

选择结构与逻辑表达式

北京理工大学计算机学院
金旭亮

三种典型的程序代码执行流程

顺序执行

按条件选择一条分支执行

选择结构

在特定场景中反复执行特定
语句

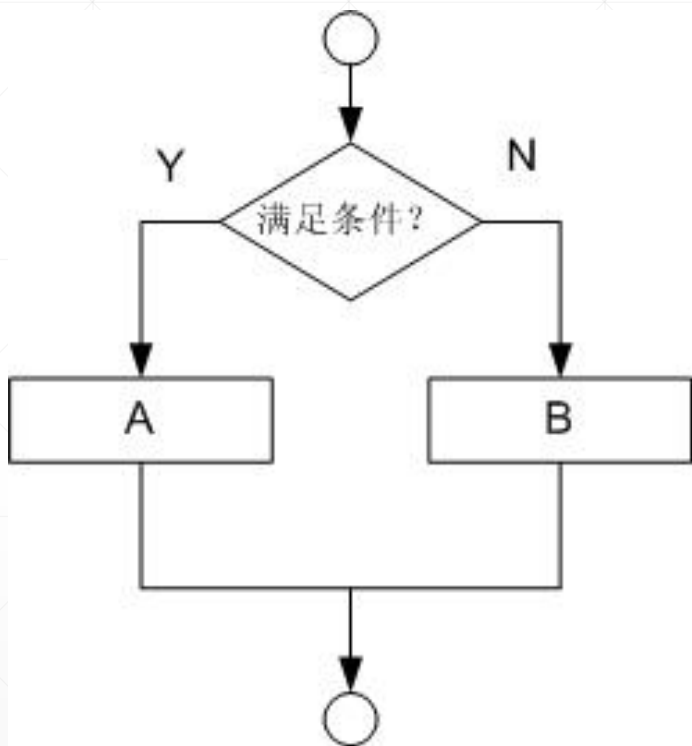
循环结构

程序流程图

在结构化程序设计中，经常使用一种称为“程序流程图”的示意图来表示程序的执行流程。

虽然在面向对象时代，流程图用得没有以前多了，但由于其拥有直观性强的优点，仍然被广泛使用，比如在行业应用软件系统开发时，使用流程图表达业务流程的处理步骤，几乎所有人都能看得懂。

使用 if/else 语句实现程序中的选择结构



编程语言中的选择结构



```
if ( 条件 )  
{  
    //执行语句块A  
}  
else  
{  
    //执行语句块B  
}
```

C#中的 if/else 语句

选择结构示例（伪代码）

使用**逻辑表达式**表达条件，“==”是逻辑判等运算符

“if”和“else”是C#中用于实现选择结构的两个“**关键字 (Keyword)**”。

if (**天气晴朗 == true**)

我们去爬香山;

else

我们呆在家里看电视;

条件满足 (true) 时, 执行这句

条件不满足 (false) 时, 执行这句

- **逻辑表达式**通常用于表示某种条件是否得到满足。
- 当程序运行时, 计算机解析逻辑表达式, 会得到一个值, 在C#中, 这个值只有“**true (真)**”或“**false (假)**”两种情况。表达式值为“true”, 表示条件满足, 为“false”, 表示条件不满足。
- 在if语句中, 基于逻辑表达式执行的结果执行特定的分支。

用于构建逻辑表达式的运算符

运算符	说明
>	大于，实例：“5>10”，值为“false”
<	小于
==	等于，实例：“9== 100”，值为“false”
>=	大于等于
<=	小于等于
!=	不等于，实例：“100 != 101”，值为“true”

注意：“=>”不是“等于大于”，它在C#程序中有特定的含义，它表示一个Lambda表达式。我们将在后面的课程中对它进行介绍。

选择结构的嵌套

```
if (条件)
{
    if (条件)
        { 语句块A }
    else
        { 语句块B }
}
else
{
    if (条件)
        { 语句块C }
    else
        { 语句块D }
}
```

