东南大學 SCHOOL OF INTEGRATED CIRCUITS, SEU

集成电路学院

计算机科学基础I

—控制

东南大学 集成电路学院 朱彬武

E-mail: bwzhu@seu.edu.cn

运算符

优先级	名称	运算符	结合关系	例子
1	括号运算符	()	自左向右	

2	自增、自减	++，--	自右向左	int i = 0, j = 0;
				j = ++i;
				j = i++;

2	逻辑非	!		int i = !(1==0)
2	按位取反	~		int i = 0; int j = ~i;
2	正号，符号	+，-		int i = 2, j =3; int k = i *(-j)
3	乘，除，取余	*，/，%	自左向右	
4	加、减	+，-	自左向右	
5	左移，右移	<<，>>	自左向右	
6	小于，小于等于， 大于，大于等于	<，<=，>，>=	自左向右	
7	等于，不等于	==，!=	自左向右	int i = 1==1 int j = 0!=1

结合关系指的是有相同优先级的运算符放在一起，按照指定顺序运算

例子：

- 左结合：a/b*c 等价 (a/b)*c
- 右结合：a=b=6 等价 a=(b=6)

运算符

优先级	名称	运算符	结合关系	例子
8	按位与	&	自左向右	101 & 110
9	按位异或	^	自左向右	101 ^ 110
10	按位或	|	自左向右	101 | 110
11	逻辑且	&&	自左向右	(1!=1)&&(1==1)
12	逻辑或	||	自左向右	(1==1)||(0!=1)
13	条件运算符	?:	自右向左	int a=1>0?2:3;
14	赋值运算符	=	自右向左	int a, b, c = 0 b=a=2 c=b+=2
14	复合赋值运算符	+=，-=，*=，/=	自右向左	
15	逗号运算符	,	自左向右	int i=0 , j=0 j=2, 3

要掌握运算符之间的优先级关系，分类记忆，大体上看是单目运算>算术运算>关系运算>位运算>逻辑运算>赋值运算

逗号运算符

- 逗号运算符：，
- 作用：依次执行多个表达式，结果为最后一个表达式的值
- 语法形式：表达式1，表达式2，…，表达式n
  - 从左到右依次执行
  - 整个逗号表达式的值等于最后一个表达式
- 使用场景：for循环
- 例题：int i = 0, j = 0；j = ++i, i++，表达式的值为（1），i的值为（2），j的值为（1）

补充说明

- /、/=、%、%=的右操作数不能是0
- %、%=和位运算的操作数不能是浮点型
- <<、>>、<<=、>>=的右操作数≥ 0且<32
- %：余数正负符号与被除数一致：-5%2 = -1
- >>：有符号整数带符号右移，无符号整数高位补0
  - x >> n 可以看作是 floor(x / 2^n)，向下取整

考试爱考（容易踩坑）

1. int a = 6; int b; int c = 1 + (b=a)；c的值是（7）。

2. int x=0,y=2,z=3; 则表达式 x==x ||(y=y^y)&&(z=z+1)执行后，说法正确的是 (D)。

A. x 的值为 1 B. y 的值为 2 C. z 的值为 4 D. 表达式值为 0

z的值为什么是3?

短路原则:
短路只适用于 逻辑与（&&） 和 逻辑或（||） 运算符。这两个运算符都是从左到右求值的。
对于逻辑与运算，如果左操作数为 false，右操作数不会被计算。
对于逻辑或运算，如果左操作数为 true，右操作数不会被计算。

实际编程和考试是两回事

可以说，考试喜欢出的复杂表达式都属于不规范的编程，不利于阅读和理解，容易造成读程序时候的误解。所以要避免写出这种复杂表达式。将复杂表达式拆开来，按照你的预期顺序写出来。

