嵌套式计算器测试计划

# 修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 说明 | 作者 |
| 2020/05/03 | 1.0.0 | 软件测试计划文档的初步构建 | 王笑然 |
| 2020/05/10 | 1.1.0 | 添加测试计划中路径测试部分 | 褚宇轩，王志远 |
| 2020/05/10 | 1.2.0 | 添加测试计划中数据流测试部分 | 王笑然，吕艺 |
| 2020/05/13 | 2.0.0 | 测试计划的完善 | 王笑然 |

目录

[嵌套式计算器测试计划 1](#_Toc40623517)

[修订历史记录 2](#_Toc40623518)

[1. 简介 4](#_Toc40623519)

[1.1 编写目的 4](#_Toc40623520)

[1.2 项目背景 4](#_Toc40623521)

[1.3 测试范围 4](#_Toc40623522)

[1.4 参考文档 4](#_Toc40623523)

[2. 测试参考文档和测试提交文档 5](#_Toc40623524)

[2.1 测试参考文档 5](#_Toc40623525)

[2.2 测试提交文档 5](#_Toc40623526)

[3. 测试进度 5](#_Toc40623527)

[4. 测试资源 6](#_Toc40623528)

[4.1 人力资源 6](#_Toc40623529)

[4.2 测试环境 6](#_Toc40623530)

[4.3 测试工具 6](#_Toc40623531)

[5. 测试策略 7](#_Toc40623532)

[5.1 路径测试 7](#_Toc40623533)

[5.1.1 Expression函数 7](#_Toc40623534)

[5.1.2 Term函数 8](#_Toc40623535)

[5.1.3 single函数 10](#_Toc40623536)

[5.1.4 primary函数 12](#_Toc40623537)

[5.1.5 fac函数 13](#_Toc40623538)

[5.1.6 Token\_Stream的get函数 14](#_Toc40623539)

[5.2 数据流测试 19](#_Toc40623540)

[5.2.1针对ts变量 19](#_Toc40623541)

[5.2.2 针对value变量 22](#_Toc40623542)

[5.2.3 针对token变量 29](#_Toc40623543)

# 简介

## 编写目的

为了全面、系统地对“计算器”代码的正确性进行评估与测试，我们在了解程序源码的基础上进行白盒测试，从路径测试和数据流测试两方面观察代码，从而保证计算器的正确性。同时我们对测试思路进行分析和说明，完成软件测试课程的“白盒测试”作业，从而编写本文档。

## 项目背景

本产品为“嵌套式计算器”，可处理加减乘除模及阶乘运算，是基于大一时数据结构大作业的计算器进行改编而成，其原型为《C++程序设计原理与实现》一书中的计算器。

算式具有四种算数的优先级，其优先级(由最高优先到最低优先)为:

1. Primary，处理符号类型的判断，如辨别下一字符为数字类型
2. Single，处理单目运算，如阶乘运算
3. Term，处理乘除模运算
4. Expression，处理加减运算

接下来对算式的字符流进行语法分析，从而判断算式中的计算部分的计算优先级并计算。

## 测试范围

本次测试基于路径和数据流两种白盒测试方法对计算器进行测试。

## 参考文档

[M] 《Software Testing􀀅A Craftsman’s Approach Fourth Edition》Paul C. Jorgensen􀀅2014

[M] 《软件测试：第3版》，人民邮电出版社，2011

[M] 《C++程序设计原理与实现》

# 测试参考文档和测试提交文档

## 测试参考文档

[M] 《Software Testing􀀅A Craftsman’s Approach Fourth Edition》Paul C. Jorgensen􀀅2014

[M] 《软件测试：第3版》，人民邮电出版社，2011

## 测试提交文档

《测试计划》

《测试报告》

# 测试进度

|  |  |  |  |
| --- | --- | --- | --- |
| 测试活动 | 计划开始日期 | 实际开始日期 | 结束日期 |
| 测试项目分析 | 2020/05/03 | 2020/05/03 | 2020/05/03 |
| 两种测试方式实现 | 2020/05/10 | 2020/05/10 | 2020/05/15 |
| 文档完善 | 2020/05/16 | 2020/05/16 | 2020/05/17 |

# 4. 测试资源

## 4.1 人力资源

|  |  |  |
| --- | --- | --- |
| 姓名 | 角色 | 具体职责或注释 |
| 王笑然 | 组长 | 数据流测试及文档的编写与整合 |
| 王志远 | 组员 | 路径测试及文档的编写 |
| 吕艺 | 组员 | 数据流测试及文档的编写 |
| 褚宇轩 | 组员 | 路径测试及文档的编写 |

## 4.2 测试环境

|  |  |
| --- | --- |
| 软件环境 | 硬件环境 |
| IntelliJ IDEA + Junit | 个人笔记本电脑 |

## 4.3 测试工具

|  |  |  |  |
| --- | --- | --- | --- |
| 用途 | 工具名称 | 生产厂商 | 版本 |
| 访问应用 | Chrome浏览器 | Google | 80.0.3987.163 |