嵌套在条件语句内部的条件语句，整个语句被当作一个语句块处理，可以看成是一个整体

除非使用“{”和“}”为语句划分了块，否则，else总是与它最近的if配套。

弄错“if”和“else”的配套关系，有可能会带来严重的问题，推荐加上足够的“{”和“}”给与明确区分。

逻辑表达式的组合

使用“&&”组合两个表达式，只有两个条件都满足（都为“true”）时，整个条件才算满足，可以将“&&”与汉语中的“**并且**”对应上。

if (天气晴朗 && 两人都有空)

张三李四一块去爬香山;

else

张三李四呆在家里看电视;

三种用于组合逻辑表达式的运算符

&& (And /与)、**||** (Or/或)、**!** (Not/非)

逻辑表达式的组合解析结果

* 假设 A 和 B 都是一个逻辑表达式，对其使用与、或、非组合运算的规律如下：

① $A \ \&\& \ B$ ：只有 A 和 B 都为 true，结果才为 true

② $A \ || \ B$ ：只要 A 和 B 中有一个为 true，结果就为 true

③ $!A$ ：结果总是与 A “相反”，比如 A 为 true，则 !A 为 false

Not 和 Or

“Not” 逻辑运算符实例

```
if (!天气下雨)
    我们去爬香山;
else
    我们呆在家里看电视;
```

“Or” 逻辑运算符实例

```
if (我发工资了 || 有人请客)
    我就去大饭店吃顿好的;
else
    我回宿舍泡康师傅方便面吃;
```

动手动脑

编写一个程序，在运行时让用户输入两个数，程序判断出大小，输出较大的数

提示：

- 可以使用 `Console.ReadLine()` 方法读取用户输入的值
- 用户输入的内容是字符串，为了比较大小，需要将其转换为数值类型再进行比较

扩充训练：

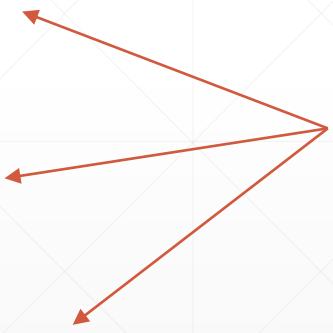
- 尝试着输入非法的无法转换为数字的字符串，看看你的程序会发生什么？
- 要解决非法数据的问题，可以使用 **异常捕获**，这块内容，将在后继课程介绍

多值选择结构

C#提供了**switch**结构，用于对同一个变量的多个值进行判断。

```
switch (intValue)
{
    case 0:
        CaseZero();
        break;
    case 1:
        CaseOne();
        break;
    default:
        CaseOthers();
        break;
}
```

程序运行时，依据“intValue”变量的真实值，选择一个分支执行。如果都不匹配，则执行“默认分支”（即名字为“default”的那个）



为了保证一个值执行一个分支，每个分支后都应该加一个break;语句（最后一个分支可略）

多值选择结构实例

```
static void DoYouPass()
{
    Console.WriteLine("请输入你的考试成绩:");
    string UserInput = Console.ReadLine();
    int score = int.Parse(UserInput);
    switch (score/10)
    {
        case 10:
        case 9:
            Console.WriteLine("你是学霸");
            break;
        case 8:
            Console.WriteLine("你的成绩还不错");
            break;
        case 7:
            Console.WriteLine("你的成绩可归于'路人甲'一类");
            break;
        case 6:
            Console.WriteLine("哥们，小心点！");
            break;
        default:
            Console.WriteLine("你的成绩？杯具了！");
            break;
    }
}
```

用户输入考试成绩，程序给出：优、良、中、及格和不及格的相应评价。

如果有两个值执行相同的代码，可以将其“合并”，写在一起。

动手试一试：

去掉所有的break;，你得到了什么结果？

学了马上用，完成两道编程题.....

1

编写一个程序，让用户输入一个今天的气温，程序给出“太热了”、“太冷了”、“真舒服”等结论。

2

编写一个最简单的计算器：

- (1) 用户输入两个数字
- (2) 用户输入“+”、“-”、“*”、“/”四个运算符之一
- (3) 程序输出计算结果