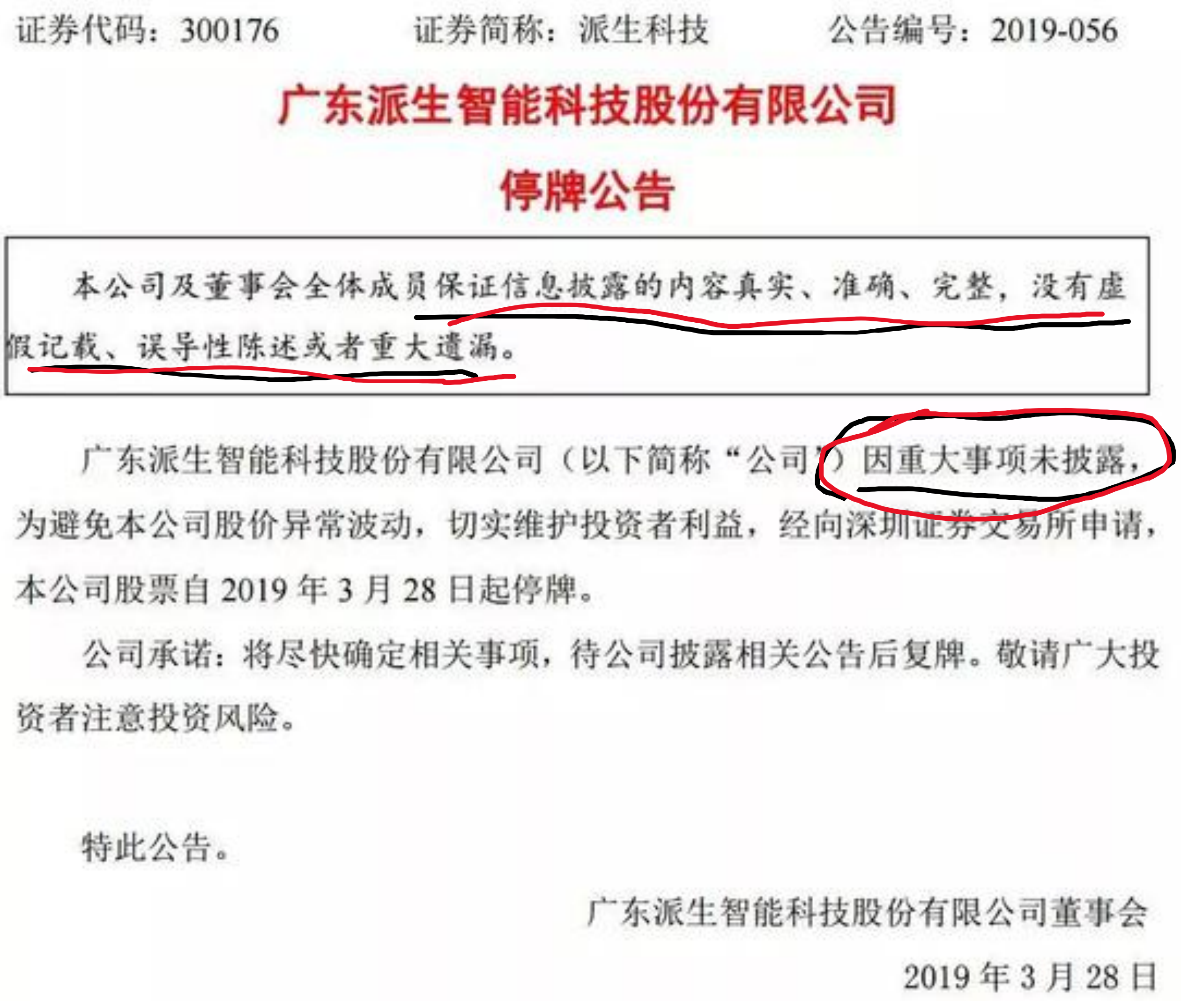
说谎者悖论



证券代码：300176　　　　证券简称：派生科技　　　　公告编号：2019-056

广东派生智能科技股份有限公司

停牌公告

本公司及董事会全体成员保证信息披露的内容真实、准确、完整，没有虚假记载、误导性陈述或者重大遗漏。

广东派生智能科技股份有限公司（以下简称“公司”）因重大事项未披露，为避免本公司股价异常波动，切实维护投资者利益，经向深圳证券交易所申请，本公司股票自 2019 年 3 月 28 日起停牌。

公司承诺：将尽快确定相关事项，待公司披露相关公告后复牌。敬请广大投资者注意投资风险。

特此公告。

广东派生智能科技股份有限公司董事会

2019 年 3 月 28 日

Turing Award

"They should not follow the trend—which right now is deep learning"

