[**白盒测试**](javascript:;)的测试方法有代码检查法、静态结构分析法、静态质量度量法、逻辑覆盖法、基本路径测试法、域测试、符号测试、Z路径覆盖、程序变异。

　　其中运用最为广泛的是基本路径测试法。

　　基本路径测试法是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。

　　设计出的测试用例要保证在测试中程序的每个可执行语句至少执行一次。

　　在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例。包括以下4个步骤和一个工具方法：

　　1. 程序的控制流图：描述程序控制流的一种图示方法。

　　2. 程序圈复杂度：McCabe复杂性度量。从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必须的测试用例数目的上界。

　　3. 导出测试用例：根据圈复杂度和程序结构设计用例数据输入和预期结果。

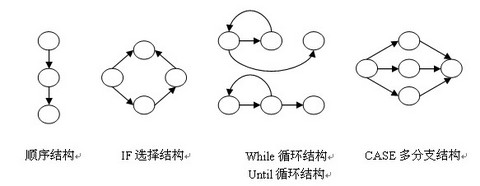
　　4. 准备测试用例：确保基本路径集中的每一条路径的执行。

　　工具方法：

　　图形矩阵：是在基本路径测试中起辅助作用的软件工具，利用它可以实现自动地确定一个基本路径集。

　　程序的控制流图：描述程序控制流的一种图示方法。

　　圆圈称为控制流图的一个结点，表示一个或多个无分支的语句或源程序语句



流图只有二种图形符号：

　　图中的每一个圆称为流图的结点，代表一条或多条语句。

　　流图中的箭头称为边或连接，代表控制流

　　任何过程设计都要被翻译成控制流图。

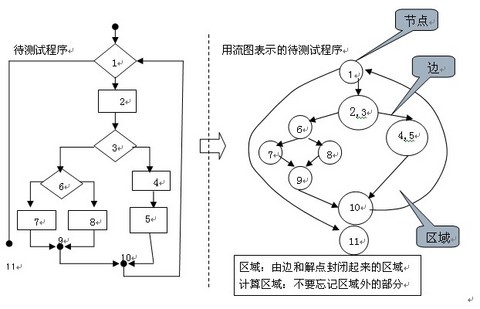
　　如何根据程序流程图画出控制流程图?

