控制流测试

Control Flow Test

目录

CONTENTS

目录

CONTENTS

目录

CONTENTS

4.1 基本概念

1、控制流（程序单元控制流）：程序指令执行的顺序，条件语句触发默认、顺序执行控制流的改变

2、程序路径(Program Path)：在程序中，从进入点到退出点的指令执行顺序

3、程序路径可以通过输入和期望输出来表示：
一个特定的输入值会使得一段特定的程序路径被执行。该程序按设定好的路径进行计算，产生期望的输出

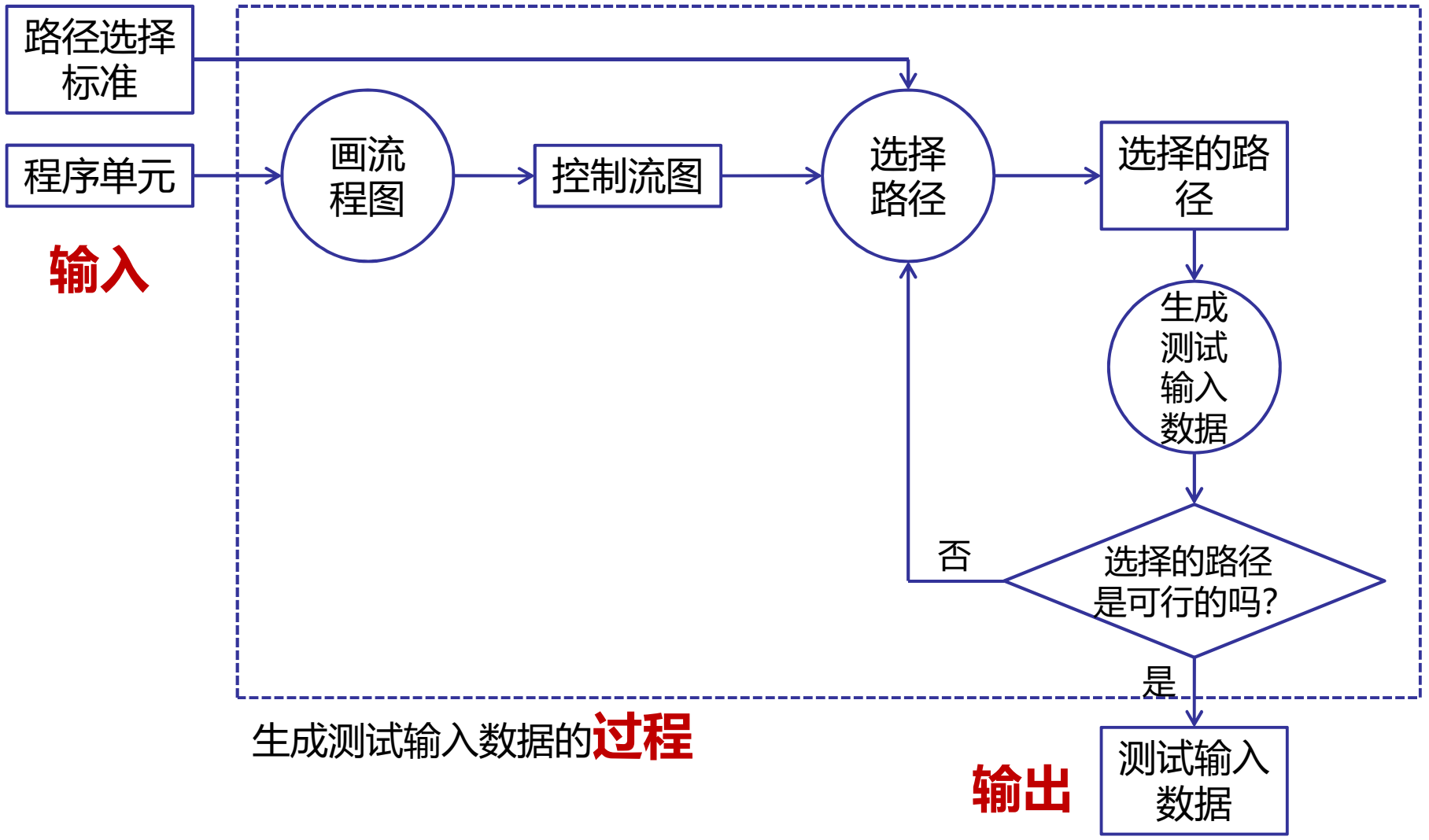
4.1 一些基本概念

4、从结构上来看，**路径**是程序单元的一个语句序列

5、从语义上来看，**路径**是程序单元的一个执行实例

6、控制流测试：从源代码生成测试用例，针对一个程序单元，**核心思想**是在程序单元中适当地选择一些执行路径，观察选择的路径是否产生了预期的结果

4.2 控制流测试数据生成基本流程



4.2 控制流测试基本流程

1、输入：程序单元的源代码和**一组路径选择标准**（4.5）是生成测试数据过程的输入项

2、控制流图的生成（4.3）：控制流图（**CFG**）是程序单元的详细图形表示

3、路径选择：从**CFG**中选择路径以满足路径选择标准

4.2 控制流测试基本流程

4、测试输入数据的生成：一个路径是可以执行的，当且仅当程序单元的一个特定的输入实例，使得沿着路径上所有条件语句，根据控制流取true或false。重要的是确定给定路径的输入特定值，使得这个路径得以执行

