"); //60~69分
10    else printf("E\n"); //60分以下
11    return 0;
12 }
```

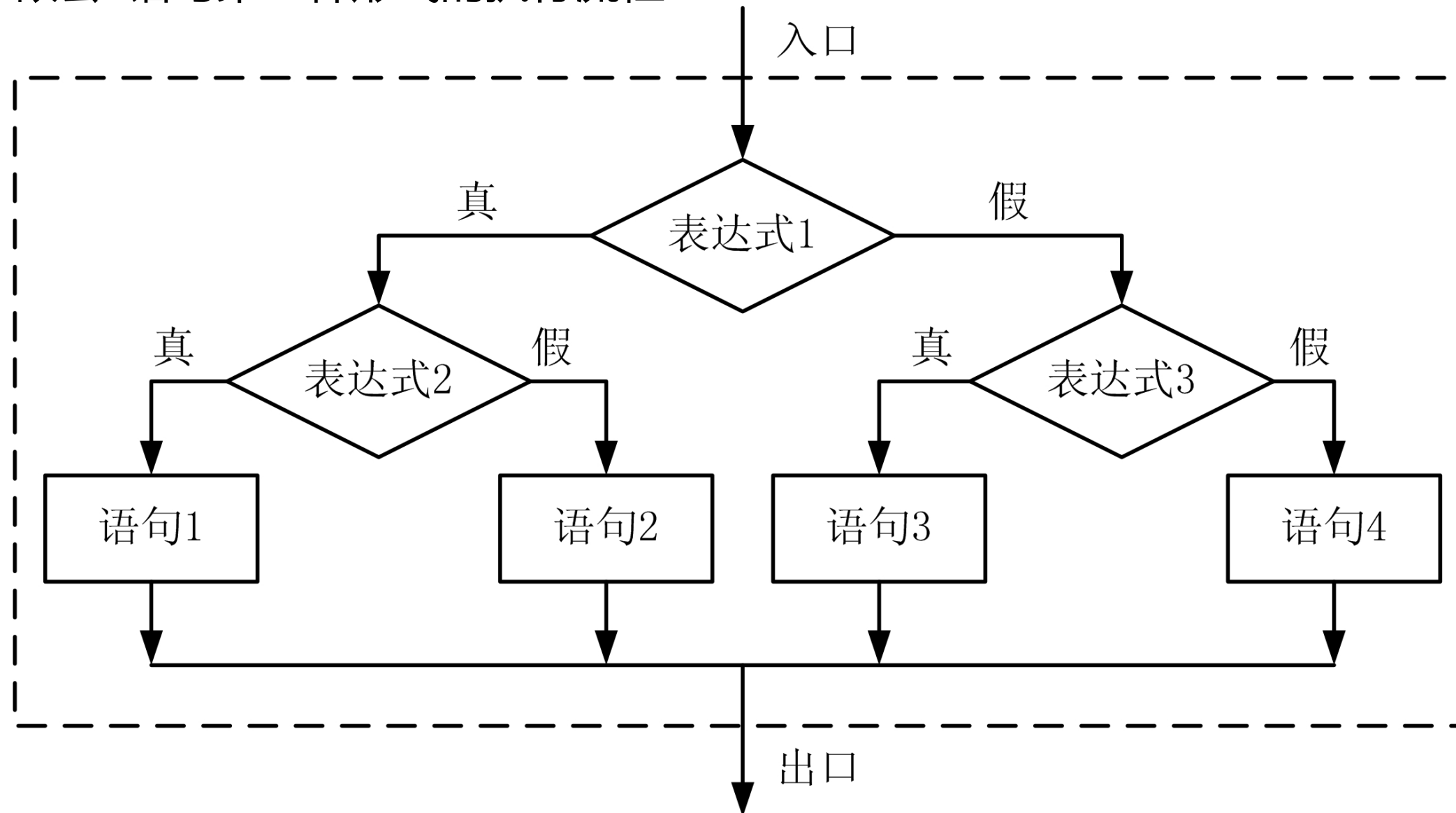

3.4.3 选择结构的嵌套

- ▶ (2) 第二种形式，在if和else分支上嵌套if语句，语法形式为：

