计算机程序设计

刘龙军 副教授

2025年9月15日

二、表达式与分支控制结构

(课本4、5、6章)

运算符和表达式

- 运算符：指定需要进行的计算，又称操作符
  - 算术运算（ + - * / % ）
  - 赋值运算（ = ）
  - 关系运算（ < <= > >= == != ）
  - 逻辑运算（ &&与 ||或 !非 ）
  - 特殊运算（指针、位运算）
- 表达式：通过运算符将常量和变量组合起来生成新的值

基本算术运算符

名称	操作符	表达式示例	结果
加法运算符	+	3+5	8
减法运算符	-	5-2	3
乘法运算符	*	3*5	15
除法运算符	/	5/3	1
模运算符(求余)	%	7%4	3

基本算术运算符

- 注意规则：
- 两个整数相除结果为整数　　1/2=0　　5/2=2
- 整数才可求余，余数的符号与左边数的符号相同。
- 3%2=1　　-3%2=-1　　3%-2=1　　-3%-2=-1　　8%4=0

算术表达式

- 算术表达式由算术运算符和操作数所组成的表达式
  - 正确：(a*x+b)*x
  - 错误：(ax+b)x
- 先乘除、后加减
- 先算左、后算右
- 无敌括弧

赋值运算符

名称	操作符	示例	等价于
简单赋值	=	y = x	
加赋值	+=	y += x	y = y+x
减赋值	-=	y -= x	y = y-x
乘赋值	*=	y *= x	y = y*x
除赋值	/=	y /= x	y = y/x
模赋值	%=	y %= x	y = y%x

赋值运算符

- 赋值运算的结果是被赋值变量赋值后的值

  int a=b=c=0;

- +=、-=、*=、/= 运算符形式看起来更直观，而且执行效率一般也能更高一些

- 注意：赋值运算符左边一定是变量，赋值语句对该变量实施写操作。

稍微复杂的例子*

- x=a+=b*=c-=5
- 等价于

```
c = c-5

b=b*c

a=a+b

x=a
```

稍微复杂的例子*

- int a, b=1, c=2;　a=b=c; a=?
- 区分：
- 运算符结合方向是“自左至右”，即: 先左后右, 例如 a - b + c , b 两侧有 - 和 + 两种运算符的优先级相同, 按先左后右结合方向, b 先与减号结合, 执行a - b 的运算, 再执行加 c 的运算
- 除了自左至右的结合性外, C 语言有 三类运算符 参与运算的结合方向是从右至左（右结合）。 即:赋值运算符, 单目运算符, 条件运算符。

复合的赋值运算符例子

- a+=3　　a=a+3
- x*=y+3　　x=x*(y+3)
- x/=x-4　　x=x/(x-4)
- x+=y　　x=x+y
- i+=j--　　i=i+(j--)
- 赋值表达式　a=b=5 ；　　b=5　　a=5
- "="的结合性为**自右至左**

关系运算符

- （假设y=10, x=5）

名称	操作符	示例	结果
大于	>	y > x	1
等于	==	y == x	0
小于	<	y < x	0
大于等于	>=	y >= x	1
小于等于	<=	y <= x	0
不等于	!=	y != x	1

- 关系表达式结果返回1：表明该关系成立
- 关系表达式结果返回0：表明该关系不成立

关系表达式*

- 关系表达式：用关系运算符将表达式连接起来称为**关系表达式**。其值非真即假。在C++语言中，用非0代表真，用0表示假。关系表达式的结果只有两个，真为1，假为0

```
//rational.cpp
#include <iostream>
// 关系操作符
int main()
{
    using namespace std;
    int y = 10;
    int x = 5;

    int result = (y>x);
    cout<<result<<endl;
    cout.setf(ios_base::boolalpha);
    cout<<(y>x)<<endl<<result<<endl;

    return 0;
}
```



```
1
true
1
```

关系表达式例子

已知：a=2　b=3　c=4

求：

a>2	0	a= ='a'	0	'a'>'A'	1
a>b+c	0	a>'a'	0	b= ='a'+1	0
a= =2	1	b=a= =2	1	c-a= =a	1