# 5. 测试策略

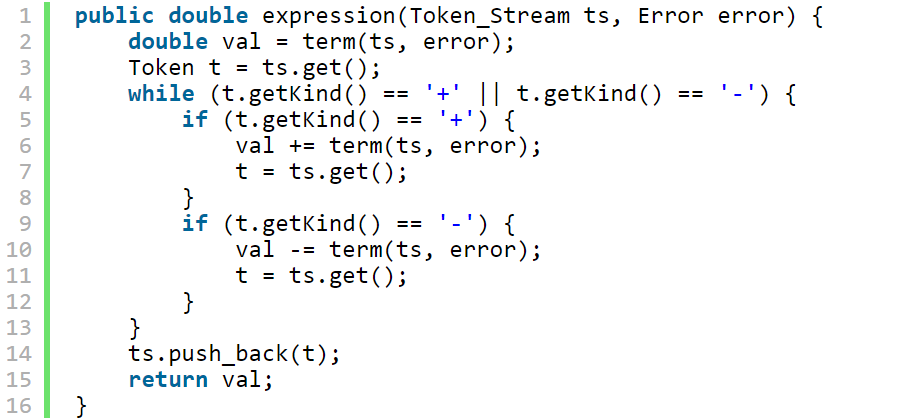
## 5.1 路径测试

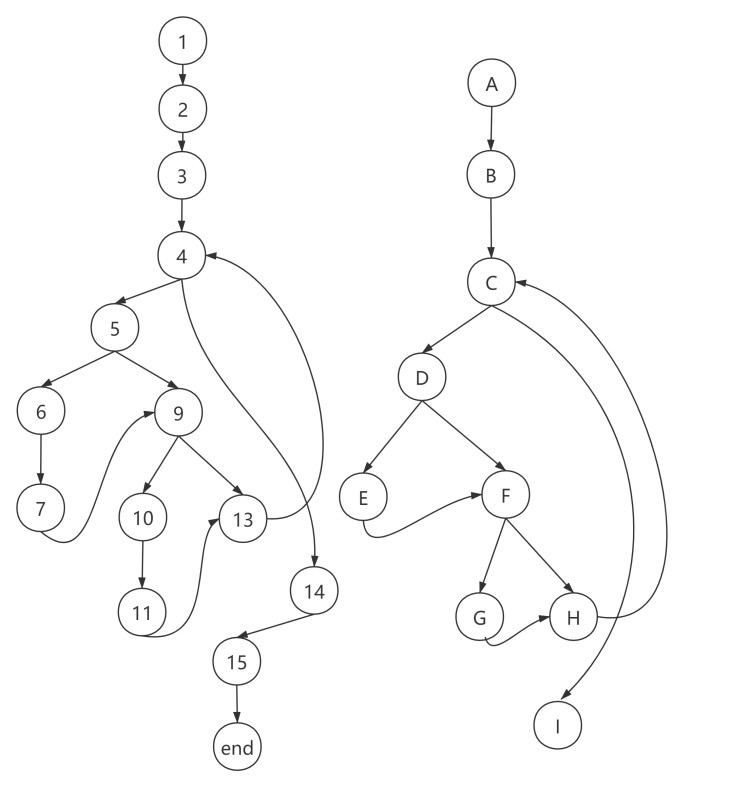
在路径测试中，我们针对每个函数，画出它们的路径图，并表示为DD-路径；

在测试覆盖指标方面，我们使用了C2作为覆盖指标，即覆盖每条DD路径+循环覆盖

### 5.1.1 Expression函数

代码及路径图如下：





#### 测试用例

可以设置1个测试用例 3 + 4 – 5 期望输出 2

循环0次 3；

循环1次 3 + 4

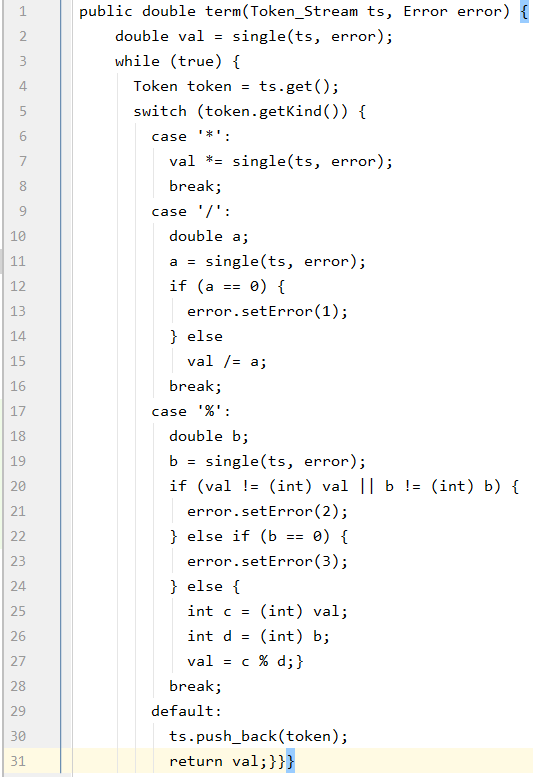
循环2次：3+4+5

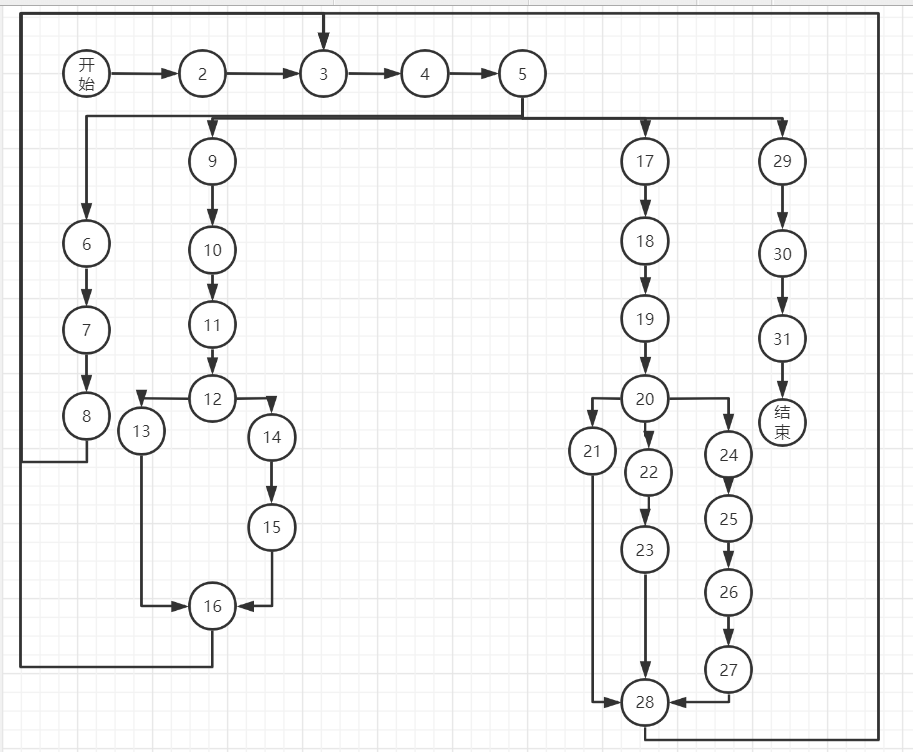
这个函数没有最大循环次数

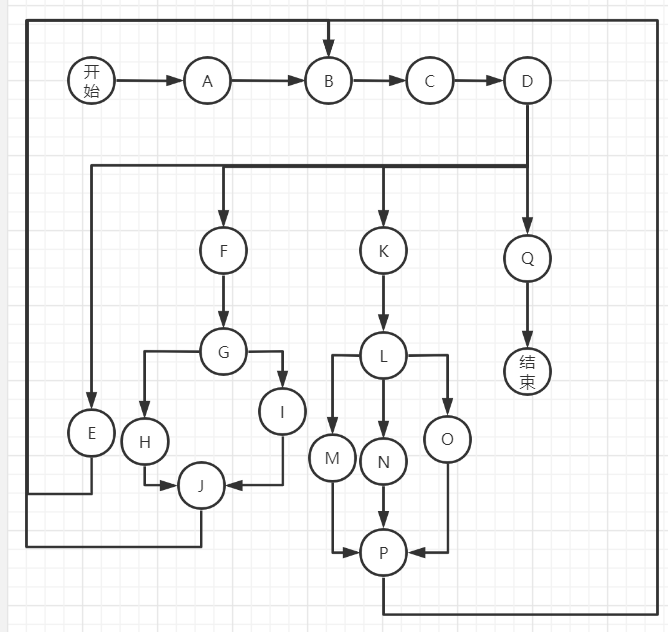
|  |  |  |
| --- | --- | --- |
| 测试输入 | 期望值 | 覆盖的DD路径 |
| 3+4-5; | 2 | <A,B,C,D,E,F,H,C,D,F,G,H,C,I> |
| 3; | 3 | <A,B,C,I> |
| 3+4; | 7 | <A,B,C,D,E,F,H,C,I> |
| 3+4+5; | 12 | <A,B,C,D,E,F,H.C,D,E,F,H,C,I> |