```
if (表达式1)
    if (表达式2) 语句1
    else 语句2
else
    if (表达式3) 语句3
    else 语句4
```

3.4.3 选择结构的嵌套

图3.6 嵌套if语句第二种形式的执行流程



3.4.3 选择结构的嵌套

- ▶ if语句嵌套的层数没有限制，可以形成多重嵌套。
- ▶ 多重嵌套的使用，扩展了程序选择的分支数目，适应了程序多分支选择流程的需要。
- ▶ 但是，嵌套的层数越多，编写和理解代码的难度就越大，所以应尽可能使if语句的嵌套层数最少。

3.4.3 选择结构的嵌套

- ▶ ①if多重嵌套容易出现if与else的配对错误，从而引起二义性。

- ▶ 例如：

```
1 if(x>1)
2     if (x>10) y=1;
3 else y=2; //第2行的else分支
```

- ▶ 和

```
1 if(x>1) {
2     if (x>10) y=1;
3 }
4 else y=2; //第1行的else分支
```

3.4.3 选择结构的嵌套

- ▶ 嵌套中的if与else的配对关系原则为：else总是匹配给上面相邻尚未配对的if。如果if和else的数目不对应，使用复合语句来明确配对关系。

3.4.3 选择结构的嵌套

▶ ②对选择条件优化可以减少if的嵌套层数。

▶ 例如：

```
1  if(x>1)
2    if (x>10) y=1;
```

▶ 是可以这样写的：

```
1  if(x>1 && x>10) y=1;
```

3.4.3 选择结构的嵌套

- ▶ ②对选择条件优化可以减少if的嵌套层数。
- ▶ 用布尔代数简化逻辑式子

基本定律	$A + 0 = A$	$A \cdot 0 = 0$	$\overline{\overline{A}} = A$
	$A + 1 = 1$	$A \cdot 1 = A$	
	$A + A = A$	$A \cdot A = A$	
	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$	
结合律	$(A + B) + C = A + (B + C)$		$(AB)C = A(BC)$
交换律	$A + B = B + A$		$AB = BA$
分配律	$A(B + C) = AB + AC$		$A + BC = (A + B)(A + C)$
摩根定律	$\overline{A \cdot B \cdot C \cdots} = \overline{A} + \overline{B} + \overline{C} \cdots$		$\overline{A + B + C \cdots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdots$
吸收律	$A + A \cdot B = A$		
	$A \cdot (A + B) = A$		
	$A + \overline{A} \cdot B = A + B$		
	$(A + B) \cdot (A + C) = A + BC$		

3.4.3 选择结构的嵌套

- ▶ ③选择正确的算法可以大幅降低if多重嵌套带来的复杂度。
- ▶ 例如：将A、B、C、D四个数按由小到大的顺序打印出来。
 - 方法一：选两个数比较，得到两个分支，在每个分支中再去与第3个数比较，然后再去与第4个数比较，最终得到24个分支。
 - 方法二：A、B比较交换得到 $A < B$ ，A、C比较交换得到 $A < C$ ，A、D比较交换得到 $A < D$ ，则最小是A。
 - B、C比较交换得到 $B < C$ ，B、D比较交换得到 $B < D$ ，则次小是B。
 - C、D比较交换得到 $C < D$ ，则最大是D，次大是C。

3.4.3 选择结构的嵌套



【例3.5】

输入4个数a, b, c, d, 按由小到大的顺序输出。

3.4.3 选择结构的嵌套



例题分析

如果按照逐一比较输出或相似的思路去求解，就会用到if多重嵌套，会有24个分支（全排列4!）。这样的嵌套是复杂的，也很难理解。例如：