释疑

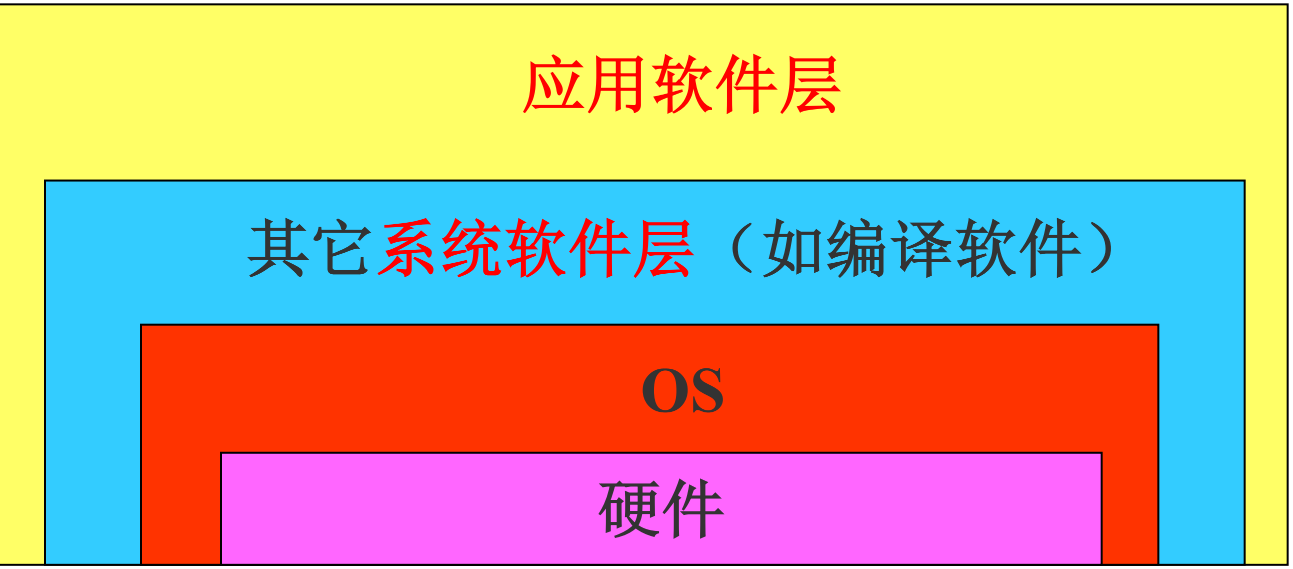
- 关于猜随机数的演示
  - rand()函数的用法示例
  - time(NULL)返回系统时间（秒级）
  - DEMO（注意用“打桩”的方式输出随机数）
- 关于隐式类型转换的一个奇怪例子
  - Convert int to unsigned int
  - DEMO

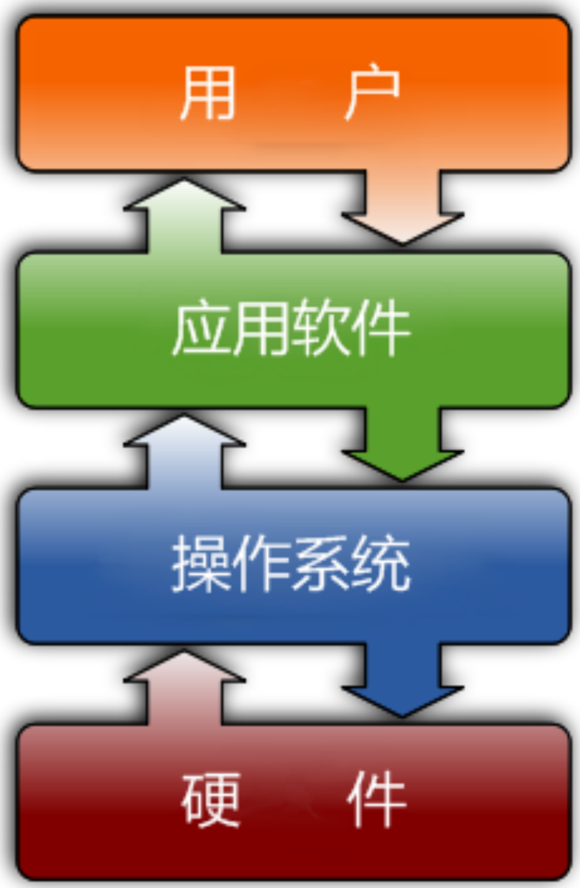
操作系统

- 操作系统的定义
  - Operating System, OS
  - 操作系统是位于硬件层之上，所有其它软件层之下的一个系统软件，是管理和控制系统中各种软硬件资源，方便用户使用计算机系统的程序集合。
- 操作系统的目的
  - 更合理地管理和分配系统资源，提高工作效率
  - 提供更友好的服务界面
    - API 、GUI
  - 为系统提供功能扩展平台

操作系统

- 操作系统的位置





操作系统

- 操作系统的主要任务
  - 进程管理（Processing management）
    - 线程
  - 内存管理（Memory management）
    - 虚存
  - 文件系统（File system）
    - 如存储格式，目录索引，后缀
  - 网络通讯（Networking）
  - 安全机制（Security）
  - 用户界面（User interface）
  - 驱动程序（Device drivers）

操作系统

- 操作系统的特性
  - 程序并发性
    - 多个程序在宏观上同时向前推进、微观上串行推进
    - 分时（桌面，移动）和实时（工业，飞机）
  - 资源共享性
    - 多个程序共用系统中的各种软硬件资源
  - 虚拟性
    - 物理上的一台设备变成逻辑上的多台设备

School of Economics ang Management, Beihang University

Introduction to C Programming

Jichang Zhao

jichang@buaa.edu.cn

Selection

Objectives

- Relational Expressions
- The `if` and `if-else` Statements
- The `if-else` Chain
- The `switch` Statement
- 错误，测试和调试
  - 4.8，务必阅读