5、路径的可行性测试：是否满足路径选择的标准

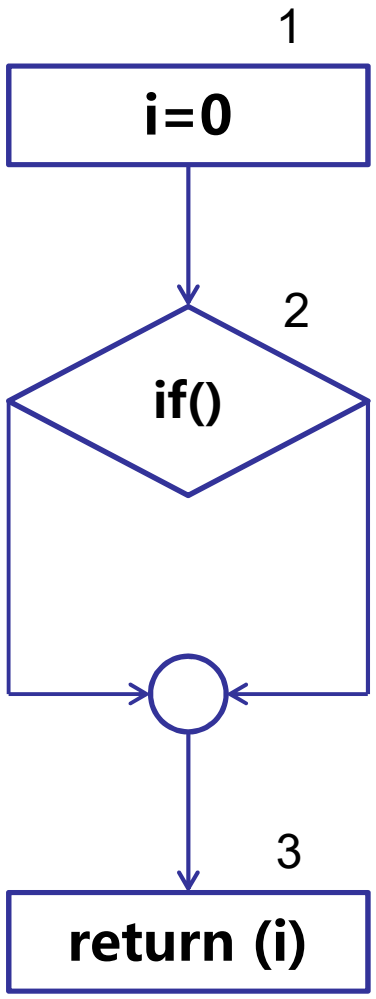
目录

CONTENTS

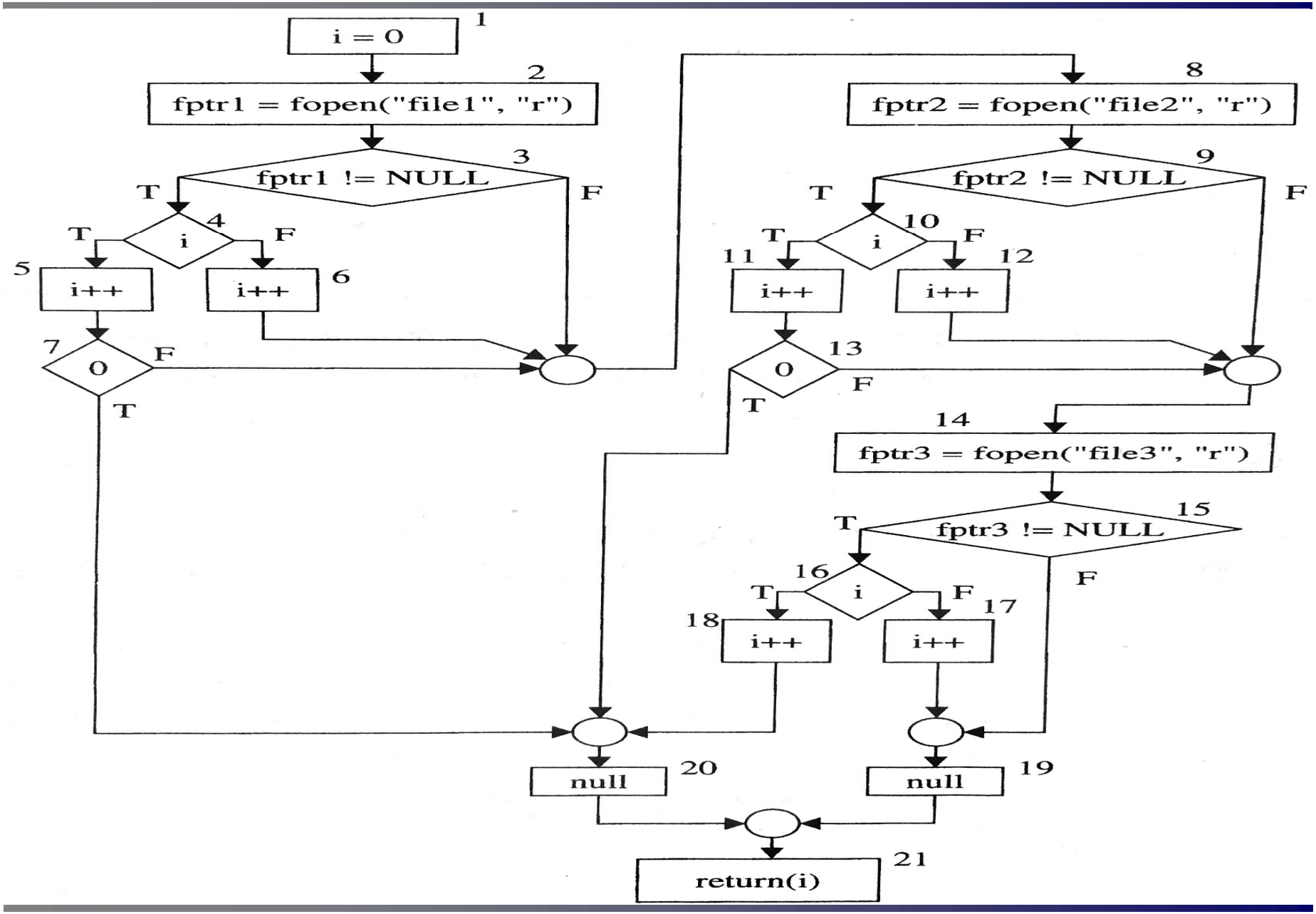
4.3 控制流图（eg1）

```
FILE * fptr1,* fptr2,*fptr3;
int openfiles(){
    int i=0;
    if(
        (((fptr1=fopen( “file1” ,” r” ))!=NULL)&&(i++)&&(0)) ||
        (((fptr2=fopen( “file2” ,” r” ))!=NULL)&&(i++)&&(0)) ||
        (((fptr3=fopen( “file3” ,” r” ))!=NULL)&&(i++)) );
    return (i);
}
```