　　在将程序流程图简化成控制流图时，应注意：

　　n 在选择或多分支结构中，分支的汇聚处应有一个汇聚结点。

　　n 边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域。

　　如下页图所示



n 如果判断中的条件表达式是由一个或多个逻辑运算符 (OR, AND, NAND, NOR) 连接的复合条件表达式，则需要改为一系列只有单条件的嵌套的判断。

　　例如：

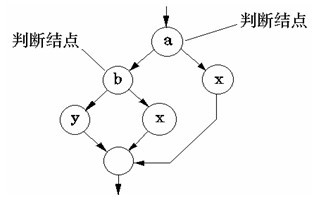
　　1 if a or b

　　2 x

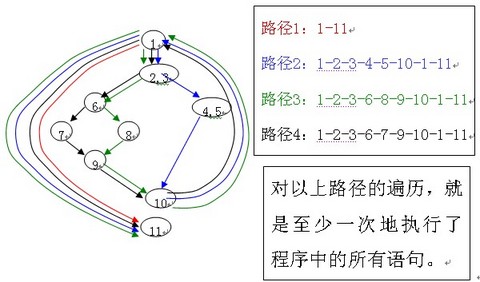
　　3 else

　　4 y

　　对应的逻辑为：



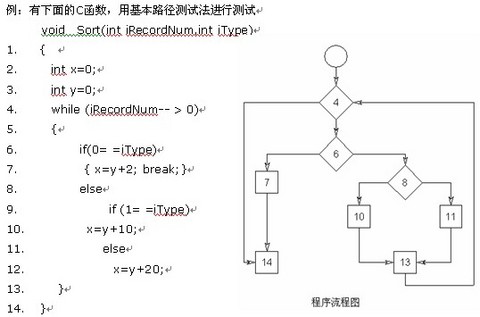
独立路径：至少沿一条新的边移动的路径



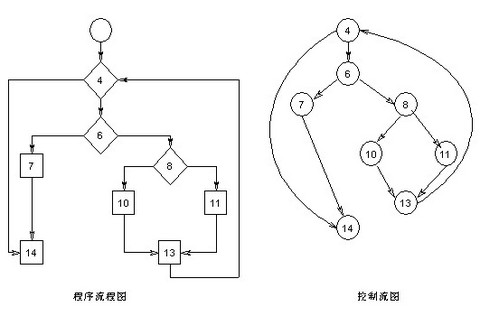
基本路径测试法的步骤：

　　o 第一步：画出控制流图

　　流程图用来描述程序控制结构。可将流程图映射到一个相应的流图(假设流程图的菱形决定框中不包含复合条件)。在流图中，每一个圆，称为流图的结点，代表一个或多个语句。一个处理方框序列和一个菱形决测框可被映射为一个结点，流图中的箭头，称为边或连接，代表控制流，类似于流程图中的箭头。一条边必须终止于一个结点，即使该结点并不代表任何语句(例如：if-else-then结构)。由边和结点限定的范围称为区域。计算区域时应包括图外部的范围。



画出其程序流程图和对应的控制流图如下



第二步：计算圈复杂度

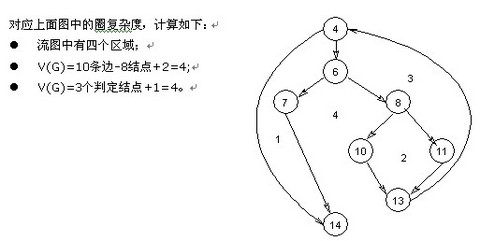
　　圈复杂度是一种为程序逻辑复杂性提供定量测度的软件度量，将该度量用于计算程序的基本的独立路径数目，为确保所有语句至少执行一次的测试数量的上界。独立路径必须包含一条在定义之前不曾用到的边。

　　有以下三种方法计算圈复杂度：

　　流图中区域的数量对应于环型的复杂性;

　　给定流图G的圈复杂度V(G)，定义为V(G)=E-N+2，E是流图中边的数量，N是流图中结点的数量;

　　给定流图G的圈复杂度V(G)，定义为V(G)=P+1，P是流图G中判定结点的数量。



第三步：导出测试用例

　　根据上面的计算方法，可得出四个独立的路径。(一条独立路径是指，和其他的独立路径相比，至少引入一个新处理语句或一个新判断的程序通路。V(G)值正好等于该程序的独立路径的条数。)

　　ü 路径1：4-14

　　ü 路径2：4-6-7-14

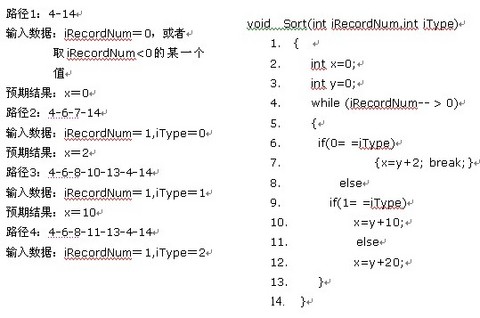
　　ü 路径3：4-6-8-10-13-4-14

　　ü 路径4：4-6-8-11-13-4-14

　　根据上面的独立路径，去设计输入数据，使程序分别执行到上面四条路径。

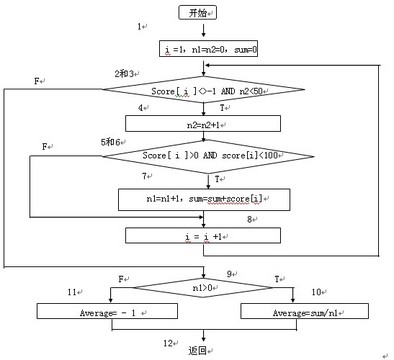
　　o 第四步：准备测试用例

　　为了确保基本路径集中的每一条路径的执行，根据判断结点给出的条件，选择适当的数据以保证某一条路径可以被测试到，满足上面例子基本路径集的测试用例是：

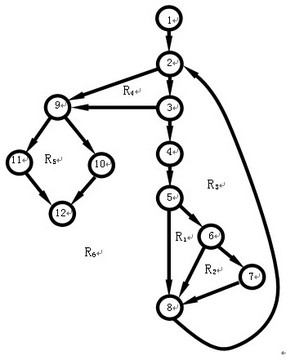


举例说明：

　　例：下例程序流程图描述了最多输入50个值(以–1作为输入结束标志)，计算其中有效的学生分数的个数、总分数和平均值。



步骤1：导出过程的流图。



步骤2:确定环形复杂性度量V(G)：

　　1)V(G)= 6 (个区域)