```
if(a>b && b>c && c>d) printf("%d,%d,%d,%d",a,b,c,d);  
else if(a>b && b>c && d>c) printf("%d,%d,%d,%d",a,b,d,c);
```

.....

可以采用两两比较交换的思路来做，由于比较完后有交换，对于后面的步骤来说这两个数已经确定了大小关系，比较的次数会越来越少。

3.4.3 选择结构的嵌套

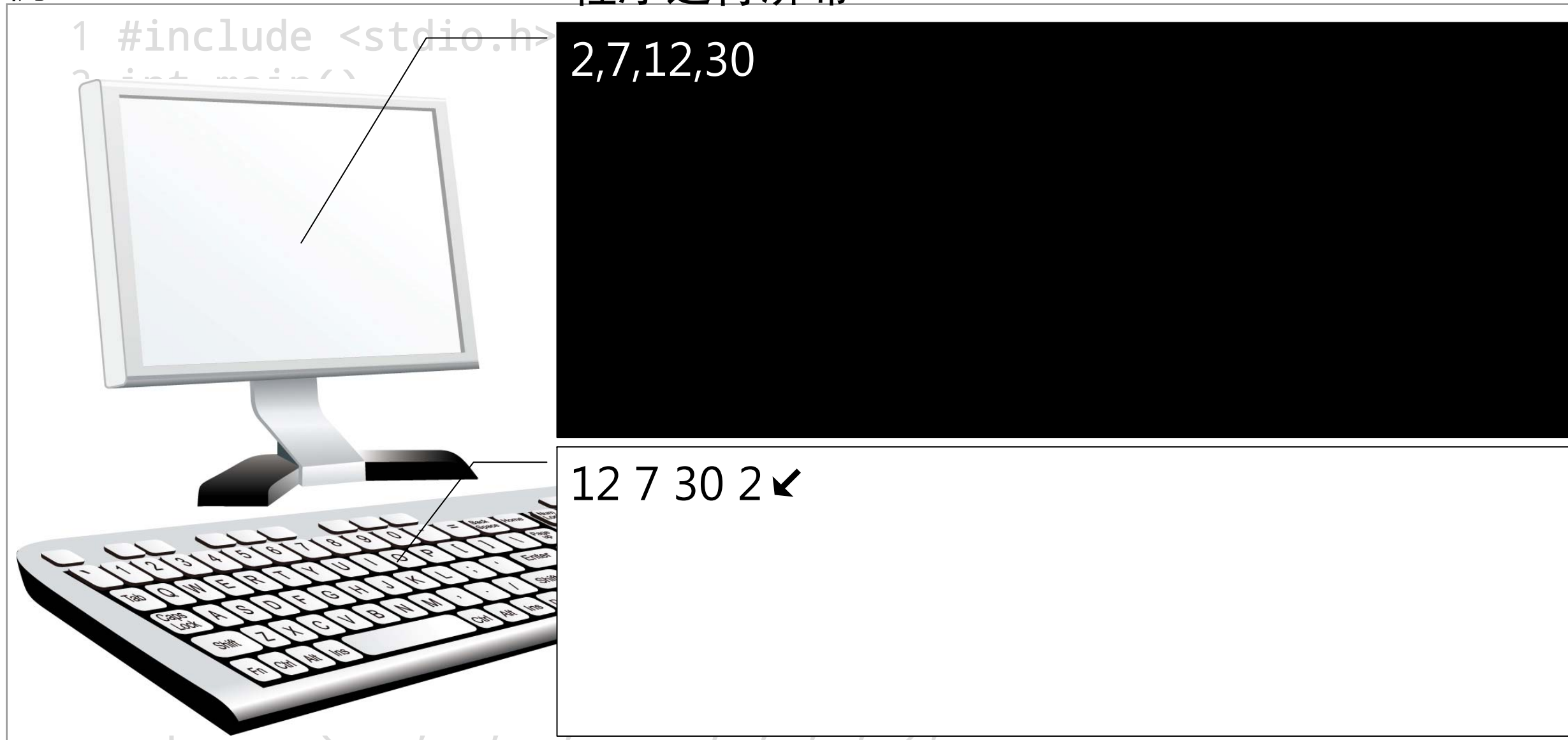
例3.5