### 5.1.2 Term函数

代码及路径图如下：





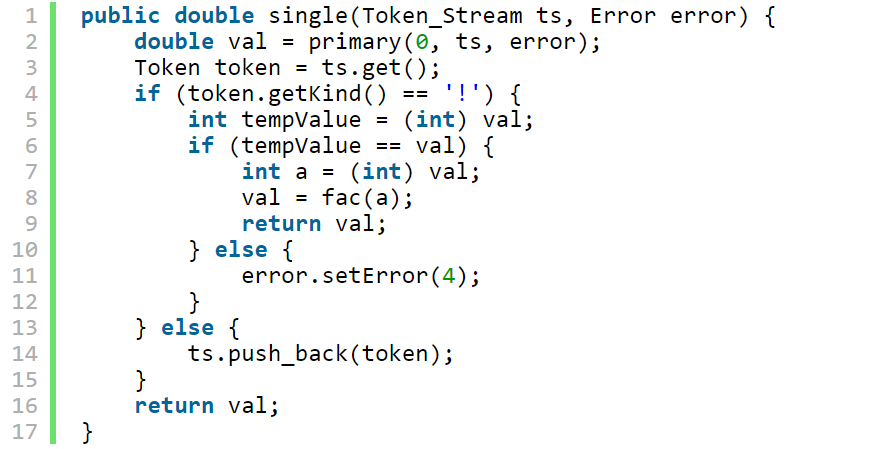


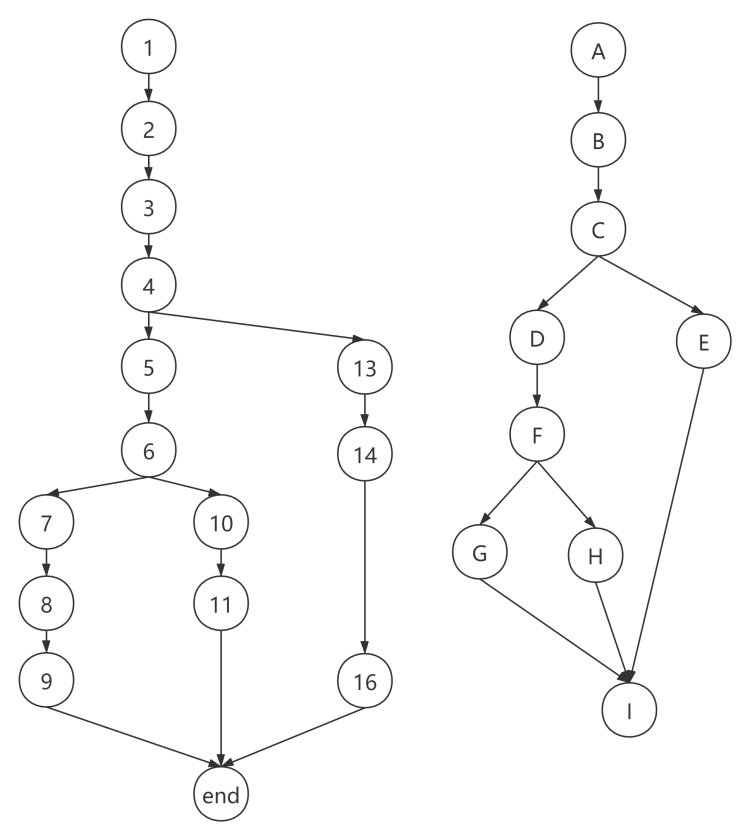
#### 测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 路径 | 输入 | 预计输出 | 实际输出 |
| 1 | ABCDEBCDQ | 3\*3; | 9 | 9 |
| 2 | ABCDQ | 3; | 3 | 3 |
| 3 | ABCDFGIJBCDQ | 3/2; | 1.5 | 1.5 |
| 4 | ABCDFGHJBCDQ | 3/0; | Error=1 | Error=1 |
| 5 | ABCDKLMPBCDQ | 3%2.3 | Error=2 | Error=2 |
| 6 | ABCDKNPBCDQ | 3%0; | Error=3 | Error=3 |
| 7 | ABCDKOPBCDQ | 3%2; | 1 | 1 |