Introduction

- **Flow of control**
  - the order in which a program’ s statements are executed
- Any algorithm can be built using combinations of four standardized flow of control structures
  - Normal flow of control for all programs is *sequential*
  - *Selection* is used to select which statements are performed next based on a **condition**
  - （**repetition, invocation)**
  - （**iteration, jump)**

Relational Expressions

- Simplest decision structure

  if (condition)

  statement executed if condition is true

  - The condition is evaluated to determine its numerical value, which is interpreted as either true (non-zero) or false (0)
  - If condition is "true" the statement following the `if` is executed; otherwise, **statement is not executed**
- The condition used in all of C's `if` statements can be any valid C expression
  - Most commonly, a **relational expression** (can yield only 0 or 1)
  - 也是一种好的风格

Relational Expressions

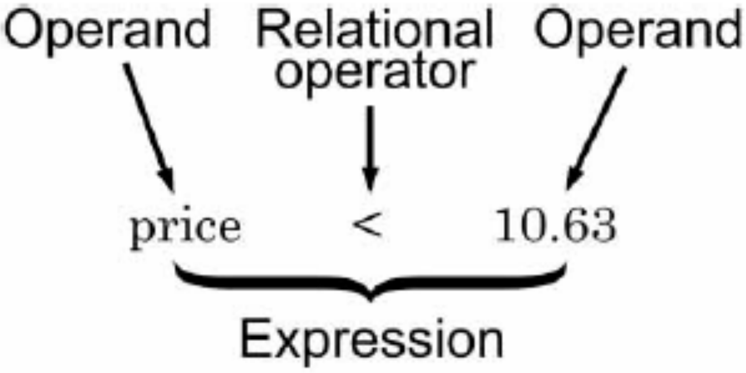


Figure 4.1 Anatomy of a simple relational expression

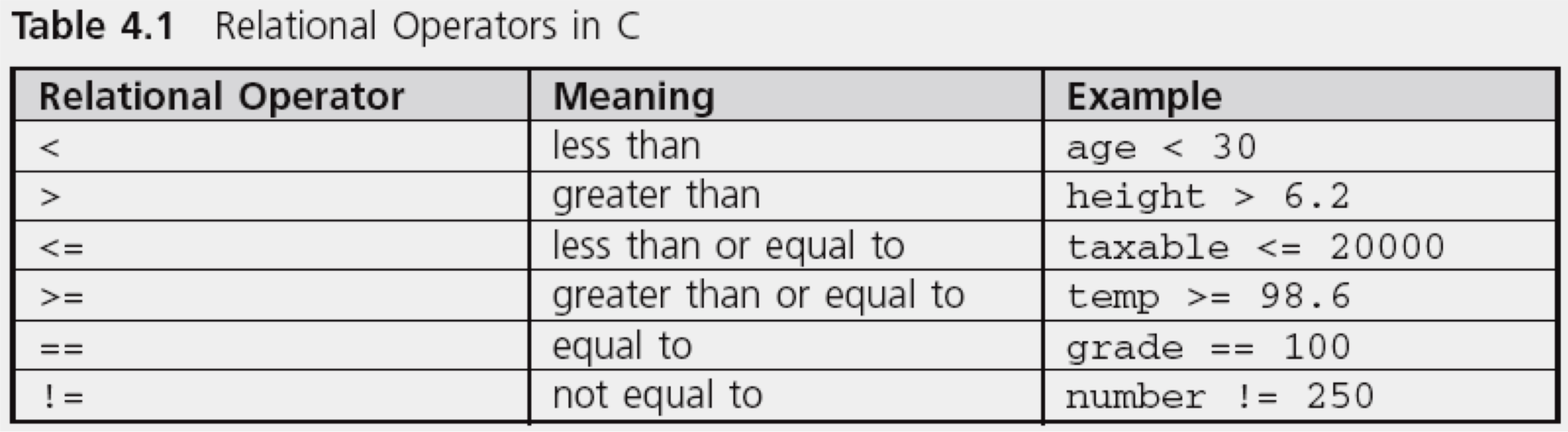


Table 4.1 Relational Operators in C

Relational Operator	Meaning	Example
`<`	less than	`age < 30`
`>`	greater than	`height > 6.2`
`<=`	less than or equal to	`taxable <= 20000`
`>=`	greater than or equal to	`temp >= 98.6`
`==`	equal to	`grade == 100`
`!=`	not equal to	`number != 250`

Relational Expressions

- Relational expressions are also known as **conditions**
- A relational expression evaluates to 1 (true) or 0 (false)
  - The expression 3 < 4 has a value of 1
  - The expression 2.0 > 3.3 has a value of 0
  - The value of `hours` > 0 depends on the value of `hours`
- Character data can also be compared using relational operators
  - 字符串不可以，需要专门的比较函数