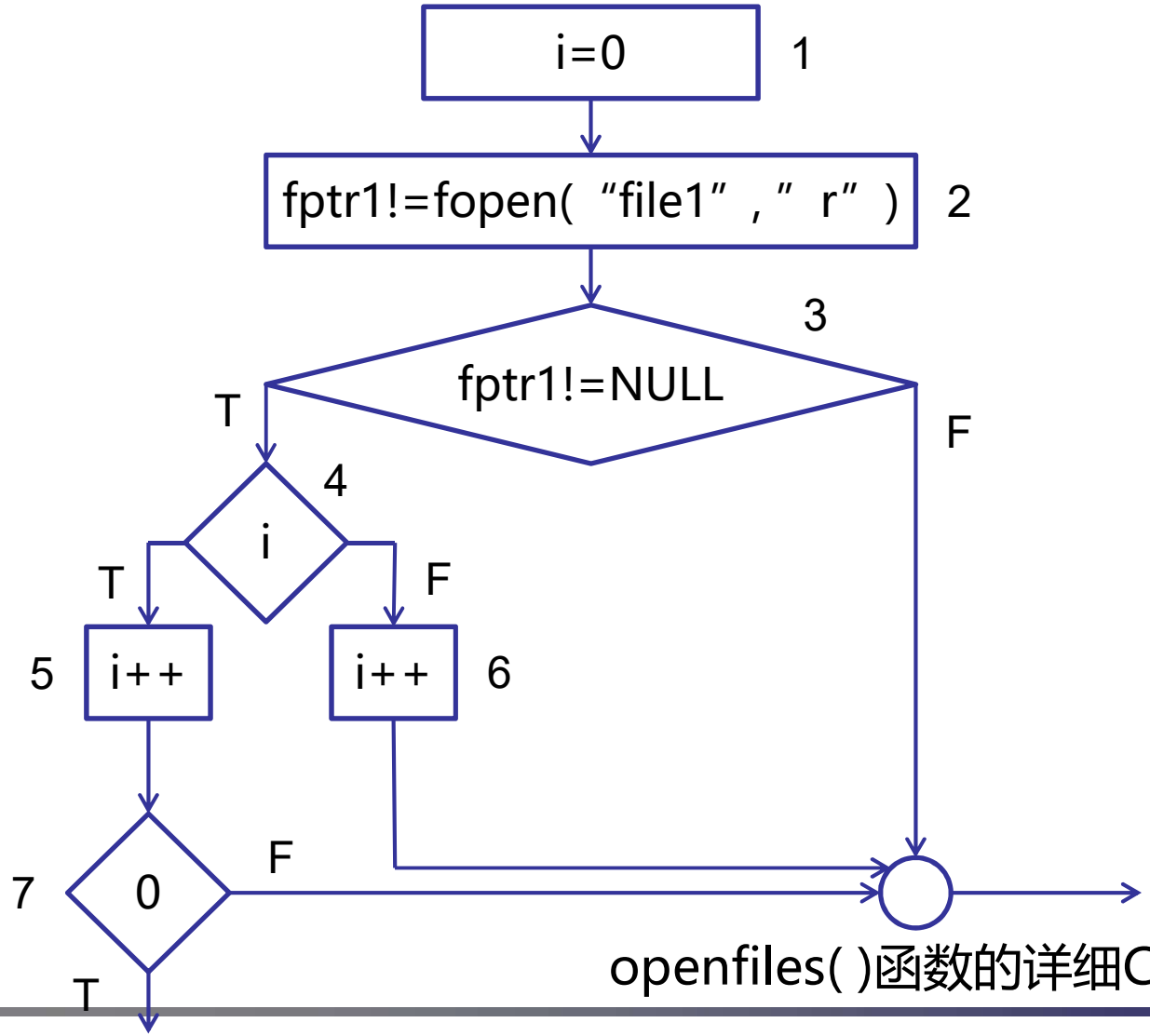
4.3 控制流图（eg1）



openfiles( )函数的抽象CFG表示


i = 0
1
fptr1 = fopen("file1", "r")
2
fptr1 != NULL
3
T
F
i
4
T
F
i++
5
i++
6
0
7
F
T
fptr2 = fopen("file2", "r")
8
fptr2 != NULL
9
T
F
i
10
T
F
i++
11
i++
12
0
13
F
T
fptr3 = fopen("file3", "r")
14
fptr3 != NULL
15
T
F
i
16
T
F
i++
18
i++
17
null
20
null
19
return(i)
21