大家找别扭

```
int a = 1;
if(a == 0)
   cout<< a;
```

(1)

```
int a = 1;
if(a = 0)
   cout<< a;
```

(2)

```
int a = 0;
if(a == 0)
   cout<< a;
```

(3)

```
int a = 0;
if(a = 0)
   cout<< a;
```

(4)

关系表达式*

- 一定要分清 == 和 =

  == 是比较运算

  = 是赋值运算

```
int a = 0;
if(0 == a)
   cout<< a;
```

```
int a = 0;
if(0 = a)
   cout<< a;
```

逻辑运算符

名称	操作符	表达式示例	说明
与运算	&&	(a>b)&&(b>c)	如果a大于b，并且b大于c，则返回1，否则返回0
或运算	||	(a>b)||(b>c)	如果a大于b，或者b大于c，则返回1，否则返回0
非运算	!	!a	如果a为1，则返回0； 如果a为0，则返回1； 求反并不改变a的值；

逻辑运算符---真值表

西安交通大学 XI'AN JIAOTONG UNIVERSITY

&&

A	B	结果
0	0	0
0	1	0
1	0	0
1	1	1

有0出0，全1出1

A,B同时成立

||

A	B	结果
0	0	0
0	1	1
1	0	1
1	1	1

有1出1，全0出0

A或B有一个成立

!

A	结果
0	1
1	0

有0出1，

有1出0

逻辑运算符

作为条件，所有非0值均为真；作为结果，只有0或1两种。

5>3 && 2 || 8<4-!0

不可写为 1<x<10　应为：1<x && x<10

当前面的表达式可以得出整个表达式的结果时，不必再求后面的表达式。

a&&b&&c 当a为0时，表达式为0，不必求b与c。

a||b||c　当a为1时，表达式为1，不必求b与c。

逻辑运算符---例子

当c=4时，以下的值各多少？

(c=1)&&(c=3)&&(c=5)　　1

(c= =1)||(c= =2) || (c= =5)　　0

(c!=2) && (c!=4) &&(c>=1)&&(c<=5)　　0

运算符的优先级（原理出自词法分析）

- ( ) [ ] -> .
- ! ~ ++ -- + - * & sizeof （类型）
- * / %
- << >>
- < <= > >=
- == !=
- &
- ^
- |
- &&
- ||
- ?:
- = += -= *= /= %= &= ^= |= <<= >>=
- ,

表：几种运算符的优先级

！（逻辑非）	（高）
算术运算符	
关系运算符	
&& 和||	
赋值运算符	（低）

如何让计算按照你期望的进行？

- 记不住怎么办？
  - 用括号来控制运算顺序更直观、方便、并减少出错的概率
  - 先算乘除、后算加减，有括号就先算括号里的
  - 括号太多，有时候不清楚
  - 注意用空格做好分离
  - 实在不行就拆分表达式

自增自减运算符

名称	操作符	示例
自增运算符	++	i++，++i
自减运算符	--	i--，--i

- **自增自减运算符只能针对变量**

i++：在使用i之后，使i的值加1。即先使用，后加。

i--：在使用i之后，使i的值减1。即先使用，后减。

++i：在使用i之前，先使i的值加1。即先加，后使用。

--i：在使用i之前，先使i的值减1。即先减，后使用。

自增自减运算符

- **注意：**

  i++ 等价于 i=i+1, 但是大多数编译器对++操作进行了优化;

  自加(减)运算符可以让程序更简洁，但是在使用过程中容易出错

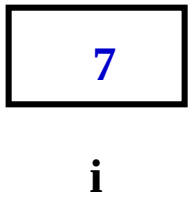
示例

```cpp
//plus.cpp
#include <iostream>
/* ++ -- 操作符 */
int main()
{
    int i = 0;
    int j = i++; //-- operator is the same as ++ operator
    int k= ++i ;

    std::cout<<"j="<<j<<";k="<<k<<"; i="<<i<<std::endl;

    return 0;
}
```