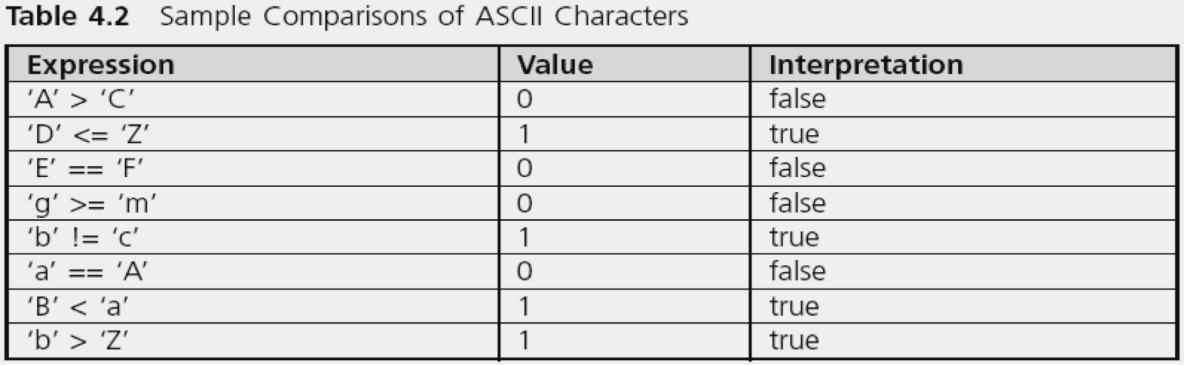


Table 4.2 Sample Comparisons of ASCII Characters

Expression	Value	Interpretation
'A' > 'C'	0	false
'D' <= 'Z'	1	true
'E' == 'F'	0	false
'g' >= 'm'	0	false
'b' != 'c'	1	true
'a' == 'A'	0	false
'B' < 'a'	1	true
'b' > 'Z'	1	true

Logical Operators

- More complex conditions can be created using the logical operations AND (&&), OR (||), and NOT (!)
- When the && is used with two expressions, the condition is true only if both expressions are true by themselves

Logical Operators

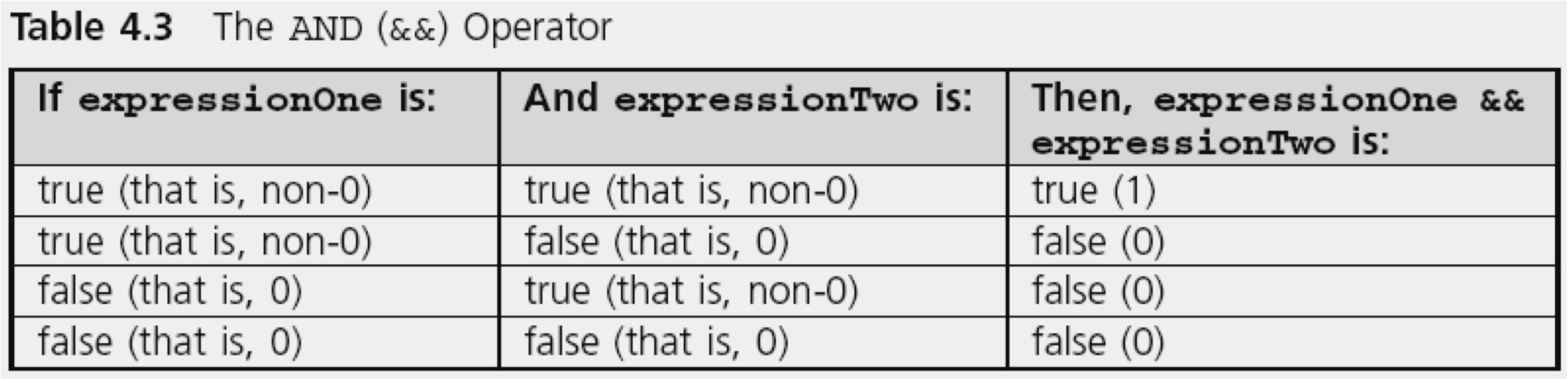


Table 4.3 The AND (&&) Operator

If expressionOne is:	And expressionTwo is:	Then, expressionOne && expressionTwo is:
true (that is, non-0)	true (that is, non-0)	true (1)
true (that is, non-0)	false (that is, 0)	false (0)
false (that is, 0)	true (that is, non-0)	false (0)
false (that is, 0)	false (that is, 0)	false (0)

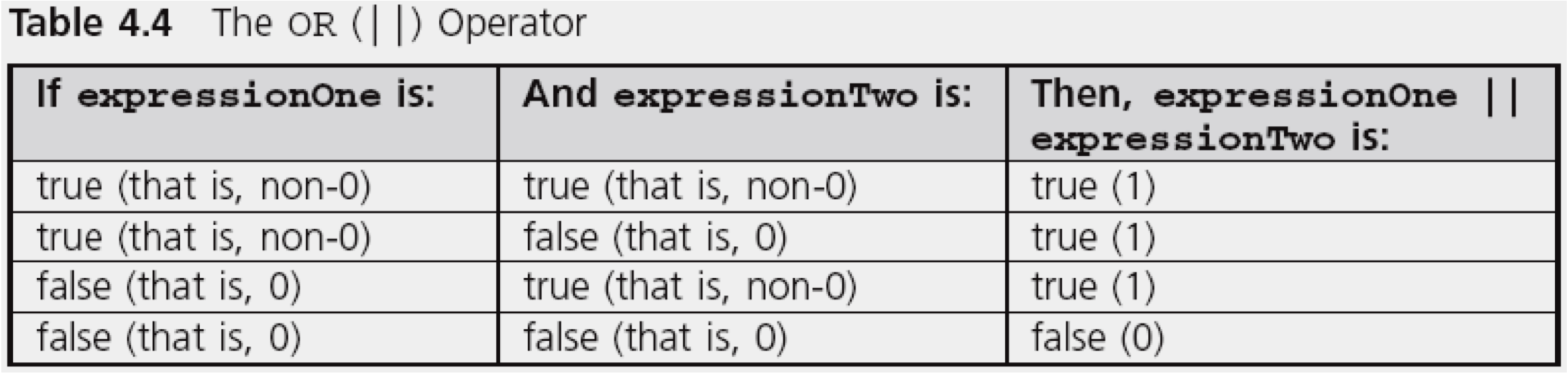


Table 4.4 The OR (| |) Operator

If expressionOne is:	And expressionTwo is:	Then, expressionOne || expressionTwo is:
true (that is, non-0)	true (that is, non-0)	true (1)
true (that is, non-0)	false (that is, 0)	true (1)
false (that is, 0)	true (that is, non-0)	true (1)
false (that is, 0)	false (that is, 0)	false (0)