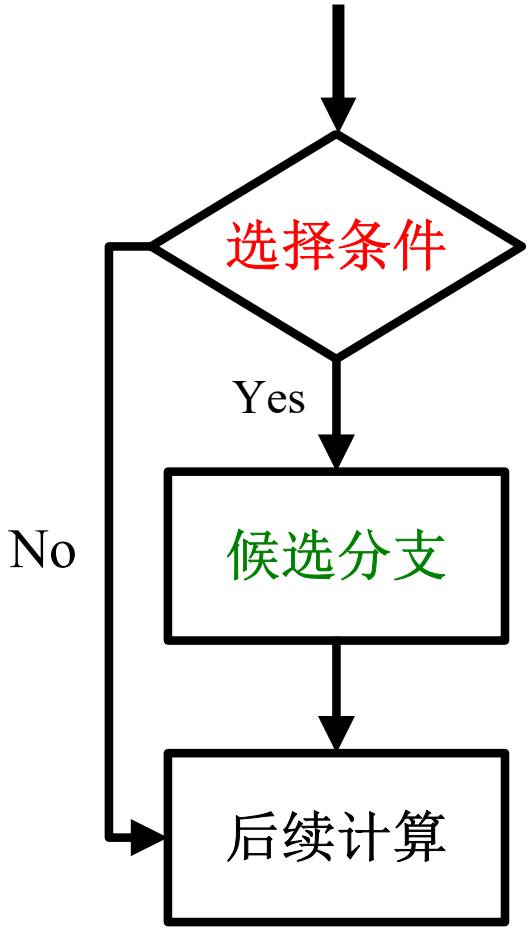
判断——if语句

不严谨的程序


```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
cout << a << " / " << b << " = " << a / b << endl;
cout << a << " * " << b << " = " << a * b << endl;
```

b=0怎么办？

条件判断



- 选择条件：b是否不为0
- 如果不为0，进入候选分支，完成除法计算

修改后的代码——加入if


```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
//判断用户输入的除数是否为0
if (b!=0){
    cout << a << " / " << b << " = " << a / b << endl;
}
cout << a << " * " << b << " = " << a * b << endl;
return 0;
```

程序调试

调试是初学者必须要掌握的技能，是帮助我们理解程序，修复程序错误的有效办法。

调试的常见方法：

（1）设置断点：让程序在特定行暂停执行，方便观察状态

用法：点击代码行左侧空白处设置断点，运行调试模式时，程序会在断点处停下来

（2）单步调试：让程序一步步执行，观察每条语句的效果

常用命令：

操作	说明
Step Over	执行当前行，不进入函数内部
Step Into	进入函数内部逐行调试
Step Out	跳出当前函数返回上层
Continue	继续运行到下一个断点

单分支判断语句


```
if (b!=0){
    cout << a << " / " << b << " = " << a / b << endl;
}
cout << a << " * " << b << " = " << a * b << endl;
```

- if语句

```
if（控制表达式）{
        语句1;
        语句2;
}
语句3;
```

- 条件为真，执行{}内的语句，条件为假则跳过
- if语句后面若还有语句，它们在if结束后会执行，无论条件如何

if语句简单练习

编写一个程序，用户输入分数，若大于等于60分且小于等于100分时，告诉用户距离满分（100分）还差多少分，再换行输出“继续努力”。

```
int score = 0;
cout << "请输入你的成绩" << endl;
cin >> score;
if(score >= 60 && score <= 100){
    cout << "距离满分还差" << 100 - score << "分" << endl;
    cout << "继续努力" << endl;
}
```

如果写成60<=score<=100
会发生什么?

条件表达式不仅可以判断一个条件，还可以组合多个条件，逻辑运算符：与（&&），或（||），非（!）

没有{}的if语句

也可以写成 !(b==0)

```
if (b!=0)
    cout << a << " / " << b << " = " << a / b << endl;
```

- 没有{}的if语句

  if（条件表达式）

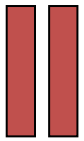
  语句1；

  语句2；

- if语句后面可以没有大括号，但这时候只有语句1是属于if语句，语句2无论条件表达式是否成立都会执行。

没有{}的if语句


```
int score = 0;
cout << "请输入你的成绩" << endl;
cin >> score;
if(score>=60)
    cout << "距离满分还差" << (100-score) << "分" << endl;
    cout << "继续努力" << endl;
```



```
int score = 0;
cout << "请输入您的成绩" << endl;
cin >> score;
if(score>=60)
    cout << "距离满分还差" << (100-score) << "分" << endl;
cout << "继续努力" << endl;
```