4.3 控制流图（eg1）

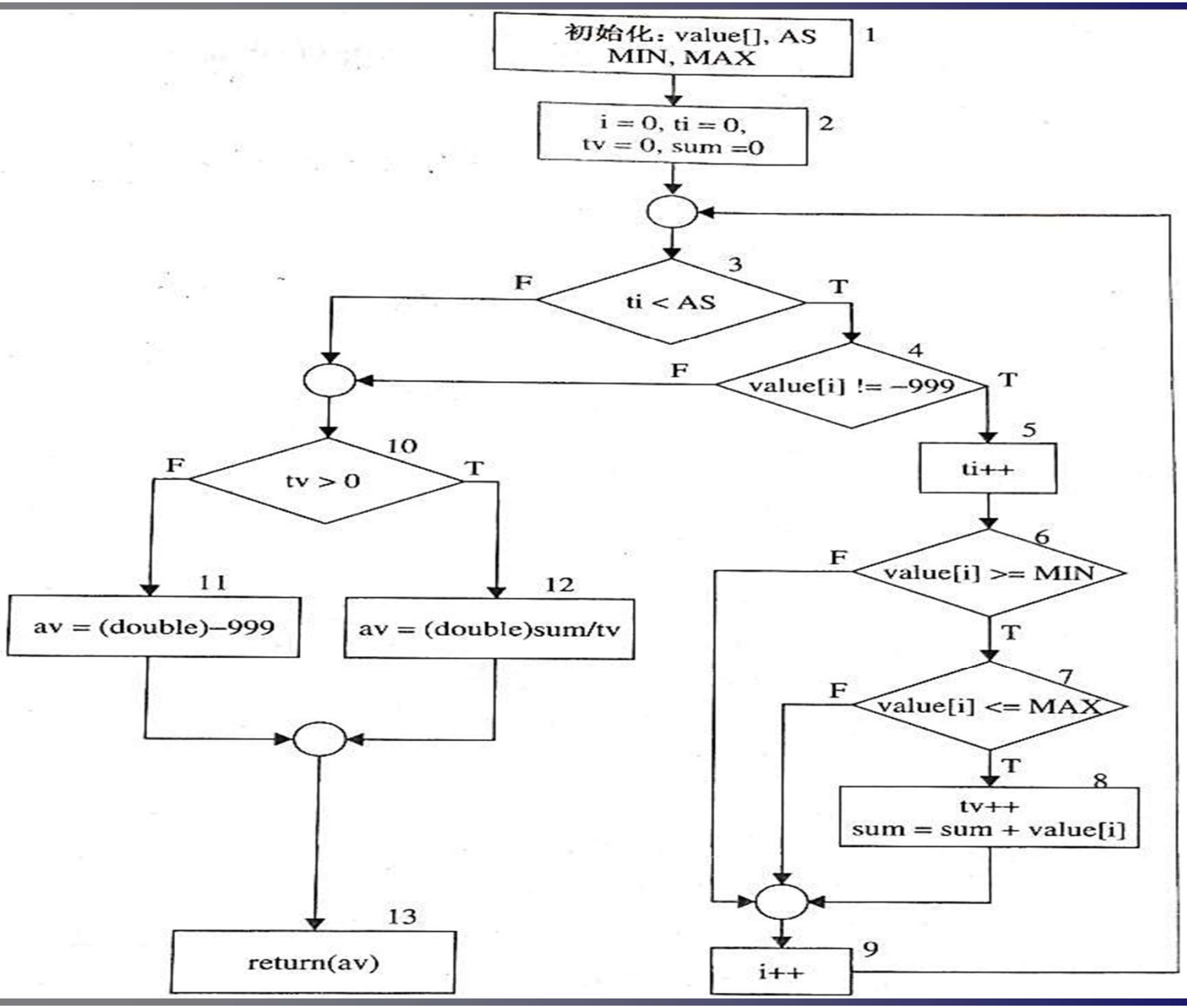


openfiles( )函数的详细CFG表示（部分）

4.3 控制流图（eg2）

```
Public static double ReturnAverage(int value[], int AS, int MIN, int MAX){
    int i, ti, tv, sum; double av;
    i=0; ti=0; tv=0; sum=0;
    while(ti<AS && value[i]!=-999) {
        ti++;
        if(value[i]>=MIN && value[i]<=MAX) {
            tv++;  sum=sum+value[i];
        }
        i++;
    }
    if(tv>0)
        av=(double)sum/tv;
    else
        av=(double) -999;
    return (av);
}
```

功能： ReturnAverage计算输入数组在[MIN,MAX]范围之间的所有数值的平均值。数组的最大长度为AS。但是数组的大小可能小于AS，在这种情况下输入的结束以-999表示


初始化：value[], AS
MIN, MAX
1
i = 0, ti = 0,
tv = 0, sum =0
2
3
F
ti < AS
T
4
F
value[i] != –999
T
5
ti++
6
F
value[i] >= MIN
T
7
F
value[i] <= MAX
T
8
tv++
sum = sum + value[i]
9
i++
10
F
tv > 0
T
11
av = (double)–999
12
av = (double)sum/tv
13
return(av)

目录

CONTENTS

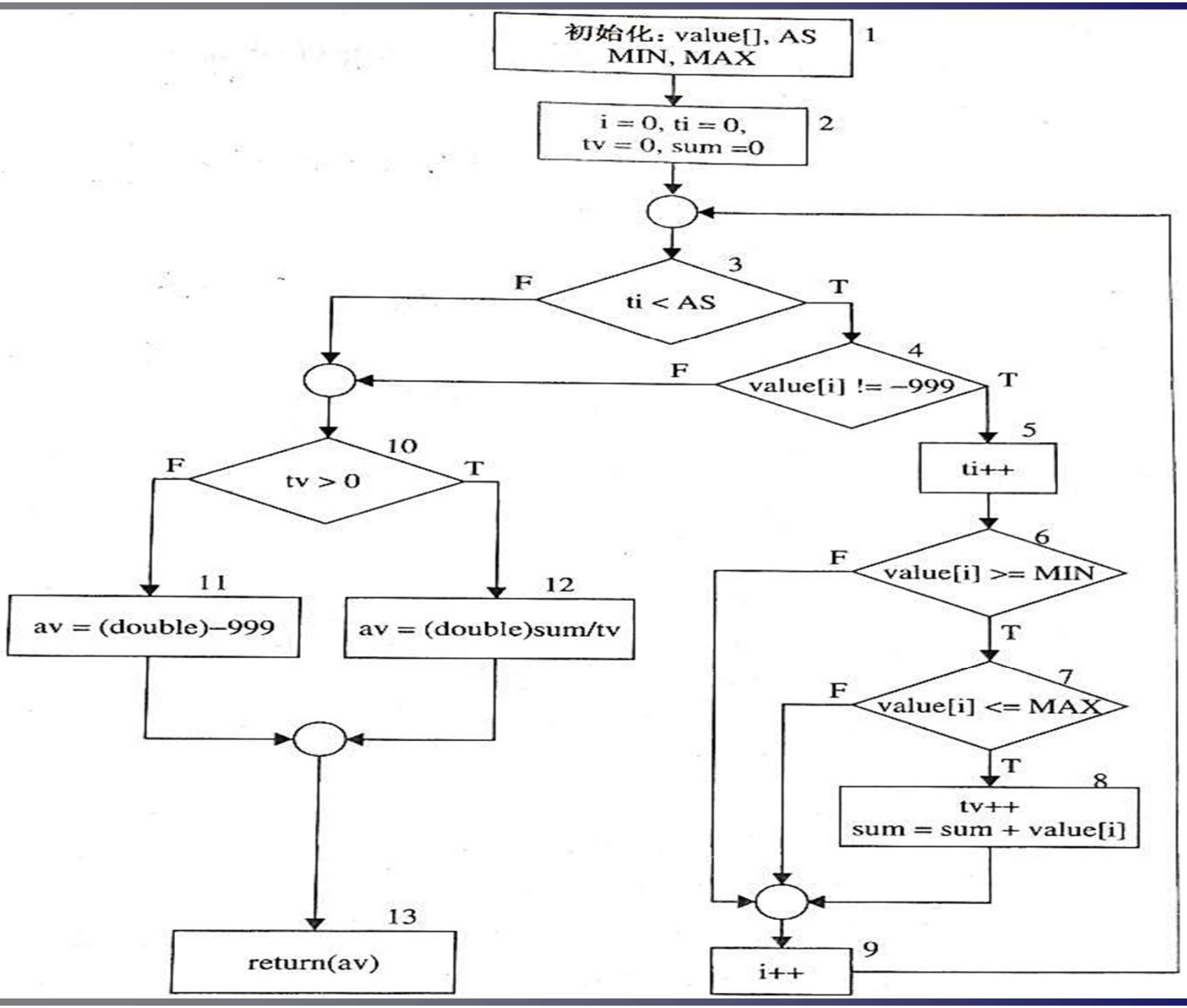
4.4 控制流图中的路径

路径代表了输入节点到输出节点之间的计算和判断节点（指明控制是通过其true分支还是false分支）的序列

4.4 控制流图中的路径

表4.1 ReturnAverage函数CFG的路径示例

Path1: 1-2-3(F)-10(T)-12-13
Path2: 1-2-3(F)-10(F)-11-13
Path3: 1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
Path4: 1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13