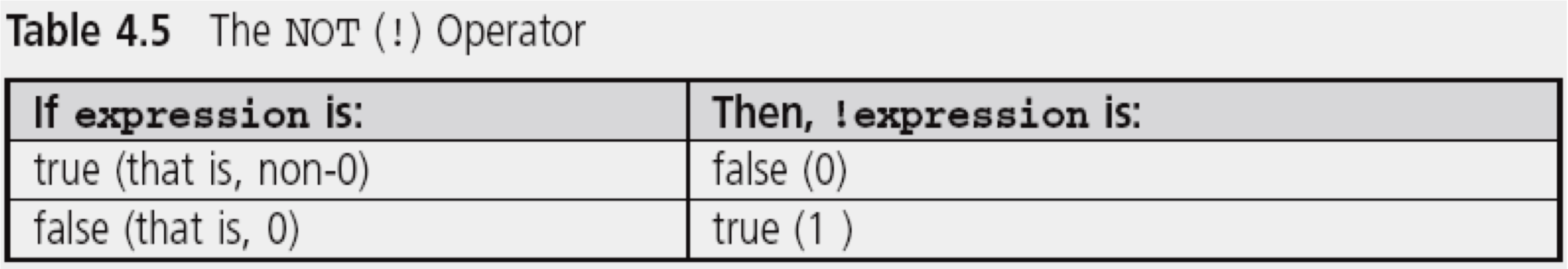


Table 4.5 The NOT (!) Operator

If expression is:	Then, !expression is:
true (that is, non-0)	false (0)
false (that is, 0)	true (1)

Logical Operators

- && is evaluated first, before ||
- The evaluation feature for the && and || operators that makes the evaluation of an expression stop as soon as it is determined that an expression is false is known as **short-circuit evaluation**
- Parentheses can be used to alter the assigned operator priority

```
(6 * 3 == 36 / 2) && (13 < 3 * 3 + 4) || !(6 - 2 < 5)
```

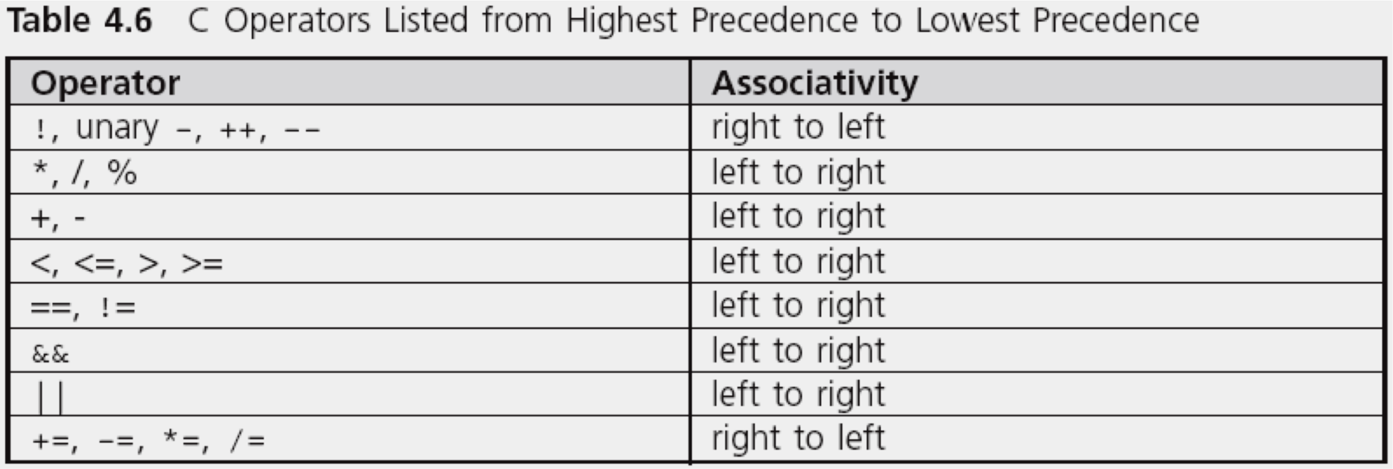
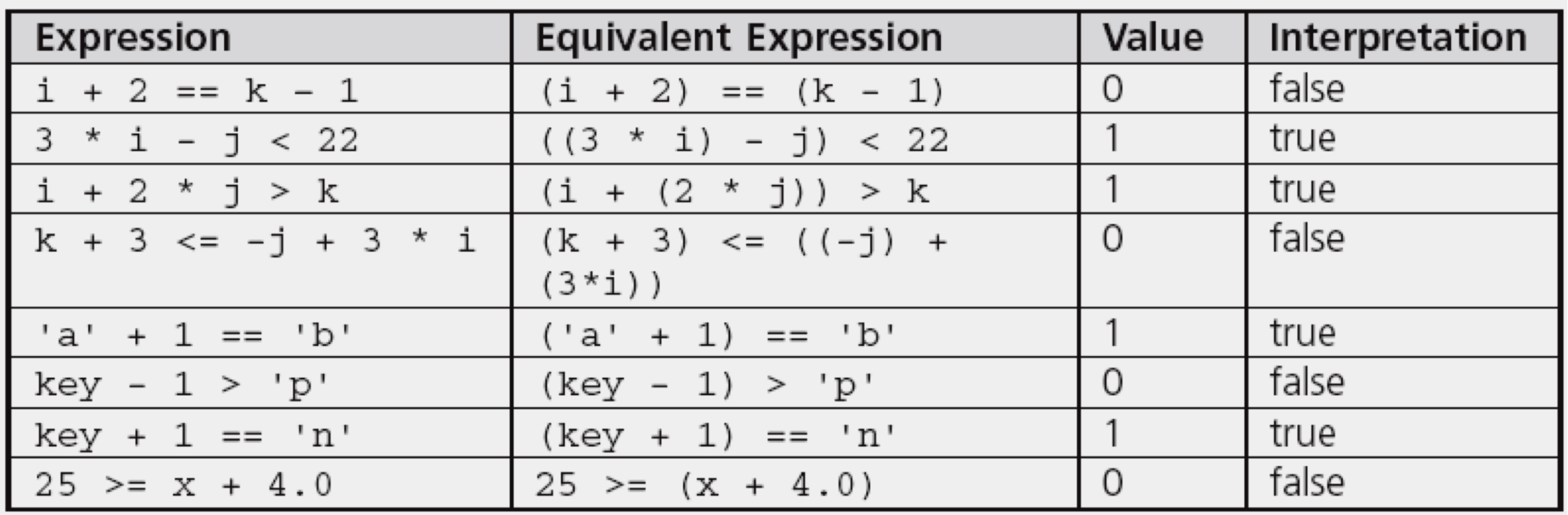