在 C++ 中，缩进（indentation）不会影响程序的运行结果或语义。

如果用户输入61分，程序输出是什么？

如果用户输入59分，程序输出是什么？

判断——if-else语句

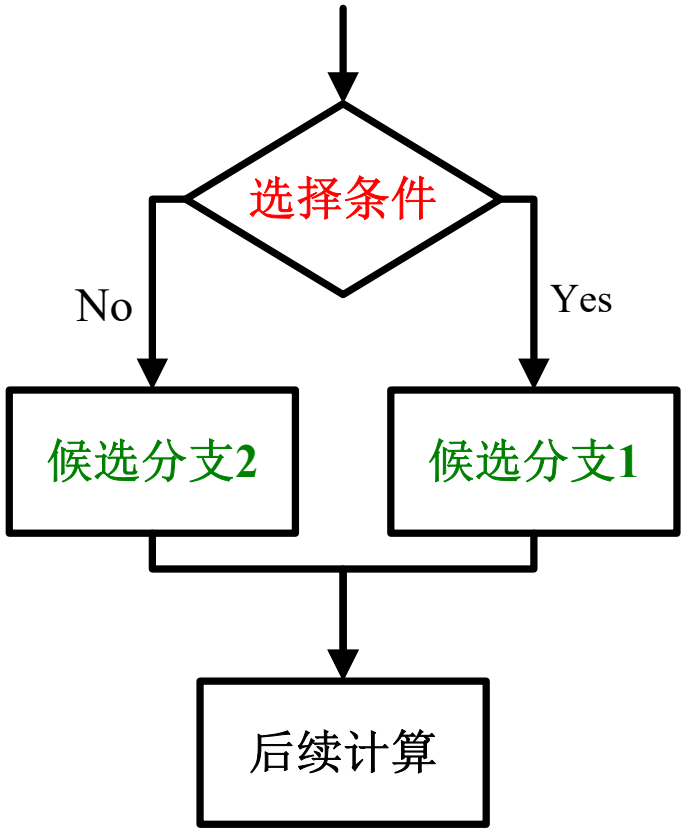
不贴心的程序


```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
//判断用户输入的除数是否为0
if (b!=0){
    cout << a << " / " << b << " = " << a / b << endl;
}
cout << a << " * " << b << " = " << a * b << endl;
return 0;
```

如果用户一不小心输入了0，虽然不会报错，但是会发生什么?

是不是应该给用户点提示?

条件判断



- 选择条件：b是否不为0
- 如果不为0，进入候选分支，完成除法计算
- 否则提示用户，输入的除数为0

修改后的代码——加入if-else


```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
//判断用户输入的除数是否为0
if (b!=0){
    cout << a << " / " << b << " = " << a / b << endl;
}
else{
    cout << "除数不应该为0" << endl;
}
cout << a << " * " << b << " = " << a * b << endl;
```

双分支判断语句


```
if (b!=0){
    cout << a << " / " << b << " = " << a / b << endl;
}
else{
    cout << "除数不应该为0" << endl;
}
cout << a << " * " << b << " = " << a * b << endl;
```

- if-else语句格式

  if（条件表达式）{

  }

  else{

  }

- 条件为真，执行if{}内的语句，条件为假则执行else内的代码
- if-else后面若还有语句，它们在if-else结束后会执行，无论条件如何

没有{}的if-else语句


```
if (b!=0)
    cout << a << " / " << b << " = " << a / b << endl;
else
    cout << "除数不应该为0" << endl;
```

- 没有{}的if语句

```
if（条件表达式）
        语句1;
else
        语句2;
语句3;
```

- else语句后面也可以没有大括号，但这时候只有语句2是属于else语句，语句3无论条件表达式是否成立都会执行。

没有{}的if-else语句


```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
if(b!=0)
    cout << a << " / " << b << " = " << a / b << endl;
else
    cout << "除数不应该为0" << endl;
    cout << a << " * " << b << " = " << a * b << endl;
```

```
double a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
if(b!=0)
    cout << a << " / " << b << " = " << a / b << endl;
else
    cout << "除数不应该为0" << endl;
cout << a << " * " << b << " = " << a * b << endl;
```