　　2)V(G)=E–N+2=16–12+2=6

　　其中E为流图中的边数，N为结点数;

　　3)V(G)=P+1=5+1=6

　　其中P为谓词结点的个数。在流图中，结点2、3、5、6、9是谓词结点。

　　步骤3：确定基本路径集合(即独立路径集合)。于是可确定6条独立的路径：

　　路径1：1-2-9-10-12

　　路径2：1-2-9-11-12

　　路径3：1-2-3-9-10-12

　　路径4：1-2-3-4-5-8-2…

　　路径5：1-2-3-4-5-6-8-2…

　　路径6：1-2-3-4-5-6-7-8-2…

　　步骤4：为每一条独立路径各设计一组测试用例，以便强迫程序沿着该路径至少执行一次。

　　1)路径1(1-2-9-10-12)的测试用例：

　　score[k]=有效分数值，当k < i ;

　　score[i]=–1, 2≤i≤50;

　　期望结果：根据输入的有效分数算出正确的分数个数n1、总分sum和平均分average。

　　2)路径2(1-2-9-11-12)的测试用例：

　　score[ 1 ]= – 1 ;

　　期望的结果：average = – 1 ，其他量保持初值。

　　3)路径3(1-2-3-9-10-12)的测试用例：

　　输入多于50个有效分数，即试图处理51个分数，要求前51个为有效分数;

　　期望结果：n1=50、且算出正确的总分和平均分。

　　4)路径4(1-2-3-4-5-8-2…)的测试用例：

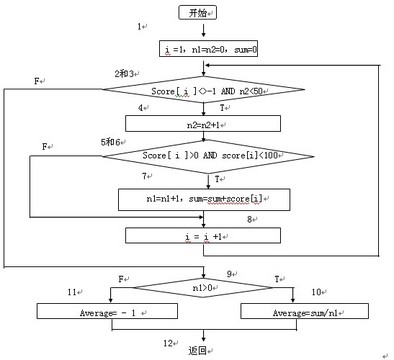
　　score[i]=有效分数，当i<50;

　　score[k]<0， k< i ;

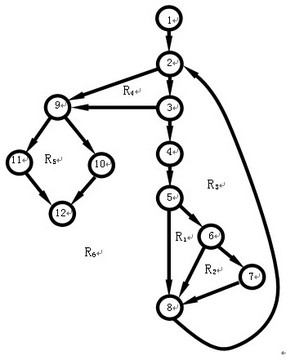
　　期望结果：根据输入的有效分数算出正确的分数个数n1、总分sum和平均分average。

举例说明：

　　例：下例程序流程图描述了最多输入50个值(以–1作为输入结束标志)，计算其中有效的学生分数的个数、总分数和平均值。



步骤1：导出过程的流图。



步骤2:确定环形复杂性度量V(G)：

　　1)V(G)= 6 (个区域)

　　2)V(G)=E–N+2=16–12+2=6

　　其中E为流图中的边数，N为结点数;

　　3)V(G)=P+1=5+1=6

　　其中P为谓词结点的个数。在流图中，结点2、3、5、6、9是谓词结点。

　　步骤3：确定基本路径集合(即独立路径集合)。于是可确定6条独立的路径：

　　路径1：1-2-9-10-12

　　路径2：1-2-9-11-12

　　路径3：1-2-3-9-10-12

　　路径4：1-2-3-4-5-8-2…

　　路径5：1-2-3-4-5-6-8-2…

　　路径6：1-2-3-4-5-6-7-8-2…

　　步骤4：为每一条独立路径各设计一组测试用例，以便强迫程序沿着该路径至少执行一次。

　　1)路径1(1-2-9-10-12)的测试用例：

　　score[k]=有效分数值，当k < i ;

　　score[i]=–1, 2≤i≤50;

　　期望结果：根据输入的有效分数算出正确的分数个数n1、总分sum和平均分average。

　　2)路径2(1-2-9-11-12)的测试用例：

　　score[ 1 ]= – 1 ;

　　期望的结果：average = – 1 ，其他量保持初值。

　　3)路径3(1-2-3-9-10-12)的测试用例：

　　输入多于50个有效分数，即试图处理51个分数，要求前51个为有效分数;