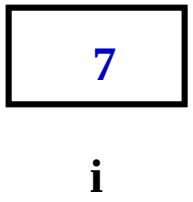

j=0;k=2; i=2

例子----注意语句内与语句外

i=6; i++; i=i+1 i=7



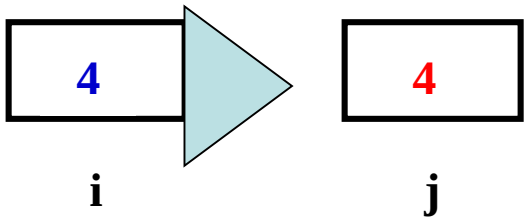
++i; i=i+1 i=7



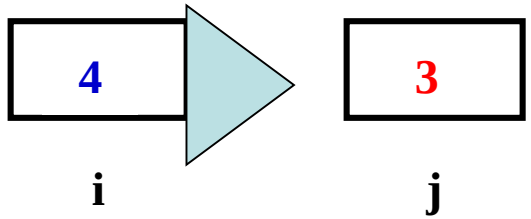
i=6; i－－; i=i－1 i=5

－－i ; i=i－1 i=5

```
int  i, j;
i=3;
j = ++i;
```




```
int  i, j;
i=3;
j = i++;
```



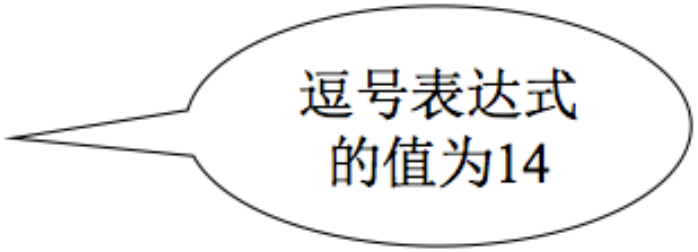
i=4 j=4
++在前, 先运算,后赋值

i=4 j=3
++在后, 先赋值,后运算

逗号运算符/逗号表达式

逗号运算符:将两个表达式连接起来,又称为"顺序求值运算符"

如： 3 +5， 6 + 8



一般形式: 表达式 1 ， 表达式 2

求解过程:

先求解表达式 1 ，再求解表达式 2 。整个逗号表达式的值是表达式 2 的值。

逗号运算符/逗号表达式

逗号表达式的一般形式可以扩展为

表达式 1，表达式 2，表达式 3，……，表达式 n

它的值为表达式 n 的值。

逗号运算符是所有运算符中级别最低的

例： ① x = （a = 3， 6 * 3）

② x = a = 3， 6 *3

赋值表达式，将一个逗号表达式的值赋给 x，x 的值等于 1 8

逗号表达式，包括一个赋值表达式和一个算术表达式，x 的值为 3，整个逗号表达式的值为 18。

逗号运算符和逗号表达式*

表达式1，表达式2，表达式3，…，表达式n
顺序求解，结果为最后一个表达式的值，并且优先级最低。

```
a=(3+4, 5*6,  2+1);          a=3

a=3*3,  a+6,  a+7;           16      a=9

(a=3*5,  a*4),  a+5          20      a=15
```

上节课回顾

- 上机时间，本周三下午2点半到5点半。助教与张玥老师都在，辅导大家编程！按组坐座位

- 运算符、表达式（注意与高中学习区分对比记忆、练习！）
- 算术、赋值、关系、逻辑、自增、自减、逗号表达式；
- 优先级
- 注意各种运算符的组合，理解：双目，单目，三目等
- （关键知识点记忆、练习）
- 第一阶段学完了，复习
- 程序的结构 顺序中的block，学习重点、编程中练习、应用！
- 没太多理论，但要多练，组织练习，针对某个平台找bug，解决之。