缩进不会影响程序的运行结果或语义。

条件运算符

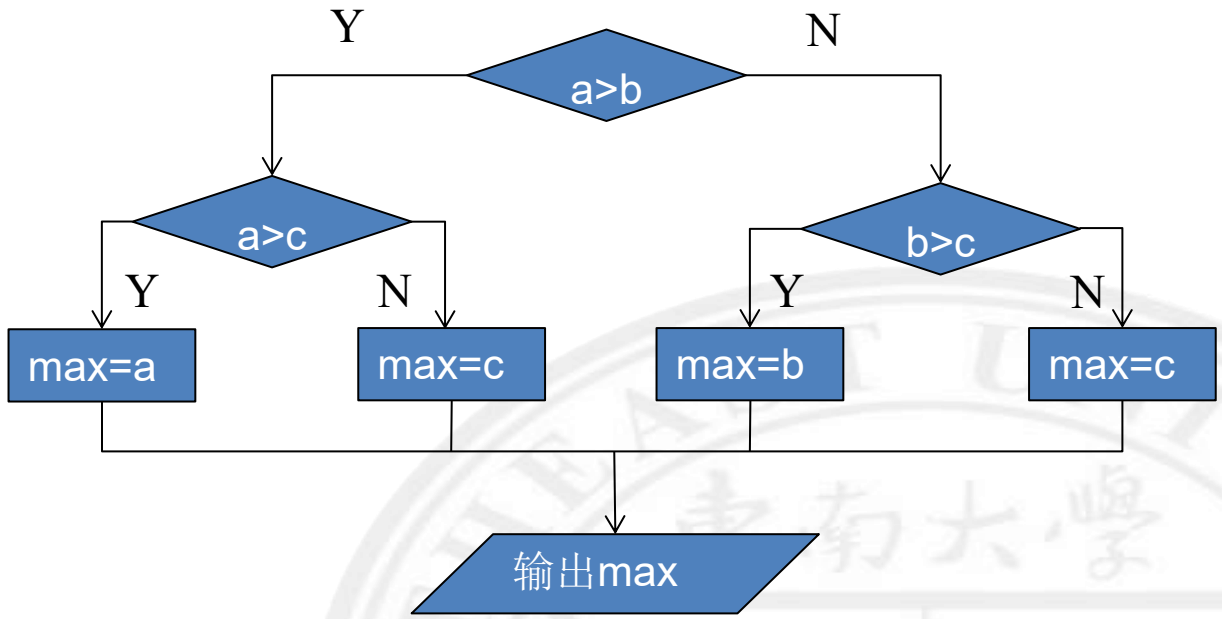
- 条件运算符：？：
- 条件运算表达式：条件？表达式1：表达式2
  - 如果条件成立，执行表达式1
  - 如果条件为假，执行表达式2
- 使用场景：把a和b中大的数赋值给变量max

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

嵌套的判断

找三个数中的最大


```
int a, b, c;
cout << "请输入你要比较大小的三个数字" <<endl;
if(a>b){
    if(a>c){
        max = a;
    }
    else{
        max = c;
    }
}
else{
    if(b>c){
        max = b;
    }
    else{
        max = c;
    }
}
cout << "您输入的最大数字是" << max << endl;
```



嵌套的判断

- 嵌套的判断语句：当判断的条件满足或者不满足的时候要执行的语句是一条if或if-else语句

```
if (math_score >= 60){
    if (science_score >= 90){
        cout << "优秀" << endl;
    }
}
```

```
if (math_score >= 60){
    if (science_score >= 90){
        cout << "优秀" << endl;
    }
    else{
        cout << "良好" << endl;
    }
}
```

if语句和if-else语句


```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
```

=

```
if (math_score >= 60){
    if (science_score >= 90){
        cout << "优秀" << endl;
    }
}
```

```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
    else
        cout << "良好" << endl;
```