初始化：value[], AS
MIN, MAX
1
i = 0, ti = 0,
tv = 0, sum =0
2
3
F
ti < AS
T
4
F
value[i] != –999
T
5
ti++
6
F
value[i] >= MIN
T
7
F
value[i] <= MAX
T
8
tv++
sum = sum + value[i]
9
i++
10
F
tv > 0
T
11
av = (double)–999
12
av = (double)sum/tv
13
return(av)

目录

CONTENTS

4.5 路径选择标准

对于只有少数路径的程序单元来说，执行所有的路径是可实现的

但是**对于有大量路径的程序单元**，

执行所有的路径是不可能的

4.5 路径选择标准

如何选取路径进行测试？**（基于已定义标准进行路径选择的优点）**

1、所有的程序构造（声明、布尔条件和返回）都必须至少执行一次

2、对于重复执行同样的路径，不再生成测试输入。特殊情况：如果执行的程序路径存在更新系统状态的可能，那么多次执行相同的路径或许会不一样

3、可以判断出程序路径是否被测试过

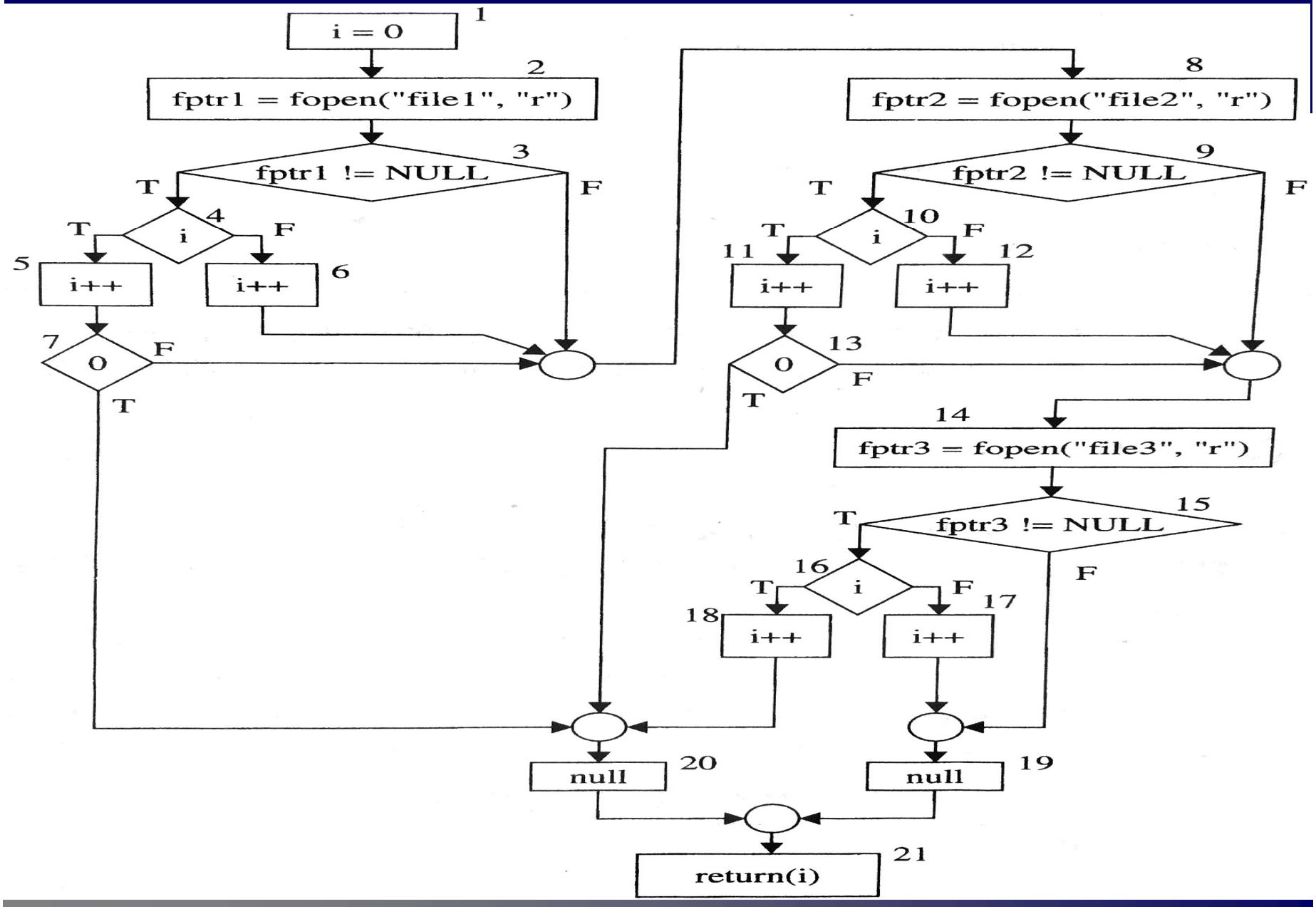
4.5 路径选择标准

全路径覆盖原则

选取程序中所有可能（可执行）的路径（简单程序可以，复杂程序很难）

例：openfile()的输入与路径（部分）

输入	路径
<no,no,no>	1-2-3(F)-8-9(F)-14-15(F)19-21
<yes,no,no>	1-2-3(T)-4(F)-6-8-9(F)-14-15(F)-19-21
<yes,yes,yes>	1-2-3(T)-4(F)-6-8-9(T)-10(T)-11-13(F)-14-15(T)-16(T)-18-20-21