程序控制结构

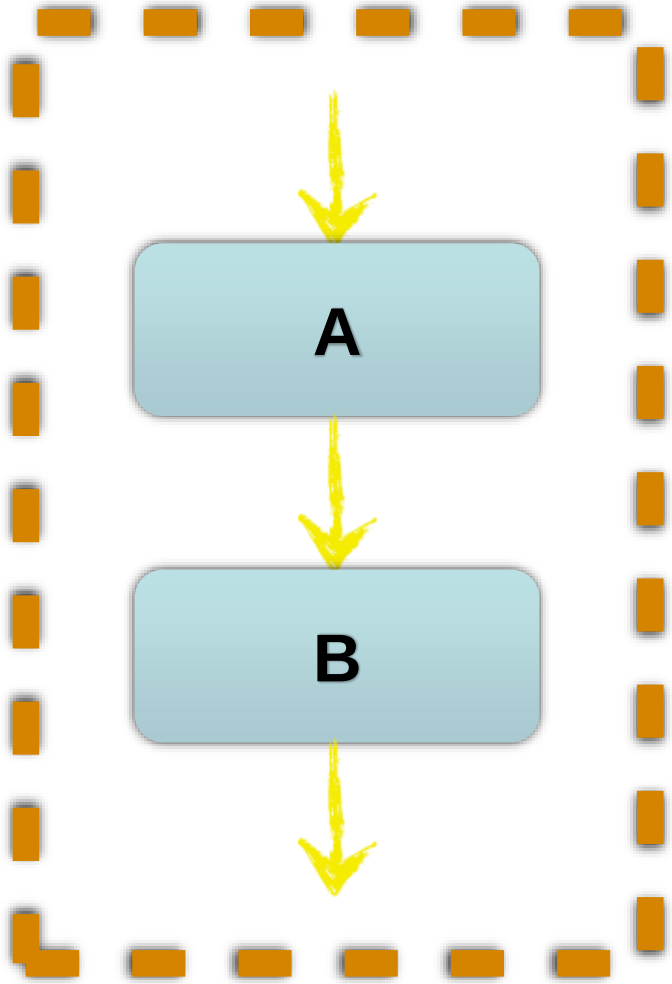
顺序结构

分支结构

循环结构

顺序结构

- 由一组顺序执行的处理块组成，每个程序块可能包含一条或一组语句，完成一项任务
- 顺序结构是最基本的算法结构



程序块（Block）*

- 一对花括号{}括住的若干条语句构成一个程序块,语法上等于单条语句，其后不需要加分号

```
{
    int num=0;
    num++;
    std::cout<<num;
}
```