=

```
if (math_score >= 60){
    if (science_score >= 90){
        cout << "优秀" << endl;
    }
    else{
        cout << "良好" << endl;
    }
}
```

虽然里面有很多行代码，但嵌套的if语句或者if-else语句依然只算一条语句，更加推荐大家按照带花括号的形式写

缩进

- 缩进并不会改变语句的语法结构

```
if (math_score >= 60)
    if (science_score >= 90)
cout << "优秀" << endl;
```

```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
```

- 缩进格式不能暗示else的匹配

```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
else
    cout << "良好" << endl;
```

```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
    else
        cout << "良好" << endl;
```

else的匹配

- **else只和最近未匹配的if匹配**

```
if (math_score >= 60)
    if (science_score >= 90)
        cout << "优秀" << endl;
else
    cout << "良好" << endl;
```

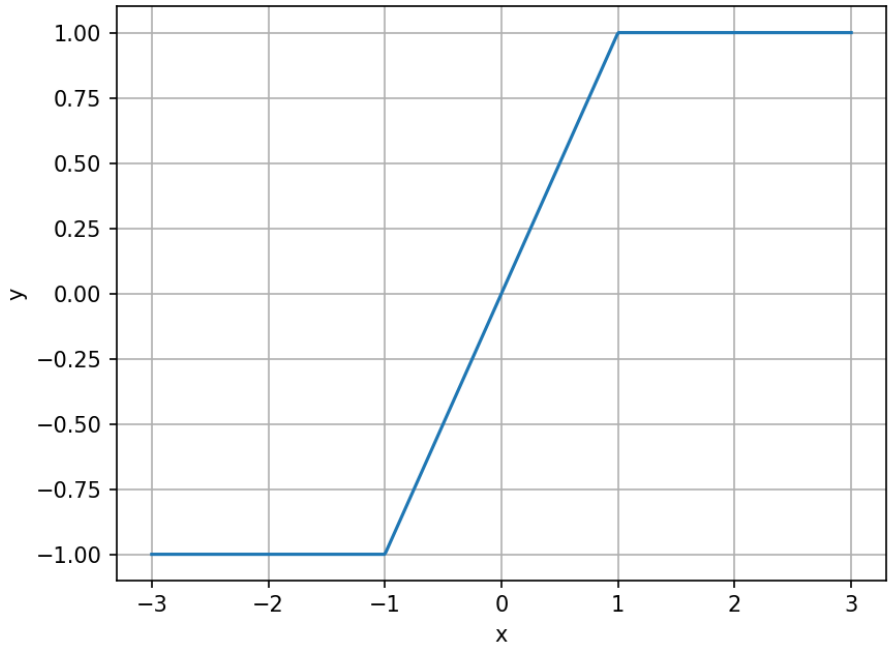
```
if (math_score >= 60){
    if (science_score >= 90)
        cout << "优秀" << endl;
}
else
    cout << "良好" << endl;
```

花括号表示if里面嵌套了一个if语句，花括号里的 if 不会和花括号外的else匹配

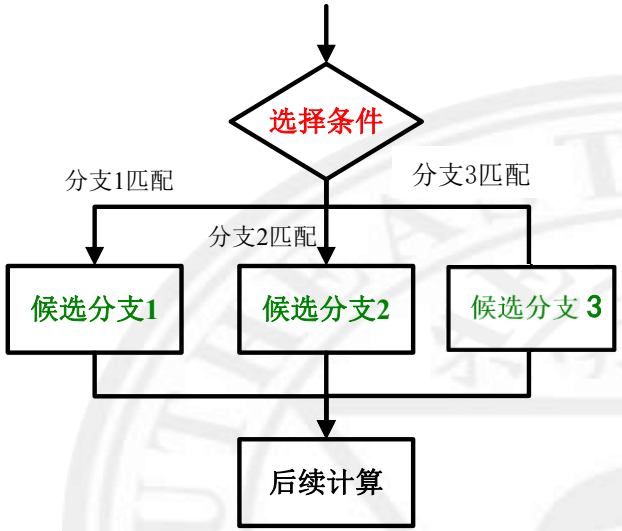
- **建议：在if或else后面总是用{}，即使只有一条语句的时候**

级联的if-else

分段函数



$$y = \begin{cases} -1 & x < -1 \\ x & -1 \le x \le 1 \\ 1 & x > 1 \end{cases}$$



多个分支怎么处理?