### 5.1.3 single函数

代码及路径图如下：





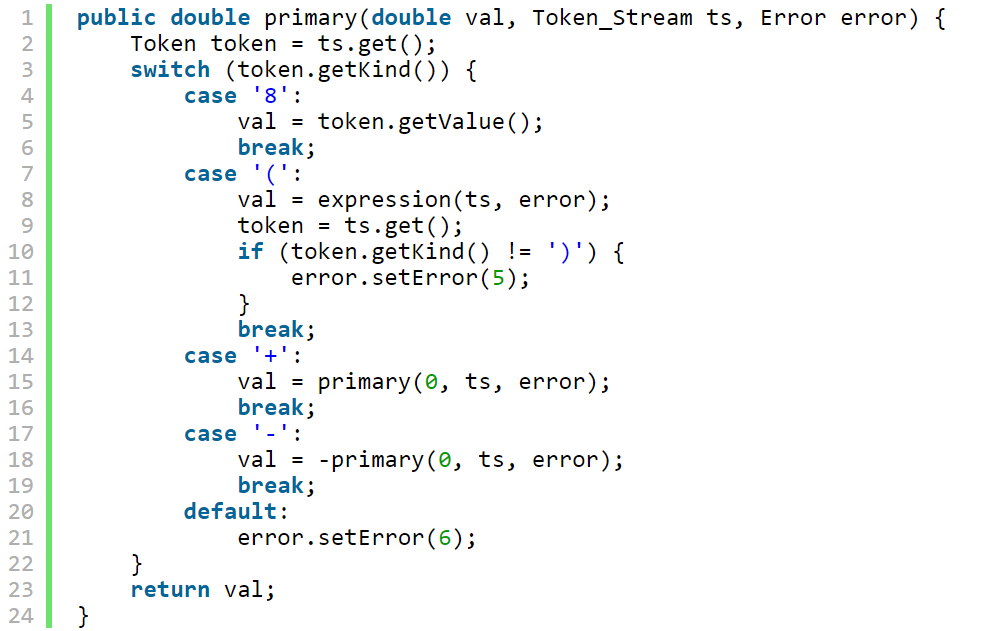
#### 测试用例

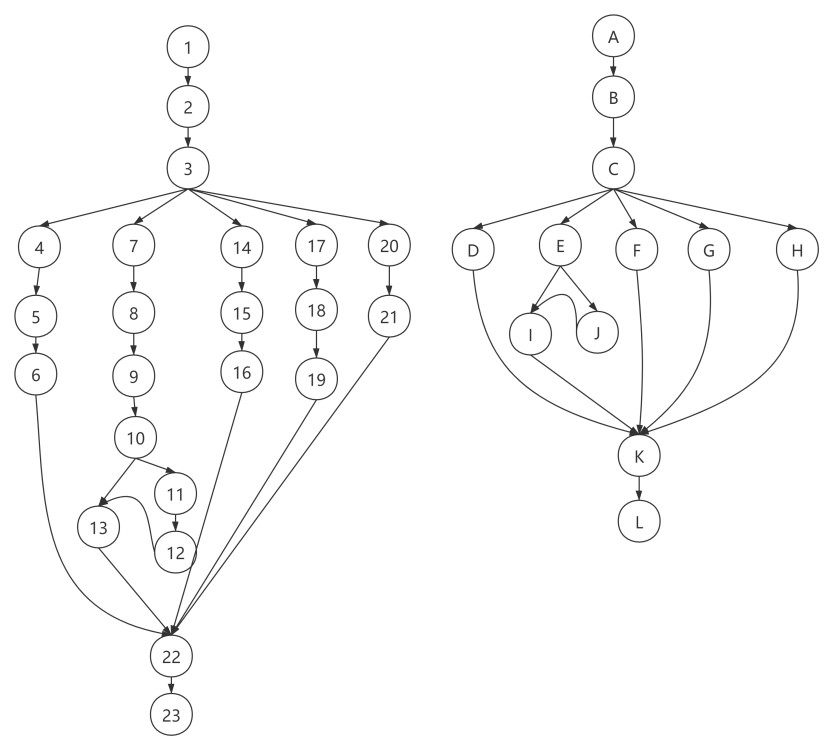
设计3个测试用例

3！ 3.5！ 3 期望值 6 error 3

|  |  |  |
| --- | --- | --- |
| 测试输入 | 期望值 | 覆盖的DD路径 |
| 3!; | 6 | <A,B,C,D,F,G,I> |
| 3.5!; | Error(4) | <A,B,C,D,F,H,I> |
| 3; | 3 | <A,B,C,E,I> |

### 5.1.4 primary函数





#### 测试用例

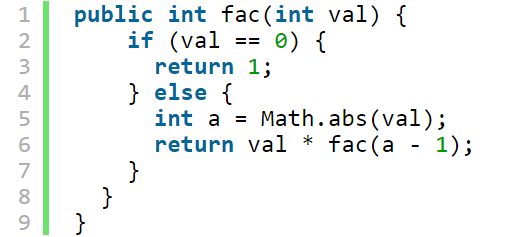
设计6个测试用例：

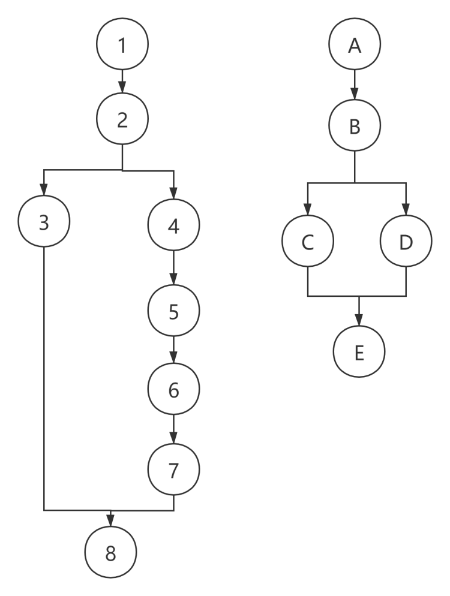
3， 3 +4 ， 4 – 3， 3 + ( 4 – 2, 3 + ( 4 -2 ) , 3 )

|  |  |  |
| --- | --- | --- |
| 测试输入 | 期望值 | 覆盖的DD路径 |
| 3; | 3 | <A,B,C,D,K,L> |
| +4; | 4 | <A,B,C,F,K,L> |
| -3; | -3 | <A,B,C,G,K,L> |
| (4-2; | Error(5) | <A,B,C,E,J,I,K,L> |
| (4-2); | 2 | <A,B,C,E,I,K,L> |
| ); | Error(6) | <A,B,C,H,K,L> |