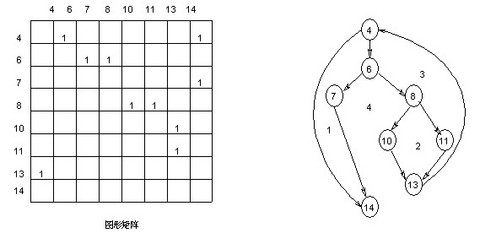
　　期望结果：n1=50、且算出正确的总分和平均分。

　　4)路径4(1-2-3-4-5-8-2…)的测试用例：

　　score[i]=有效分数，当i<50;

　　score[k]<0， k< i ;

　　期望结果：根据输入的有效分数算出正确的分数个数n1、总分sum和平均分average。



  连接权为“1”表示存在一个连接，在图中如果一行有两个或更多的元素“1”，则这行所代表的结点一定是一个判定结点，通过连接矩阵中有两个以上(包括两个)元素为“1”的个数，就可以得到确定该图圈复杂度的另一种算法。

**2.3.3 基本路径测试法**

基本路径测试法是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行的路径集合，从而设计测试用例的方法。设计出的测试用例要保证在测试中程序的每个执行语句至少执行一次。

基本路径测试法的重点内容如下：

程序的控制流图：描述程序控制流的一种图示方法。

程序环形复杂度：McCabe复杂性度量。从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必需的测试用例数目的上界。

导出测试用例：根据圈复杂度和程序结构设计用例数据输入和预期结果。

准备测试用例：确保基本路径集中的每一条路径的执行。

程序控制流图（可简称流图）是对程序流程图进行简化后得到的，它突出表示程序控制流的结构。程序控制流图是描述程序控制流的一种方式，其要点如下：

1）图形符号：圆圈代表一个结点，表示一个或多个无分支的语句或源程序语句。

2）程序控制流边和点圈定的部分叫做区域。当对区域计数时，图形外的一个部分也应记为一个区域。

3）判断语句中的条件为复合条件（即条件表达式由一个或多个逻辑运算符连接的逻辑表达式（a and b）)时，则需要改变复合条件的判断为一系列只有单个条件的嵌套的判断。

结点由带标号的圆圈表示，可代表一个或多个语句、一个处理框序列和一个条件判定框（假设不包含复合条件）。控制流线由带箭头的弧或线表示，可称为边，它代表程序中的控制流。为了满足路径覆盖，必须首先确定具体的路径以及路径的个数。我们通常采用控制流图的边（弧）序列和节点列表示某一条具体路径。

路径测试就是从一个程序的入口开始，执行所经历的各个语句的完整过程。任何关于路径分析的测试都可以叫做路径测试。

完成路径测试的理想情况是做到路径覆盖，但对于复杂性高的程序要做到所有路径覆盖（测试所有可执行路径）是不可能的。

在不能实现所有路径覆盖的前提下，如果某一程序的每一个独立路径都被测试过，那么可以认为程序中的每个语句都已经检验过了，即达到了语句覆盖。这种测试方法就是通常所说的基本路径测试方法。

基本路径测试方法是在控制流图的基础上，通过分析控制结构的环形复杂度，导出执行路径的基本集，再从该基本集设计测试用例。基本路径测试方法包括以下4个步骤：

1）画出程序的控制流图。

2）计算程序的环形复杂度，导出程序基本路径集中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必需的测试用例数目的上界。

3）导出基本路径集，确定程序的独立路径。

4）根据步骤3）中的独立路径，设计测试用例的输入数据和预期输出。

程序控制流图如图2-4所示。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20101014/2237510.jpg> |
| （点击查看大图）图2-4 程序控制流程 |

程序的环形复杂度也称为圈复杂度，它是一种为程序逻辑复杂度提供定量尺度的软件度量。

将环形复杂度用于基本路径方法，可以提供程序基本集的独立路径数量，确保所有语句至少执行一次测试。独立路径是指程序中至少引入了一个新的处理语句集合或一个新条件的程序通路，包括一组以前没有处理的语句或条件的一条路径。通常环形复杂度以图论为基础，提供软件度量。可用如下方法来计算环形复杂度：

1）控制流图中区域的数量对应于环形复杂度。

2）给定控制流图G的环形复杂度V（G），其定义为：

V（G）= E-N+2

其中，E是控制流图中边的数量，N是控制流图中的节点数量。

计算环形复杂度如图2-5所示。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20101014/2237511.jpg> |
| 图2-5 计算环形复杂度图 |