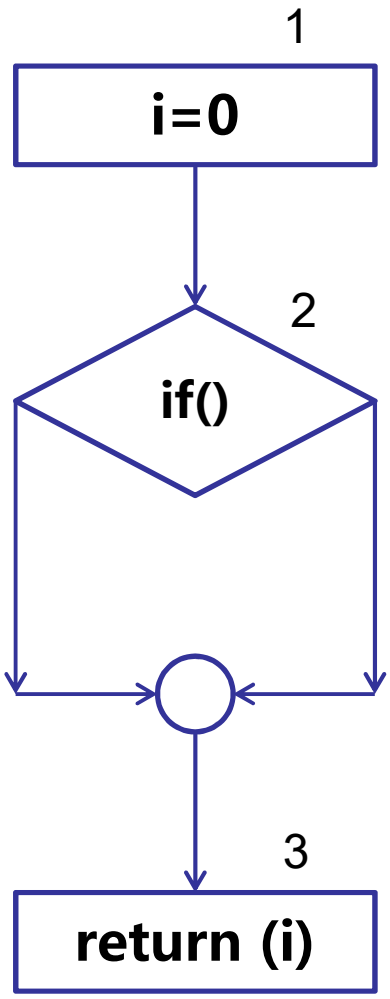

i = 0
1
fptr1 = fopen("file1", "r")
2
fptr1 != NULL
3
T
F
i
4
T
F
i++
5
i++
6
0
7
F
T
fptr2 = fopen("file2", "r")
8
fptr2 != NULL
9
T
F
i
10
T
F
i++
11
i++
12
0
13
F
T
fptr3 = fopen("file3", "r")
14
fptr3 != NULL
15
T
F
i
16
T
F
i++
18
i++
17
null
20
null
19
return(i)
21

4.5 路径选择标准

语句覆盖原则

选取路径（一般是多条路径），使得程序单元中所有的语句都至少被执行一次（覆盖CFG中的所有节点）。完全语句覆盖是**最弱**的覆盖标准，**任何测试套件，如果没有达到语句覆盖，则认为是不可接受的**

路径选取规则：选择较短的路径；选择越来越长的路径，如果有必要可以执行循环多次；选择任意长度的“复杂”的路径



openfile函数的抽象CFG表示

4.5 路径选择标准

分支覆盖原则

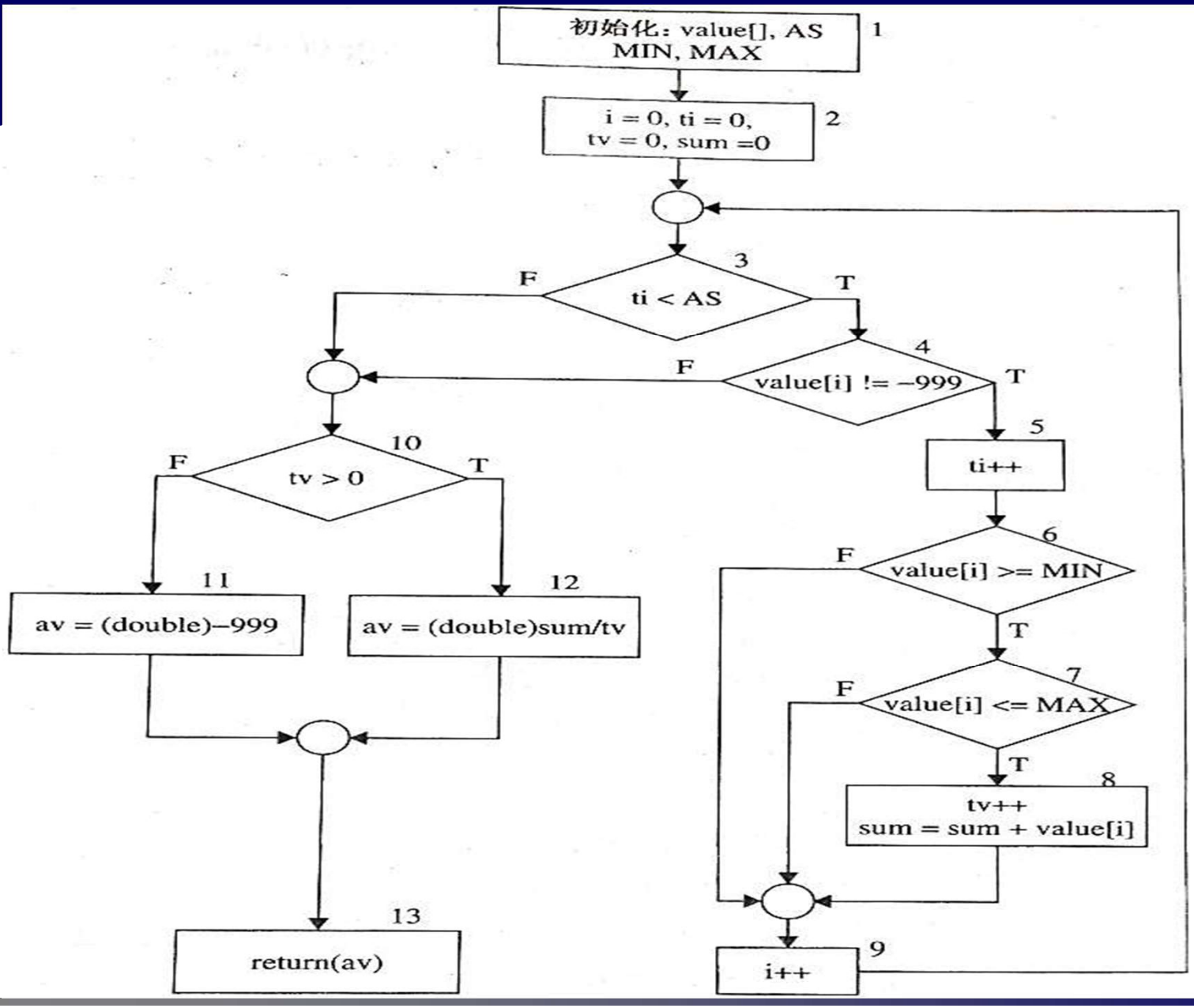
选择一些路径使得**每一条分支**至少被一条路径包含。即每一个条件取true和false至少各一次