### 5.1.5 fac函数

代码及路径图如下：

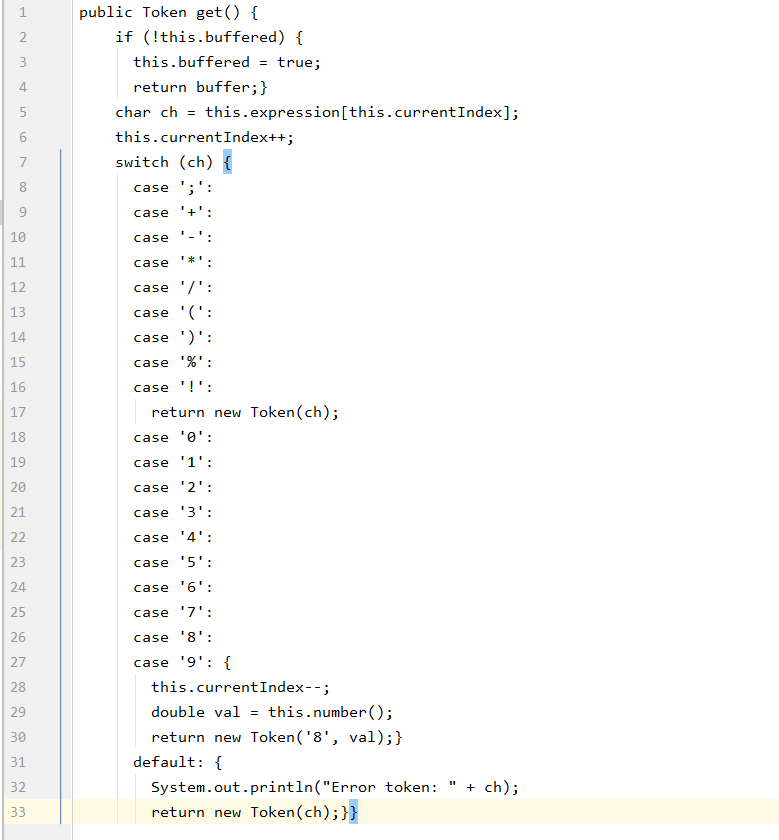


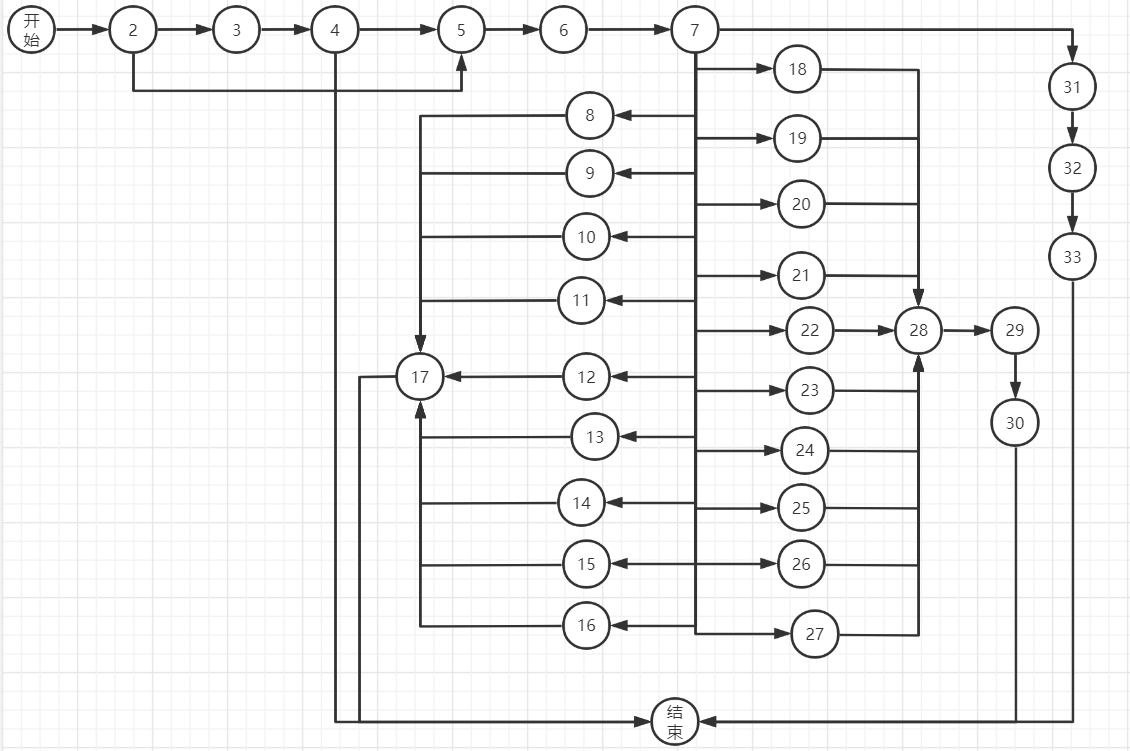


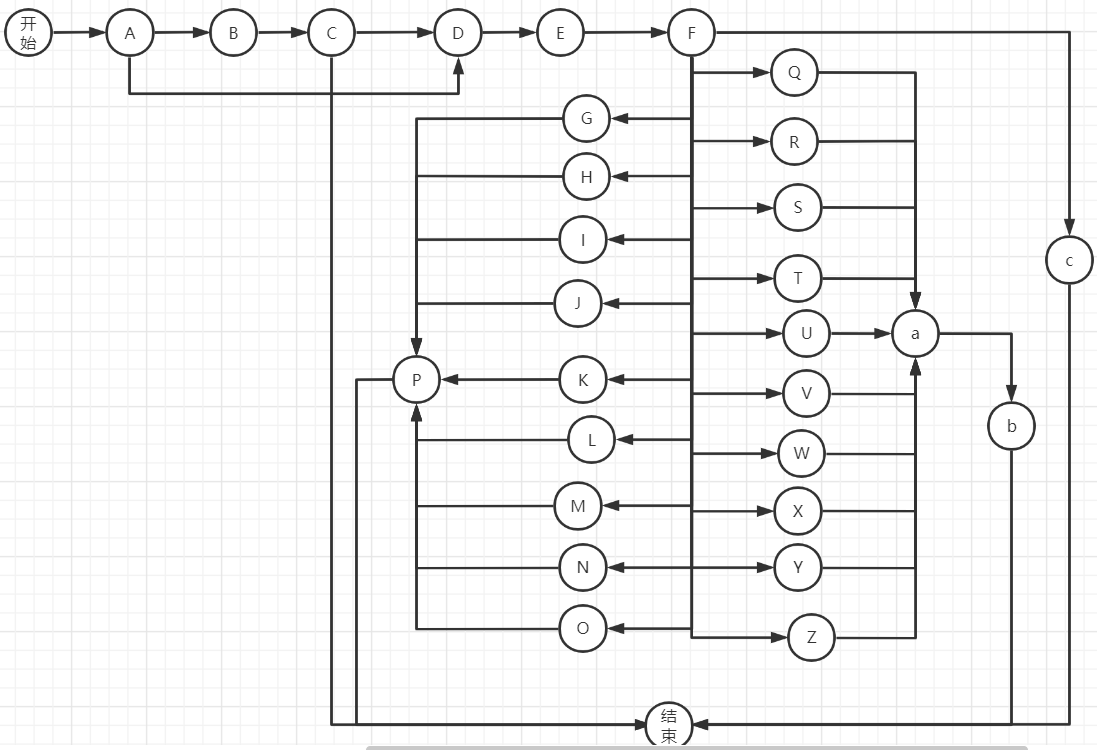
#### 测试用例

|  |  |  |
| --- | --- | --- |
| 测试输入 | 期望值 | 覆盖的DD路径 |
| 1 | 1 | <A,B,C,E> |
| 3 | 6 | <A,B,D,E> |