```
1  #include <stdio.h>
2  int main()
3  {
4      int a,b,c,d,t;
5      scanf("%d%d%d%d",&a,&b,&c,&d);
6      if(a>b) t=a,a=b,b=t; //结果a<=b
7      if(a>c) t=a,a=c,c=t; //结果a<=c
8      if(a>d) t=a,a=d,d=t; //结果a<=d
9      //结果 a < b,c,d
10     if(b>c) t=b,b=c,c=t; //结果b<=c
11     if(b>d) t=b,b=d,d=t; //结果b<=d
12     //结果 a < b < c,d
13     if(c>d) t=c,c=d,d=t; //结果c<=d
14     //结果 a < b < c < d
15     printf("%d,%d,%d,%d\n",a,b,c,d);
```

3.4.3 选择结构的嵌套

例3.5

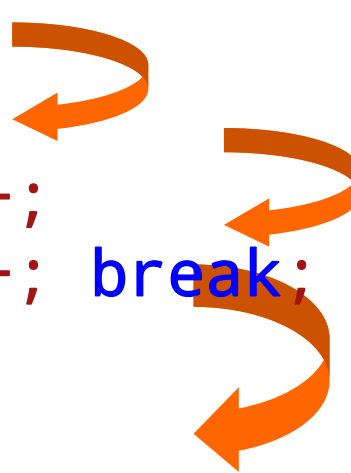
程序运行屏幕



3.4.3 选择结构的嵌套

- ▶ 2. switch语句的嵌套
- ▶ switch语句是可以嵌套的。例如：

```
1 int a=15, b=21, m=0;  
2 switch(a%3) {  
3     case 0: m++;  
4         switch(b%2) {  
5             default: m++;  
6             case 0 : m++; break;  
7         }  
8     case 1: m++;  
9 }
```

The diagram illustrates the execution flow of the nested switch statements. Three orange curved arrows are present: the first arrow starts at the 'switch(b%2)' statement on line 4 and points to the 'default' case on line 5; the second arrow starts at the 'break;' statement on line 6 and points to the closing brace of the inner switch on line 7; the third arrow starts at the 'break;' statement on line 6 and points to the 'case 1' on line 8, indicating that the outer switch continues its execution.

3.4.3 选择结构的嵌套

- ▶ 第3行case分支执行，接下来第4行switch语句执行，当第6行break执行时结束“switch(b%2)”语句，转到第8行继续执行上一级“switch(a%3)”语句的case分支。
- ▶ 从上面的例子看出，switch语句中的break语句仅终止包含它的switch语句，而不是嵌套中的所有switch语句。使用嵌套的switch语句后，程序的分支变得复杂。

3.4.3 选择结构的嵌套

- ▶ 3. if语句和switch语句的替换
- ▶ 从逻辑上看，if语句和switch语句是可以相互替换的。
- ▶ switch语句的分支判定是按相等来处理的，而if语句的条件判定就要比它宽广得多。显然，switch语句可以直接用第一种嵌套形式的if语句写出来。

3.4.3 选择结构的嵌套

- ▶ 但if语句用switch语句来写，就有难度了。原因是用相等判定去代替区间判定有实现上的困难，例如“ $a < x < b$ ”如何用“x等于什么”的意思来描述。
- ▶ 一般地，switch语句的程序完全可以用if语句写出来，if语句的程序一定条件下可以用switch语句写出来。
- ▶ 使用switch语句比用if-else语句简洁，可读性高。遇到多分支选择的情形，应当尽量选用switch语句，避免采用嵌套较深的if-else语句。

CP 程序设计