程序块与变量

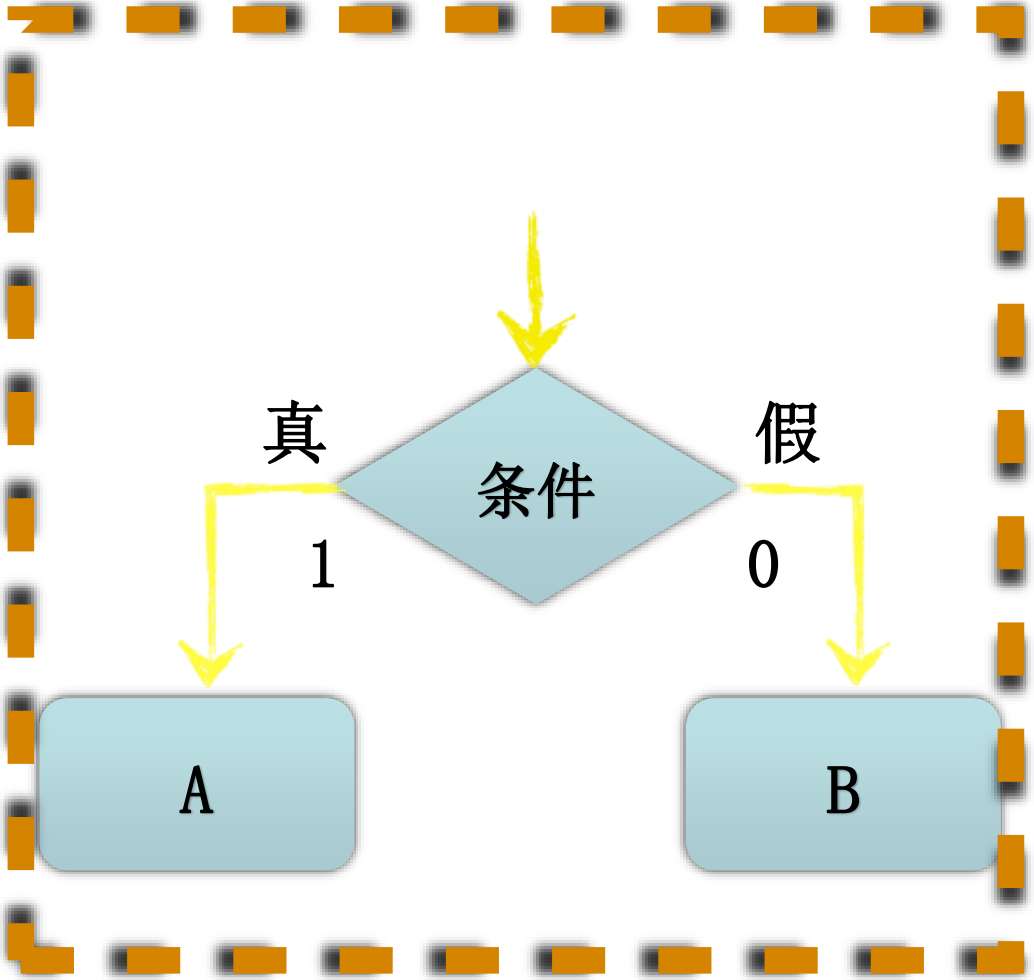
- 同一个语句块内的变量不可同名，不同语句块可以同名
  - 各司其职
  - 尽量不要在下层语句块内定义变量，更不要定义同名变量

```
int main()
{
   int a = 1;

   {
    int a = 2;
    std::cout<< “In the INNER block 1, a =
“<<a<< ‘\n’ ;
   }
   {
    int a = 3;
    std::cout<< “In the INNER block 2, a =
“<<a<< ‘\n’ ;
   }
    std::cout<< “In the OUTTER block, a =
“<<a<< ‘\n’ ;
   return 0;
}
```

```
In the INNER block 1, a = 2
In the INNER block 2, a = 3
In the OUTER block, a = 1
```

选择结构

- 根据某一条件的判断结果，确定程序的流程，即选择哪一个程序分支中的处理块去执行
- 最基本的分支结构是二路选择结构
- 以条件判断为起点，如果判断结果为真，则执行A程序块的操作，否则执行B程序块的操作



选择语句

- if 语句
- switch 语句

If 语句

- 选择结构最常用的形式
- 三种形式的if语句
- if语句是用来判定给定的条件是否满足，根据判定的结果来选择执行不同的操作

```
if(表达式)
        程序块1
else
        程序块2
```

```
if(表达式)
        程序块1
```

```
if(表达式1)
        程序块1
else if(表达式2)
        程序块2
else
        程序块3
```

if语句示例

```
if(表达式)
        语句或程序块
```

```
if(x>y)
      cout<<x;
```

```
if(flag)
      cout<<x;
```

```
if(!flag)
      cout<<x;
```

if~else语句示例

```
if(表达式)
    语句或程序块
else
    语句或程序块
```

```
if(x>y)
            cout<<x;
else
            cout<<y;
```

```
if(x>y)
{
        z=x;
      cout<<z;
}
else
{
        z=y;
        cout<<z;
}
```

if~else if~else语句示例

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

```
if(score>=90)
        level=’A’;
else if(score>=80)
        level=’B’;
else if(score>=60)
        level=’C’;
else
        level=’D’;
```

示例

输入两个实数，按代数值由小到大的顺序输出

```
//compare.cpp
//比较两个整数大小,按代数值由小到大的顺序输出

#include <iostream>
int main( )
{
    using namespace std;
    int num1, num2, tmp;
    cin>>num1>>num2;
    if(num1>num2)
    {
        tmp  = num1;
        num1 = num2;
        num2 = tmp;
    }
    cout<<num1<<endl<<num2<<endl;
    return 0;
}
```