### 5.1.6 Token\_Stream的get函数



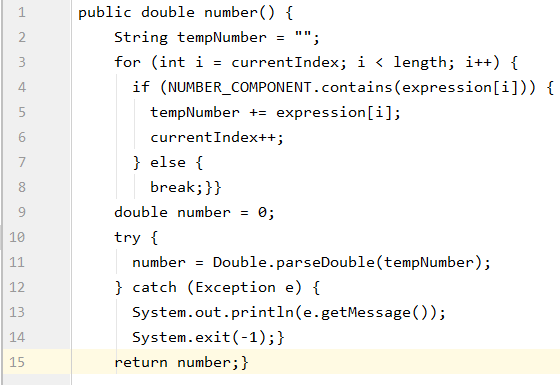


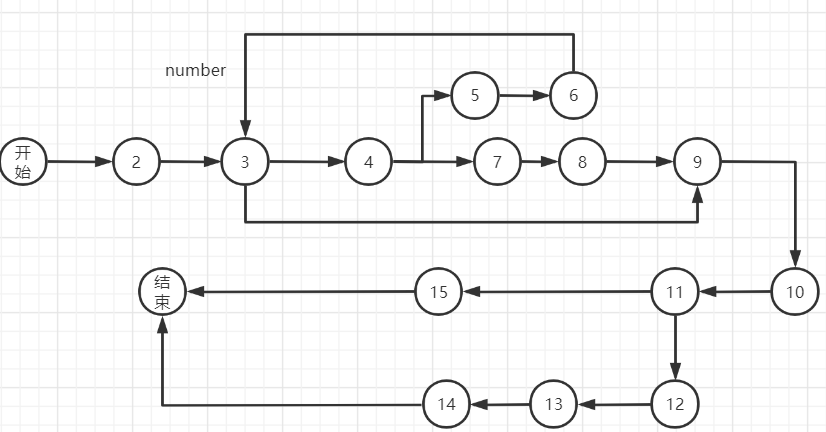


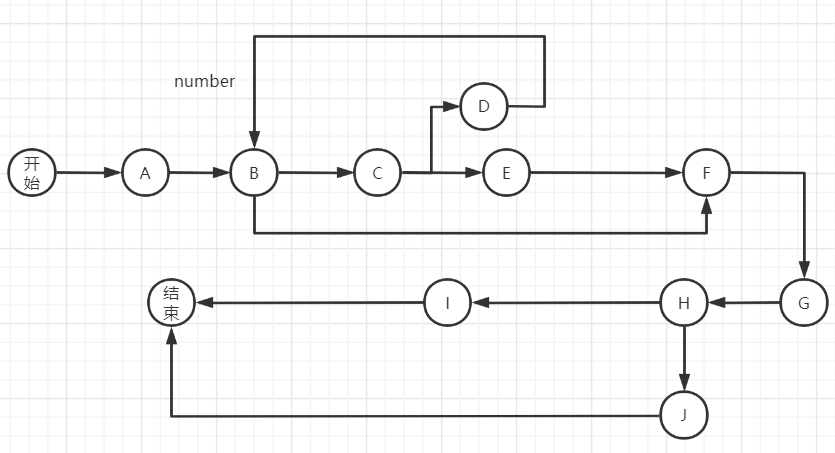
#### 测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 路径 | 输入 | 预期输出 | 实际输出 |
| 1 | ADEFGP | ; | Token.getKind()=; | Token.getKind()=; |
| 2 | ADEFQab | 0 | Token.getValue()  =0 | Token.getValue()  =0 |
| 3 | ADEFHP | + | Token.getKind()=+ | Token.getKind()=+ |
| 4 | ADEFIP | - | Token.getKind()=- | Token.getKind()=- |
| 5 | ADEFJP | \* | Token.getKind()=\* | Token.getKind()=\* |
| 6 | ADEFKP | / | Token.getKind()=/ | Token.getKind()=/ |
| 7 | ADEFLP | % | Token.getKind()=% | Token.getKind()=% |
| 8 | ADEFMP | ( | Token.getKind()=( | Token.getKind()=( |
| 9 | ADEFNP | ) | Token.getKind()=) | Token.getKind()=) |
| 10 | ADEFOP | ! | Token.getKind()=! | Token.getKind()=! |
| 11 | ADEFRab | 1 | Token.getValue()  =1 | Token.getValue()  =1 |
| 12 | ADEFSab | 2 | Token.getValue()  =2 | Token.getValue()  =2 |
| 13 | ADEFTab | 3 | Token.getValue()  =3 | Token.getValue()  =3 |
| 14 | ADEFUab | 4 | Token.getValue()  =4 | Token.getValue()  =4 |
| 15 | ADEFVab | 5 | Token.getValue()  =5 | Token.getValue()  =5 |
| 16 | ADEFWab | 6 | Token.getValue()  =6 | Token.getValue()  =6 |
| 17 | ADEFXab | 7 | Token.getValue()  =7 | Token.getValue()  =7 |
| 18 | ADEFYab | 8 | Token.getValue()  =8 | Token.getValue()  =8 |
| 19 | ADEFZab | 9 | Token.getValue()  =9 | Token.getValue()  =9 |
| 20 | ADEFc | ~ | 在log日志中出现Error token: ~ | 在log日志中出现Error token: ~ |
| 21 | ABCD | push\_back(Token(**'\*'**  )) | Token.getKind()=**'\*'** | Token.getKind()=**'\*'** |

#### 5.1.7 Token\_Stream的number函数







#### 测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 路径 | 输入 | 预计输出 | 实际输出 |
| 1 | ABFGHJ | **""** | 在log中出现empty String，同时返回0 | 在log中出现empty String，同时返回0 |
| 2 | ABCDBEFGHI | **"1.2"** | 1.2 | 1.2 |

## 5.2 数据流测试

数据流测试是针对数据内容进行测试的测试方法，因此个人认为应该分为全局变量和局部变量分别测试  
在计算器代码中，TokenStream和Error类是每个计算唯一的,同时每个计算部分的val是连续的流，分析的时候应该改着眼于全局的数据流图，而函数内部变量如token只需要在函数级别进行跟踪，画出数据流图  
数据流图可以使用processon进行绘制

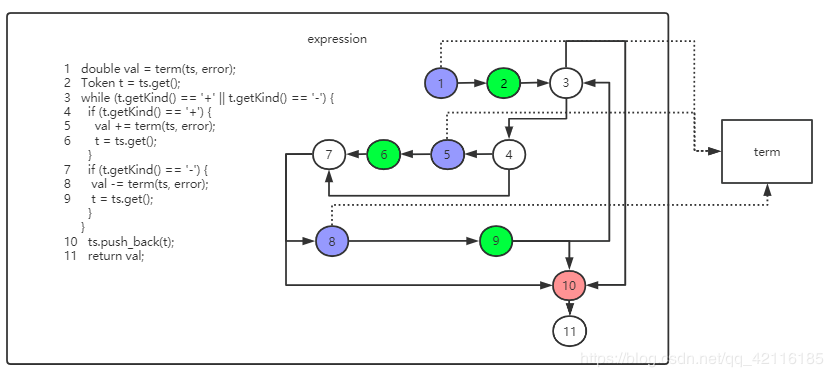
### 5.2.1针对ts变量

数据流图中红色为def，蓝色为def/use，绿色为use,虚线为函数调用

#### def/use图

|  |  |  |
| --- | --- | --- |
| 函数 | def行数 | use行数 |
| main | 3 | 6 |
| expression | 2, 6, 9, 10 | 1, 2, 5, 6, 8, 9 |
| term | 3, 29 | 1, 3, 6, 10, 18 |
| single | 2, 12 | 1, 2 |
| primary | 1, 9 | 1, 8, 9, 15, 18 |