4.5 路径选择标准

分支覆盖原则-例（图4.7中CFG语句覆盖和分支覆盖路径）

输入	路径
SCPath1	1-2-3(F)-10(F)-11-13
SCPath2	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13

输入	路径
BCPath1	1-2-3(F)-10(F)-11-13
BCPath2	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
BCPath3	1-2-3(F)-4(F)-10(F)-11-13
BCPath4	1-2-3(T)-4(T)-5-6(F)-9-3(F)-10(F)-11-13
BCPath5	1-2-3(T)-4(T)-5-6(T)-7(F)-9-3(F)-10(F)-11-13

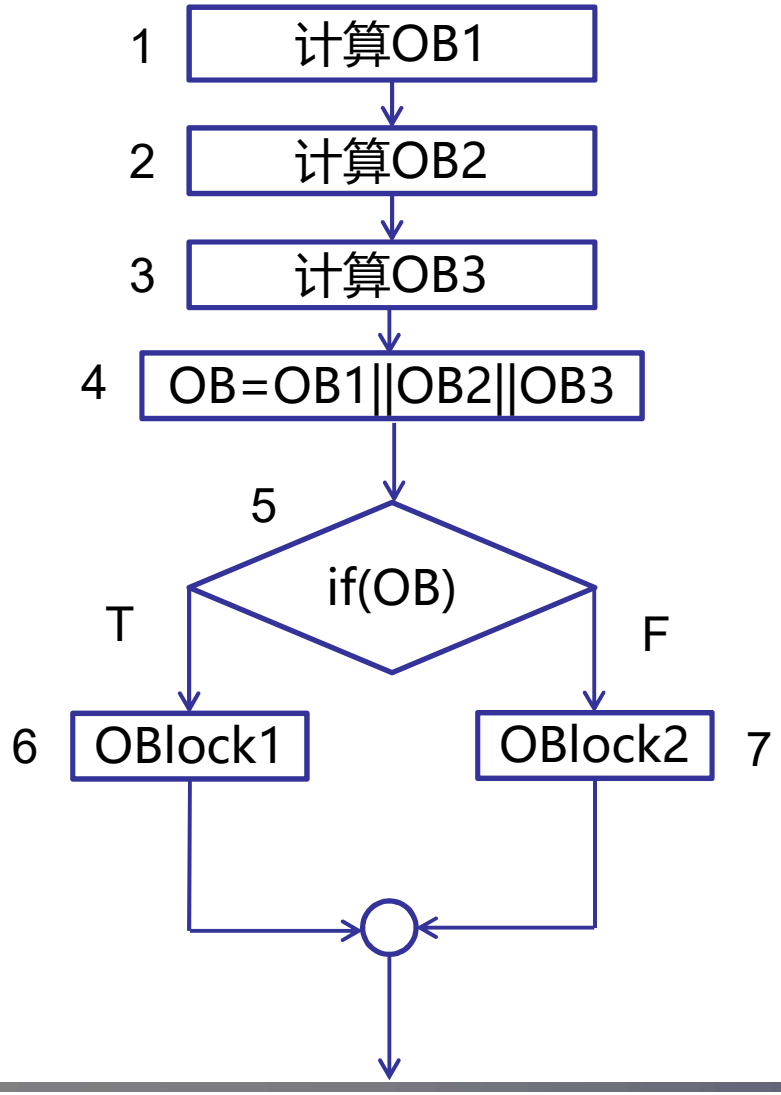

初始化：value[], AS
MIN, MAX
1
i = 0, ti = 0,
tv = 0, sum =0
2
3
F
ti < AS
T
4
F
value[i] != –999
T
5
ti++
6
F
value[i] >= MIN
T
7
F
value[i] <= MAX
T
8
tv++
sum = sum + value[i]
9
i++
10
F
tv > 0
T
11
av = (double)–999
12
av = (double)sum/tv
13
return(av)