级联的if-else语句


```
int x = 0, func = 0;
cin << x;
if(x <= -1){
    func = -1;
}
else if(x > -1 && x < 1){
    func = x;
}
else{
    func = 1;
}
```

级联的if-else语句格式：

```
if（表达式1）{
    语句1;
}
else if（表达式2）{
    语句2;
}
else {
    语句3;
}
```

还可以继续插入else if

从上到下依次判断条件，一旦有一个条件为真，就执行对应的语句块，其他的就跳过不再判断。

级联的if-else和嵌套的if-else语句转换


```
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 {
    语句4;
}
```

=

```
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{
         语句4;
      }
   }
}
```

级联的if-else语句可以改写成嵌套的if-else，但是会随着分支增加，代码不断缩进，不利于阅读。

switch-case语句

switch-case引入

```
if(season==1){
    cout << "Spring" << endl;
}
else if(season==2){
    cout << "Summer" << endl;
}
else if(season==3){
    cout << "Autumn" << endl;
}
else if(season==4){
    cout << "Winter" << endl;
}
else{
    cout << "请输入数字1-4中的一个数字" << endl;
}
```

多重级联的if-else在分支较多的情况下，有时候需要经过多次判断才能找到对应的分支，效率较低。

修改后的代码——switch-case语句


```
if(season==1){
    cout << "Spring" << endl;
}
else if(season==2){
    cout << "Summer" << endl;
}
else if(season==3){
    cout << "Autumn" << endl;
}
else if(season==4){
    cout << "Winter" << endl;
}
else{
    cout << "请输入数字1-4中的一个数字" << endl;
}
```

```
switch(season){
    case 1:
        cout << "Spring" << endl;
        break;
    case 2:
        cout << "Summer" << endl;
        break;
    case 3:
        cout << "Autumn" << endl;
        break;
    case 4:
        cout << "Winter" << endl;
        break;
    default:
        cout << "请输入数字1-4中的一个数字" << endl;
}
```

不需要做多次判断，根据season的结果找到一个入口

switch-case语句解析（1）

```
switch（控制表达式）{
        case 常量1:
                …
                break;
        case 常量2:
                …
                break;
        …
        default:
                …

}
```

- 新增关键字：switch、case、break、default
- 控制表达式只能是整数或字符型的结果，不能是浮点数或字符串
- case后面必须跟常量，常量可以是常数，也可以是常数计算的表达式

switch-case语句解析（2）

```
switch（控制表达式）{
        case 常量1：
                …
                break;
        case 常量2：
                …
                break;
        …
        default:
                …
}
```

可以没有break

- switch语句可以看作是一种基于计算结果的跳转，计算控制表达式的值后，程序会跳转到相匹配的case（分支标号）处。
- 分支标号只是说明switch内部位置的路标，在执行完分支中的最后一条语句后，如果后面没有break，就会顺序执行到下面的case里面去，直到遇到下一个break，或者switch结束为止。

不带break的switch-case语句


```
switch(season){
    case 1:
    case 2:
        cout << "Summer" << endl;
    case 3:
        cout << "Autumn" << endl;
        break;
    case 4:
        cout << "Winter" << endl;
        break;
    default:
        cout << "请输入数字1-4中的一个数字" << endl;
}
```

- season为1时，输出是什么?
- season为2时，输出是什么?
- season为3时，输出是什么?
- season为4时，输出是什么?
- season为5时，输出是什么?

switch-case程序练习

编写程序将一个百分制成绩转换为五分制成绩，转换规则

- 大于等于90分为A；
- 小于90且大于等于80为B；
- 小于80且大于等于70为C；
- 小于70且大于等于60为D；
- 小于60为E