Table 4.6 C Operators Listed from Highest Precedence to Lowest Precedence

Operator	Associativity
!, unary -, ++, --	right to left
*, /, %	left to right
+, -	left to right
<, <=, >, >=	left to right
==, !=	left to right
&&	left to right
||	left to right
+=, -=, *=, /=	right to left

Quiz

```
char key = 'm';
int i = 5, j = 7, k = 12;
double x = 22.5;
```



Expression	Equivalent Expression	Value	Interpretation
`i + 2 == k - 1`	`(i + 2) == (k - 1)`	0	false
`3 * i - j < 22`	`((3 * i) - j) < 22`	1	true
`i + 2 * j > k`	`(i + (2 * j)) > k`	1	true
`k + 3 <= -j + 3 * i`	`(k + 3) <= ((-j) + (3*i))`	0	false
`'a' + 1 == 'b'`	`('a' + 1) == 'b'`	1	true
`key - 1 > 'p'`	`(key - 1) > 'p'`	0	false
`key + 1 == 'n'`	`(key + 1) == 'n'`	1	true
`25 >= x + 4.0`	`25 >= (x + 4.0)`	0	false

Quiz

```
int i = 9, j = 15;
double a = 3.5, b = 2.1, complete = 0.0;
```

Expression	Value	T or F
a > b	T	
a==b || i<j || complete	T	
a / b <1 && complete < 3.0	F	
i<4 && j == 13 || a < b && j*i >= 23	F	