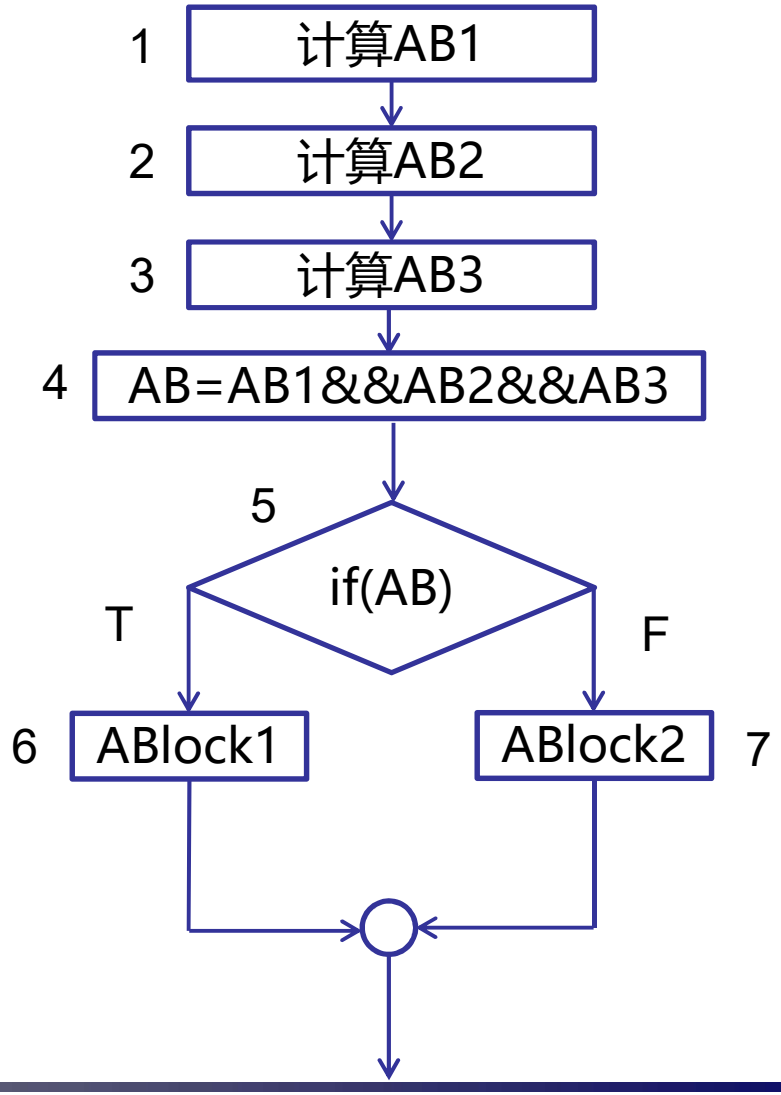
4.5 路径选择标准

谓词覆盖原则（条件组合覆盖）

选择路径，使得所有可能的条件都被执行，并且影响选定的路径所有可能的真值条件的组合全都执行一次

4.5 路径选择标准-谓词覆盖例





4.5 路径选择标准-谓词覆盖例

情况	OB1	OB2	OB3	OB
1	T	F	F	T
2	F	F	F	F

测试用例满足完全语句覆盖和分支覆盖

4.5 路径选择标准-谓词覆盖例

测试用例如何满足谓词覆盖?

情况	OB1	OB2	OB3	OB
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	T
5	F	T	T	T
6	F	T	F	T
7	F	F	T	T
8	F	F	F	F

目录

CONTENTS

4.6 生成测试输入

需要识别输入来强制路径的执行

4.6 生成测试输入-术语及过程

1、输入向量： 是所有读入例程的数据实体的集合，数据的值在进入例程前必须固定

例程输入向量的一些不同的形式：程序的输入参数；全局变量和常量；文件（是否存在或其内容）；汇编语言编程中的寄存器内容；网络连接；定时器

input vector:<value[], AS, MIN, MAX>

4.6 生成测试输入-术语及过程

2、谓词

决策点的逻辑函数

ti<AS
value[i]!=-999
value[i]>=MIN
value[i]<=MAX
tv>0

4.6 生成测试输入-术语及过程

3、路径谓词

与一条路径相关的谓词集（包含谓词的真值）

1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13

ti<AS ≡ True----------节点3
value[i]!=-999 ≡ True----------节点4
value[i]>=MIN ≡ True----------节点6
value[i]<=MAX ≡ True----------节点7
ti<AS ≡ False----------节点3
tv>0 ≡ True----------节点10

4.6 生成测试输入-术语及过程

4、谓词解释

用**符号替换**一个路径上的操作的过程，以便**只用输入向量和常量来表达谓词。**这是因为局部变量在为促使路径执行而选择输入时并没有发挥实际作用

节点	节点描述	谓词解释
1	input vector:<value[ ], AS,MIN，MAX>	
2	i=0,ti=0,tv=0,sum=0	
3(T)	ti<AS	0<AS
4(T)	value[i]!=-999	value[0]!=-999
5	ti++	ti=0+1=1
6(T)	value[i]>=MIN	value[0]>=MIN
7(T)	value[i]<=MAX	value[0]<=MAX
8	tv++	tv=0+1=1
	sum=sum+value[i]	sum=0+value[0]=value[0]
9	i++	i=0+1=1
3(F)	ti<AS	1<AS
10(T)	tv>0	1>0
12	av=(double)sum/tv	av=(double)value[0]/1
13	return(av)	return(value[0])

4.6 生成测试输入-术语及过程

5、路径谓词表达式

它没有局部变量，只由输入向量及可能的常量向量元素组成

它是由输入向量及可能的常量向量构成的**约束集**

通过求解路径谓词表达式中的约束集，可以生成路径促成输入

如果没能求得约束集的解，**那便不存在任何的输入可以使选定的路径执行**。即认为选定的路径是不可行的。

不可行的路径并不意味着路径谓词表达式中一个或多个成员是不可满足的。它只意味着路径谓词表达式中所有成员的**总组合是不可满足**的

路径谓词表达式的不可行性表明，需要考虑其他路径以满足所选路径选择标准

4.6 生成测试输入-术语及过程

5、路径谓词表达式

$0<AS \equiv True$

$value[0]!=-999 \equiv True$

$value[0]>=MIN \equiv True$

$value[0]<=MAX \equiv True$

$1<AS \equiv False$

$1>0 \equiv True$

4.6 生成测试输入-术语及过程

6、从路径谓词表达式生成输入数据

求解相应的路径谓词表达式，生成输入数据，能够迫使程序执行选定的路径

AS = 1
MIN = 25
MAX = 35
value[0] = 30

4.7 选择测试数据的示例

ReturnAverage函数的满足语句覆盖和分支覆盖的测试用例

测试数据集	输入向量			
	AS	MIN	MAX	Value[]
1	1	5	20	[10]
2	1	5	20	[-999]
3	1	5	20	[4]
4	1	5	20	[25]

4.7 选择测试数据的示例

假如ReturnAverage函数中存在如下错误，第一组测试用例无法发现：

正确	错误
av=(double)sum/tv	av=(double)sum/ti
sum=sum+value[i]	sum=value[i]

4.7 选择测试数据的示例

结论

生成测试数据时必须满足一定的**选择标准**，因为这些选择标准确定了程序期望覆盖的一些方面

在满足覆盖标准后应该生成额外的测试，它们将会比为了满足覆盖标准的简单测试要长很多

给定程序的一组测试数据，可以向程序注入一些错误，而不被那些测试用例所发现（**变异测试**）

4.9 本章小结

小结

本章主要是对控制流测试的概念、方法、测试步骤做了简要介绍，理解起来比较容易，但是要通过大量的案例才能更容易理解，所以要有实践过程

内容包括：基本**概念**（程序路径，控制流）；控制流图（**CFG**）及如何绘制；**路径选择标准**（覆盖标准，如何选取路径）；**如何识别输入来强制路径执行**（输入向量，谓词，路径谓词，谓词解释，路径谓词表达式）

控制流测试

End