#### 数据流图

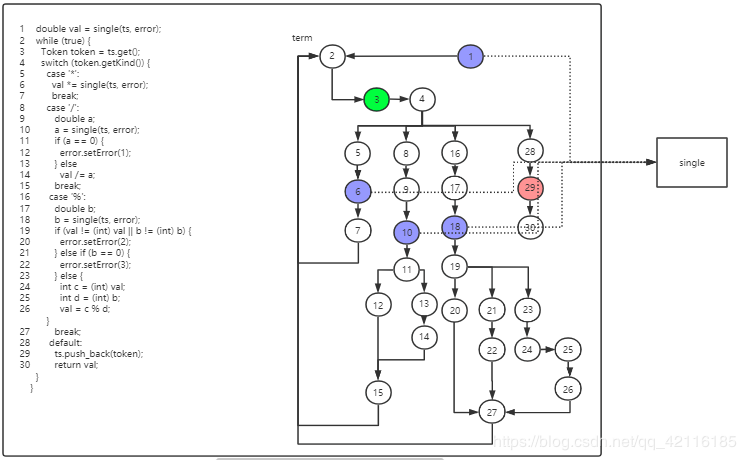


#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 2 | 5 | <2, 3, 4, 5> |
| 2 | 8 | <2, 3, 4, 7, 8> |
| 2 | 9 | <2, 3, 4, 7, 8, 9> |
| 6 | 8 | <6, 7, 8> |
| 6 | 9 | <6, 7, 8, 9> |
| 9 | 5 | <9, 3, 4, 5> |
| 9 | 6 | <9, 3, 4, 5, 6> |
| 1, 5, 9 | primary的1 | <1或5或9, term的1, single的1, primary的1> |
| 10 | primary的8 | <10, 11, primary的8> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| 1-1+1+1-1 | <2, 3, 4, 5>  <2, 3, 4, 7, 8>  <2, 3, 4, 7, 8, 9>  <6, 7, 8>  <6, 7, 8, 9>  <9, 3, 4, 5>  <9, 3, 4, 5, 6> | 1 |

#### 数据流图

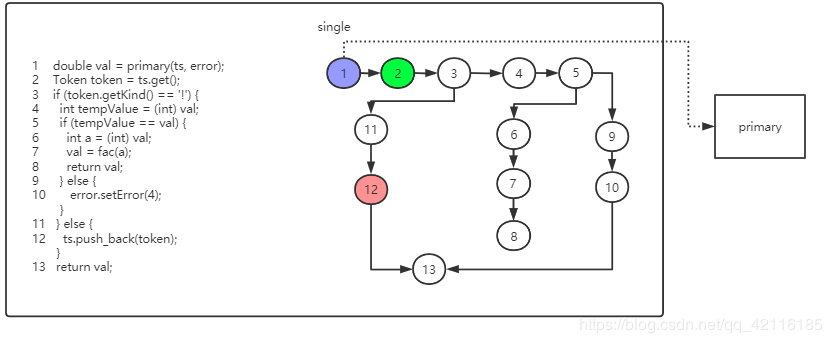


#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 3 | 6 | < 3, 4, 5, 6> |
| 3 | 10 | < 3, 4, 8, 9, 10> |
| 3 | 18 | < 3, 4, 17, 18> |
| {1, 6, 10, 18} | primary的1 | <{1, 6, 10, 18}, single的1, primary的1> |
| 29 | expression的{1, 5, 8} | <29, 30, expression的{1, 5, 8}> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| 1\*1/1%1 | < 3, 4, 5, 6>  < 3, 4, 8, 9, 10>  < 3, 4, 17, 18> | 0 |

#### 数据流图

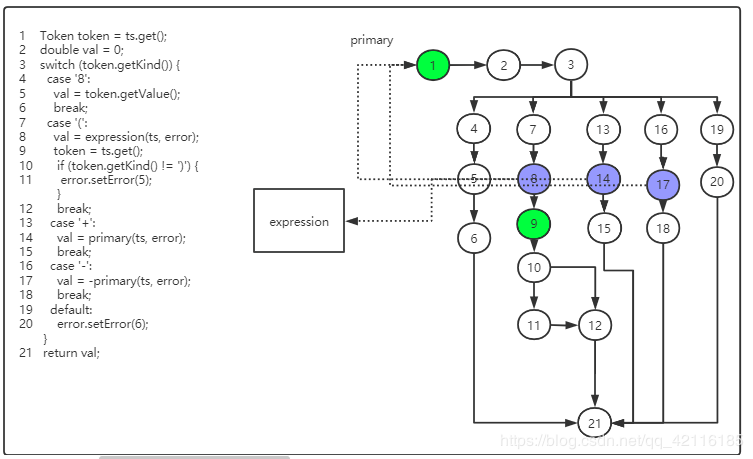


#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | primary{1} | <1, primary{1}> |
| 12 | term{1, 6, 10, 18} | <1, term{1, 6, 10, 18}> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| 3！ | <1, primary{1}> | 6 |

#### 数据流图



#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | 8 | <1, 2, 3, 7, 8> |
| 1 | 9 | <1, 2, 3, 7, 8, 9> |
| 1 | 15 | <1, 2, 3, 13, 14, 15> |
| 1 | 18 | <1, 2, 3, 16, 17, 18> |
| 8 | 1 | <8, expression{1}, term{1}, single{1}, 1> |
| 9 | single{1} | <9, {10, 11, 12}/{10, 12}, 21, single{1}> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| (1) | <1, 2, 3, 7, 8>  <1, 2, 3, 7, 8, 9> | 1 |
| +1 | <1, 2, 3, 13, 14, 15> | 1 |
| -1 | <1, 2, 3, 16, 17, 18> | -1 |