示例2

编程求解一元二次方程

```
//equation.cpp
#include <iostream>
#include<cmath>
//求解一元二次方程 ax^2+bx+c=0
int main()
{
    float a,b,c,x1,x2,delta;
    std::cin>>a>>b>>c;
    delta = b*b - 4*a*c;          // 计算 b^2-4ac 的值
    if (delta >= 0) //开根号
    {
        x1 = (-b + sqrt(delta)) / 2 / a;
        x2 = (-b - sqrt(delta)) / 2 / a;
        std::cout<<"root1=" <<x1<<", root2="<<x2;
    }
    else
        std::cout<<"no root";

    return 0;
}
```

示例3

编程判断某一年份是否闰年

```
/* 判断某一年份是否闰年*/
int main()
{
  int year;

  std::cin>>year;

  if ( (year % 4 == 0 && year % 100 !=0) || year % 400
==0)
      std::cout<<year<< " is a leap year" ;
    else
      std::cout<<year<< " is not a leap year" ;

  return 0;
}
```

换一种写法

体会“代码即注释”的方法

```
//leap2.cpp
#include <iostream>
// 判断某一年份是否闰年,体现代码即注释的境界
int main()
{
    int year=0;
    int leapFlag=0;
    std::cin>>year;
    leapFlag = (year % 4 == 0 && year % 100 !=0) || (year % 400==0);
    if (leapFlag)
        std::cout<<"it is a leap year"<<std::endl;
    else
        std::cout<<"it is not a leap year"<<std::endl;

    return 0;
}
```

请点评

- 以下程序段编译能通过，执行也不出错，但是执行结果不正确，请分析一下哪里错了。还有，既然错了为什么编译能通过呢？

```
#include <iostream>

int main()
{
    int x = -1;
    if (x > 0);
        std::cout<< “x is positive.\n”;

    return 0;
}
```

请点评

- 以下程序段编译能通过，执行也不出错，但是执行结果不正确，请分析一下哪里错了。还有，既然错了为什么编译能通过呢？

```
#include <iostream>

int main()
{
    int x = -1;
    if (x > 0);
        std::cout<< “x is positive.\n”;

    return 0;
}
```

if语句嵌套的使用

- if语句又包含一个或多个if语句称为if语句的嵌套

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

请点评

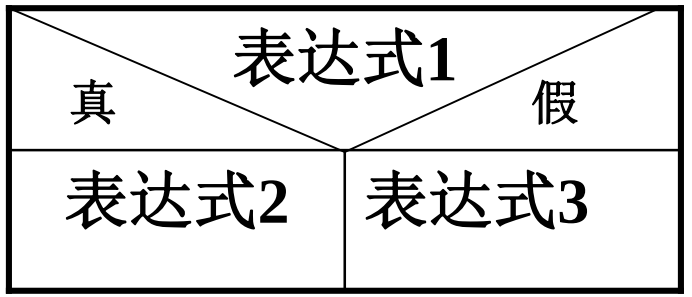
- 配对原则：else总是寻找在它之前的最近未配对的if配对
- 建议用{}确定配对关系

```
#include <iostream> //筛子游戏
/* if else 匹配演示 */
int main()
{
        int i;

   cin>>i;

   if (i != 0)    // 穷举i的范围
    if (i > 0)
        std::cout<< “big\n” ;
   else
       std::cout<< “zero\n” ;

   return 0;
}
```

条件运算符（类分支条件）

- 是C＋＋中的唯一的三目运算符。

表达式1？表达式2 ：表达式3




```
max=a>b?a:b ;   // 求a, b中的大者
```

当 a=2　b=1　a>b为真，表达式的值等于a, max值为2

当 a=1　b=2　a>b为假，表达式的值等于b, max值为2

注意：

1. 条件运算符的优先级比赋值运算符高

   x=(x=3) ? x+2 : x-3

2. 结合方向自左至右　a>b?a:c>d?c:d　　x=5

3. 三个表达式的类型可不同　z=a>b?'A':a+b

条件运算符（类分支条件）

```
x=9, y=6,  z=5;
x=((x+y)%z>=x%z+y%z)?1:0;
cout<<"x= "<<x<<endl;
```