Quiz

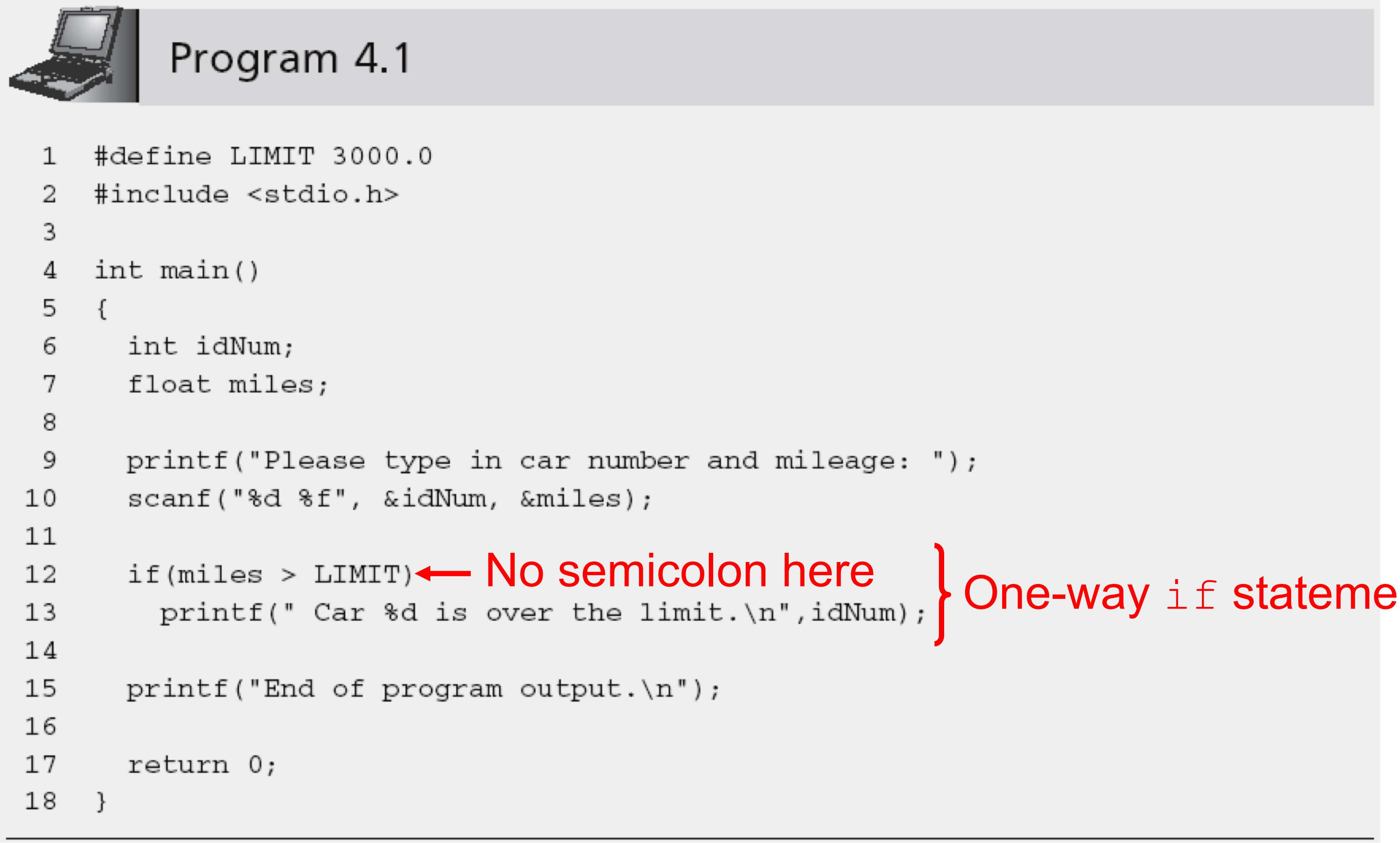
- `if (c = ‘ ‘ || c == ‘\t’ || c == ‘\n’)`
    - `c=getc(f);`
    - 并不能忽略空白符\t和\n
    - 在关系表达式里进行变量更新要注意，在短路求值下，有可能不会实现更新
- 一种好的风格
    - if (age==40)应该写成if (40==age)
    - P136-137注解

补充：条件表达式

- Conditional operator
  - expresssion1 ? expression2 : expression3
  - Ternary operator
  - 如果expression1成立，那么表达式2，否则表达式3

  - int i, j, k;
  - i=1;
  - j=2;
  - k=i>j ? i : j; // k=2
  - k=(i>=0 ? i : 0)+j; // k=3

The if and if-else Statements



Program 4.1

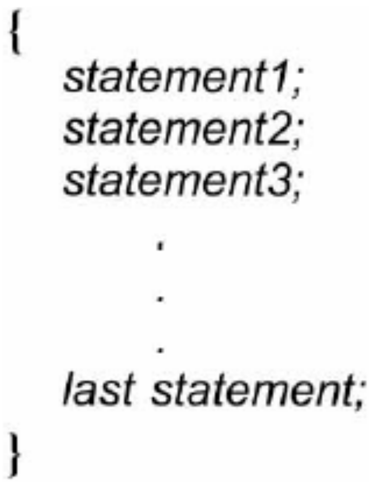
```c
#define LIMIT 3000.0
#include <stdio.h>

int main()
{
  int idNum;
  float miles;

  printf("Please type in car number and mileage: ");
  scanf("%d %f", &idNum, &miles);

  if(miles > LIMIT)
    printf(" Car %d is over the limit.\n",idNum);

  printf("End of program output.\n");

  return 0;
}
```

No semicolon here

One-way if statement

Compound Statements

- Although only a single statement is permitted in an `if` statement, this statement can be a single **compound statement**



```
{
    statement1;
    statement2;
    statement3;
         .
         .
         .
    last statement;
}
```

Figure 4.3 A compound statement

Compound Statements

- For example,

```
if (expression)
{
   statement1; /*as many statements as necessary*/
   statement2; /*can be placed within the braces*/
      •        /*each statement must end with ; */
      •
      •
   statementn;
}
```

- For very short statements, you can code a complete `if` statement placed on a single line
  - `if (grade > 69) ++passTotal;`

The `if-else` Statement

- The most commonly used `if-else` statement is

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

  - If the value of *`expression`* is 0 *`statement2`*, the statement after the reserved word `else`, is executed

The `if-else` Chain

- **Nested `if` statement:**

```
if (expression1)
  statement1;
else
  if (expression2)
    statement2;
  else
    statement3;
```

- Whether the indentation exists or not, the compiler will, by default, associate an `else` with the closest previous unpaired `if`, unless braces are used to alter this default pairing