计算结果为节点数量N=8，

导出边的数量E=10（用①、②、③、④、⑤、⑥、⑦、⑧、⑨、⑩编号表示），

V（G）= E-N+2=10-8+2= 10（条边）- 8（个节点）+ 2 = 4，

导出独立路径用路径1、路径2、路径3、路径4编号表示。

**四、基本路径测试**

　　如果把覆盖的路径数压缩到一定限度内，例如，程序中的循环体只执行零次和一次，就成为基本路径测试。它是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。

设计出的测试用例要保证在测试中，程序的每一个可执行语句至少要执行一次。

1.程序的控制流图

　　控制流图是描述程序控制流的一种图示方法。基本控制构造的图形符号如图10-4-7所示。符号○称为控制流图的一个结点，一组顺序处理框可以映射为一个单一的结点。控制流图中的箭头称为边，它表示了控制流的方向，在选择或多分支结构中分支的汇聚处，即使没有执行语句也应该有一个汇聚结点。边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域。

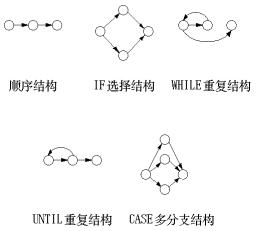
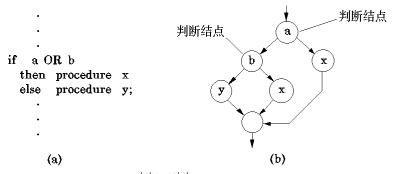
[](http://www.51testing.com/batch.download.php?aid=23416)

图 10-4-7　控制流图的各种图形符号

　　如果判定中的条件表达式是复合条件时，即条件表达式是由一个或多个逻辑运算符（OR，AND，NAND，NOR）连接的逻辑表达式，则需要改复合条件的判定为一系列只有单个条件的嵌套的判定。例如对应图 10-4-8（a）的复合条件的判定，应该画成如图 10-4-8（b）所示的控制流图。条件语句ifaORb中条件a和条件b各有一个只有单个条件的判定结点。

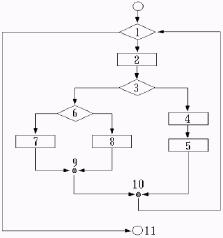
[](http://www.51testing.com/batch.download.php?aid=23417)

　　2.计算程序环路复杂性

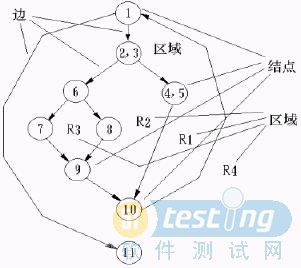
　　进行程序的基本路径测试时，程序的环路复杂性给出了程序基本路径集合中的独立路径条数，这是确保程序中每个可执行语句至少执行一次所必需的测试用例数目的上界。所谓独立路径，是指包括一组以前没有处理的语句或条件的一条路径。如在图10-4-9（b）所示的控制流图中，一组独立的路径是

　　path1：1-11  
　　path2：1-2-3-4-5-10-1-11  
　　path3：1-2-3-6-8-9-10-1-11  
　　path4：1-2-3-6-7-9-10-1-11

　　路径path1，path2，path3，path4组成了图 10-4-9（b）所示控制流图的一个基本路径集。只要设计出的测试用例能够确保这些基本路径的执行，就可以使得程序中的每个可执行语句至少执行一次，每个条件的取真和取假分支也能得到测试。基本路径集不是唯一的，对于给定的控制流图，可以得到不同的基本路径集。

[](http://www.51testing.com/batch.download.php?aid=23418)

（a）程序流程图

[](http://www.51testing.com/batch.download.php?aid=23419)

（b）控制流图

图 10-4-9　程序流程图与对应的控制流图

　　通常环路复杂性可用以下3种方法求得。

　　（1） 将环路复杂性定义为控制流图中的区域数。

　　（2） 设E为控制流图的边数，N为图的结点数，则定义环路复杂性为V（G）＝E－N＋2。

　　（3） 若设P为控制流图中的判定结点数，则有V（G）＝P＋1。

　　因为图 10-4-9（b）所示控制流图有4个区域。其环路复杂性为4。它是构成基本路径集的独立路径数的上界。可以据此得到应该设计的测试用例的数目。

　　3.导出测试用例

　　利用逻辑覆盖方法生成测试用例，确保基本路径集中每条路径的执行。