x=0

```
x=1;  y=2; z=3;
x+=y+=z;
cout<<（ z+=x>y?x++:y++）<<endl;
```

y=y+z=5

x=x+5=6

9

条件运算符（类分支条件）

```
void   main(void )
{   int  x=1,y=2,z=3;
    x+=y+=z;
    cout<<x<y?y:x<<endl;
   cout<<x<y?x++:y++<<endl;
   cout<<x<<“,”<<y<<endl;
   cout<<z+=x>y?x++:y++<<endl;
   cout<<y<<“,”<<z<<endl;
   x=3;  y=z=4;
   cout<<(z>=y&&y= =x)?1:0<<endl;
   cout<<z>=y&&y>=x<<endl;
}
```

x	y	z	输出
6	5	3	
6	5	3	6
6	6	3	5
6	6	3	6,6
6	7	9	9
6	7	9	7,9
3	4	4	
3	4	4	0
3	4	4	1

switch语句

- 适用于多分支场景
- 执行过程：
  - 当表达式值为常量表达式1时，执行语句1;
  - 当表达式值为常量表达式2时，执行语句2;
  - ……
  - 当表达式值为常量表达式n时,执行语句n;
  - 否则，执行语句n+1

```
switch(表达式)
{
    case  常量表达式1:
      语句1
        break;
    case  常量表达式2:
    语句2
        break;
  ……
    case  常量表达式n:
        语句n
        break;
    default:
        语句n+1
        break;
}
```

switch与if的区别

- 在某些情况下，switch语句会使代码更清晰
- 有时候编译器会对switch语句进行整体优化，使它比等价的if/else语句所生成的指令效率更高

switch 与 if

```
switch(表达式)
{
    case  常量表达式1:
                语句1
        break;
    case  常量表达式2:
                        语句2
        break;
  ......
    case  常量表达式n:
            语句n
        break;
    default:
            语句n+1
        break;
}
```

=

```
if(常量表达式1)
      语句1
else if(常量表达式2)
   语句2
......
else if(常量表达式n)
   语句n
else
         语句n+1
```

switch示例

- 编程将百分制成绩转换成5级记分制（编程三步走）

请点评*

```
#include <iosteam>

int main()
{
  int score;

  scanf("%d", &score);
  switch(score)
  {
      case score >= 90:
          cout<<"A\n"; break;
      case score >= 80:
          cout<<"B\n"; break;
      case score >= 70:
          cout<<"C\n"; break;
      case score >= 60:
          cout<<"D\n"; break;
      default:
          cout<<"E\n"; break;
  }

  return 0;
}
```

case后面跟常量表达式

正确的写法*

```
//switch.cpp
……
    using namespace std;
    int score, level;
   cin>>score;
  level = score/10;
  switch(level)
  {
          case 10:
      case 9:
         cout<<"A\n";
          break;
      case 8:
         cout<<"B\n";
          break;
      case 7:
         cout<<"C\n";
           break;
      case 6:
         cout<<"D\n";
          break;
      default:
         cout<<"E\n";
        break;
  }
  ….
```

注意switch语句中的break

- **执行过程：**
  - 当表达式值为常量表达式1时，执行语句1到语句n+1;
  - 当表达式值为常量表达式2时，执行语句2到语句n+1;
  - ……
  - 当表达式值为常量表达式n时,执行语句n到语句n+1;
  - 否则，执行语句n+1

```
switch  （表达式）
{
    case  常量表达式1:
        语句1
    case  常量表达式2:
        语句2
    ......
    case  常量表达式n:
        语句n
    default: 语句n+1
}
```

试试去掉break*

```
switch(level)
{
    case 10:
    case 9:
        cout<<"A\n";
     case 8:
        cout<<"B\n";
    case 7:
       cout<<"C\n";
    case 6:
      cout<<"D\n";
    default:
        cout<<"E\n";
}
```

break的用法

```
…
    level = score/10;
    switch(level)
    {
        case 10:
        case 9:
        case 8:
        case 7:
        case 6:
            cout<<"Pass\n";
            break;
        default:
            cout<<"Fail\n";
             break;
    }
    return 0;
…
```

本章结束！