The `if-else` Chain

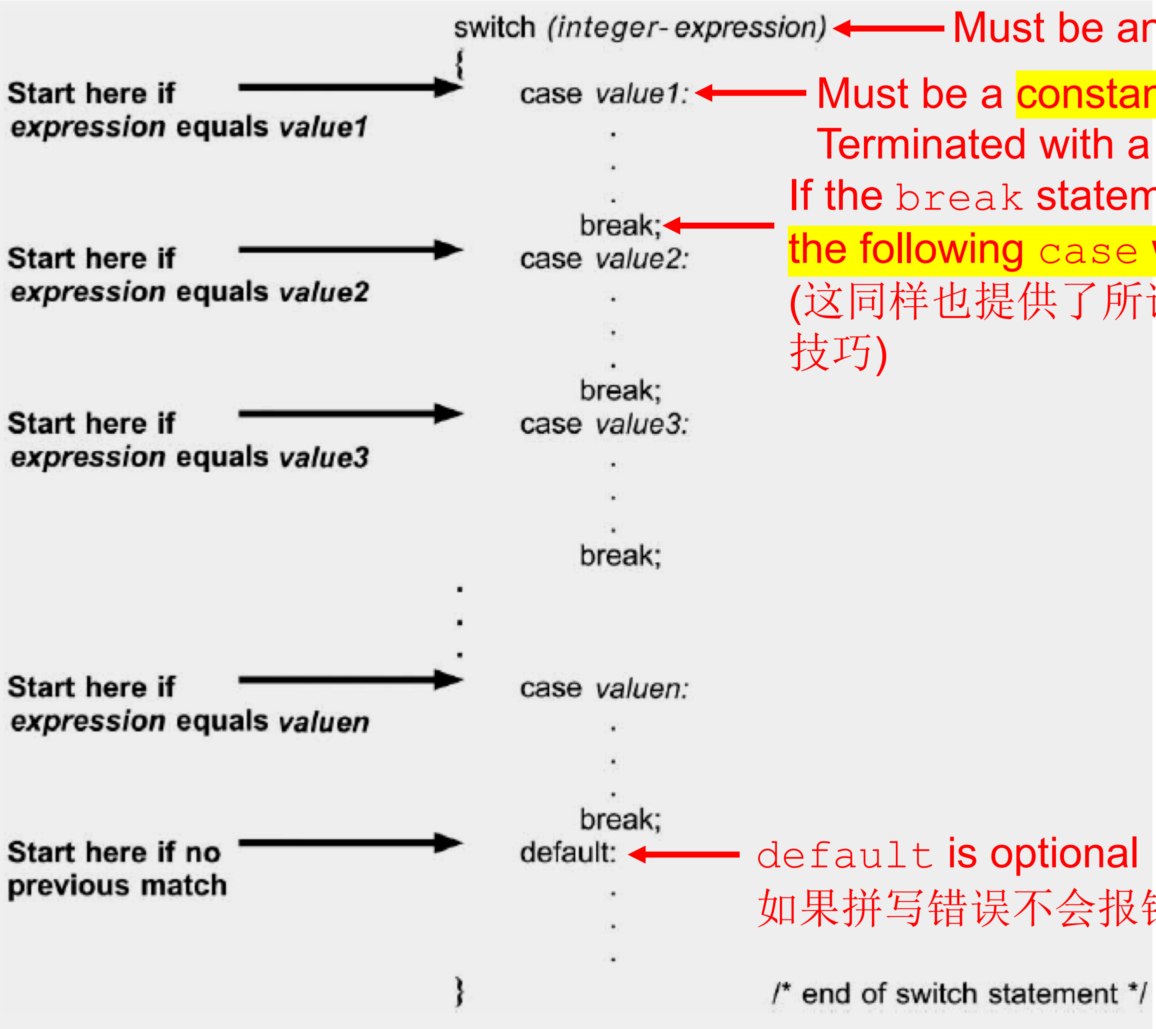
- `if-else` chain:

```
if (expression1)
  statement1;
else if (expression2)
  statement2;
else
  statement3;
```

Quiz

- if (x==0)
  - if (y==0) error();
- else{ // suspending else
  - z=x+y;
- }

The switch Statement



```
switch (integer-expression)
{
     case value1:
        .
        .
        .
        break;
     case value2:
        .
        .
        .
        break;
     case value3:
        .
        .
        .
        break;
  .
  .
  .
     case valuen:
        .
        .
        .
        break;
     default:
        .
        .
        .
}                         /* end of switch statement */
```

Start here if *expression* equals *value1*
Start here if *expression* equals *value2*
Start here if *expression* equals *value3*
Start here if *expression* equals *valuen*
Start here if no previous match

Figure 4.6 The expression determines the entry point

Must be an integer

Must be a constant expression
Terminated with a colon

If the break statement was omitted, the following case would be executed (这同样也提供了所谓的“多重标签”技巧)

default is optional
如果拼写错误不会报错

补充：switch 的一些特殊用法

- 多重标签
  - 1. 如读入的字母不区分大小写时
  - …
  - switch(letter)
  - {
    - case ‘a’ :
    - case ‘A’ : _count_a++; break;
    - …
  - }
  - 2. 利用switch实现100分制向五级制转换，其中90-100为A, 80-89为B, 70-79为C, 60-69为D, 0-59为F。如输入84，返回B。

补充：控制流对程序复杂性的影响

- 分支带来的问题
  - 有些代码可能不会被执行
  - 出现组合“爆炸”
  - 圈复杂度
- 对测试的要求
  - 能否遍历所有的路径？
  - 尽可能覆盖所有路径

Homework

- 1. P138, 1
- 2. P139, 4
- 3. P139, 6
- 4. P139, 8
- 5. P145编程题, 1
- 6. P146, 3
- 7. P152, 1（要求用switch实现）
- 8. P156, 6
- 9. 编写一个程序，要求用户输入24小时制时间，然后显示12小时制和格式。如输入21:11，应输出9:11 PM。注意不要把12:00显示成0:00。