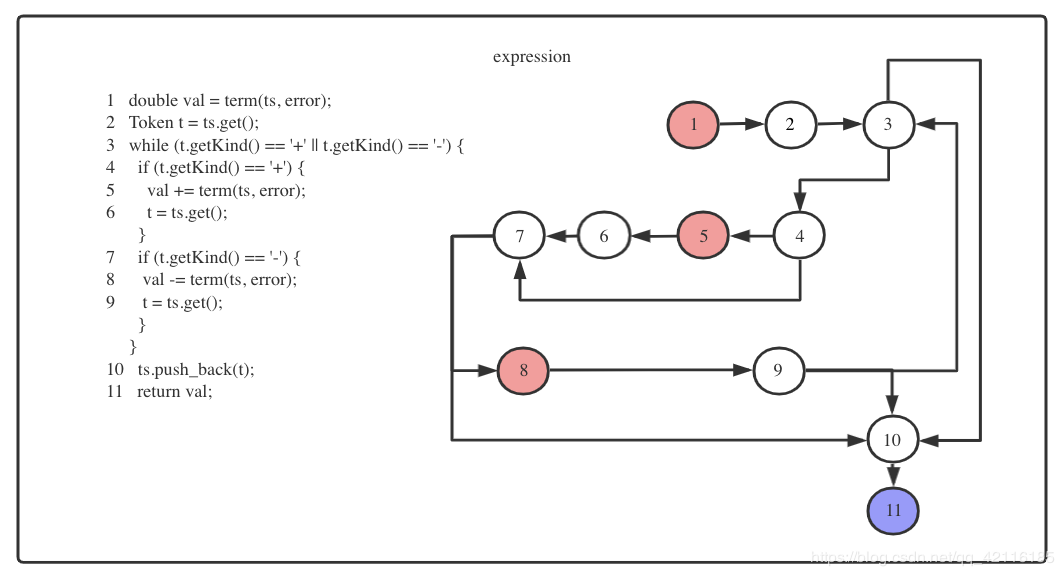
### 5.2.2 针对value变量

#### def/use图

|  |  |  |
| --- | --- | --- |
| **函数** | **def行数** | **use行数** |
| primary | 2,5,8,14,17 | 21 |
| expression | 1,5,8 | 11 |
| term | 1,6,14,19,26 | 19, 24,30 |
| single | 1,7 | 4, 5, 6, 8, 13 |
| value | 1 | 2,5,6 |

#### **expression函数**

#### 数据流图



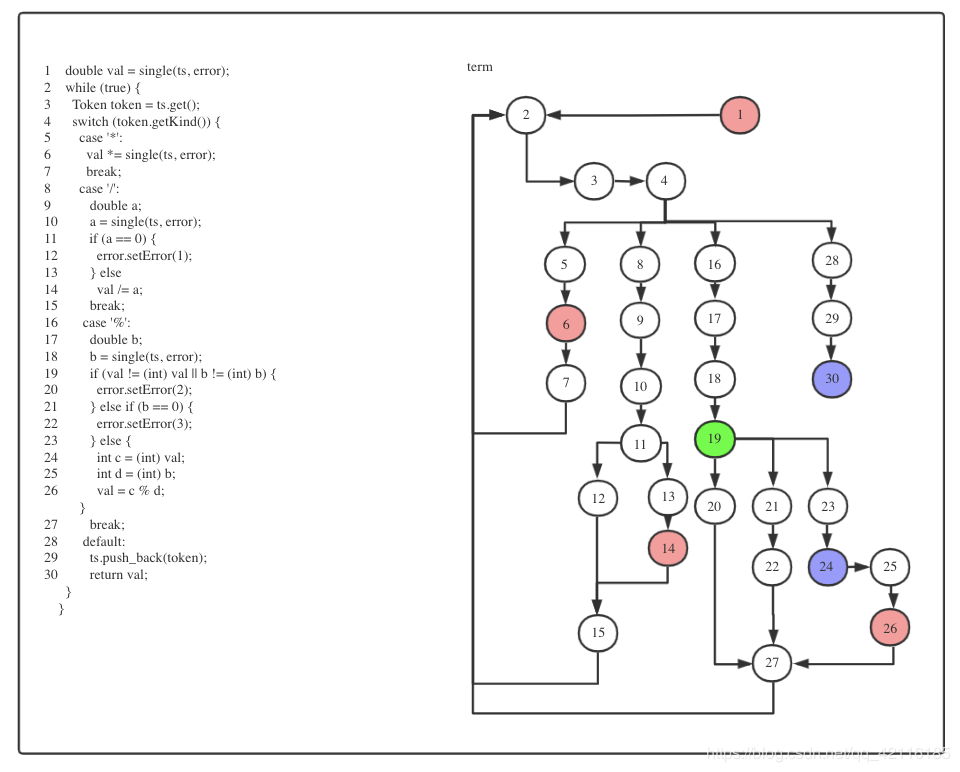
#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | 11 | <1,2,3,4,5,6,7,8,9,10,11> |
| 5 | 11 | <5, 6,10,11> |
| **定义节点** | **使用节点** | **路径** |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| new TokenStream(“1+1”), new Error() | <1,2,3,4,5,6,7,8,9,10,11>,<5, 6,10,11> | 2 |
| new TokenStream(“1-1”), new Error() | <8, 9, 10, 11> | 0 |

#### term函数

#### 数据流图



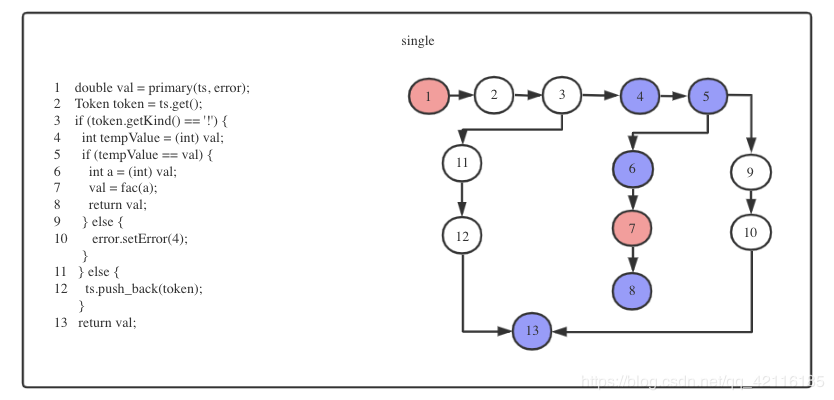
#### 路径

|  |  |  |
| --- | --- | --- |
| 定义节点 | 使用节点 | 路径 |
| 1 | 19 | <1, 2, 3, 4, 16,17,18,19> |
| 1 | 24 | <1,2,3,4,16,17,18,19,23,24> |
| 1 | 30 | <1,2,3,4,28,29,30> |
| 6 | 19 | <6,7,2,3,4,16,17,18,19> |
| 6 | 24 | <6,7,2,3,4,16,17,18,19,23,24> |
| 6 | 30 | <6,7,2,3,4,28,29,30> |
| 14 | 19 | <14,15,2,3,4,16,17,18,19> |
| 14 | 24 | <14, 15, 2,3,4,16,17,18,19,23,24> |
| 14 | 30 | <14,15,2,3,4,28,29,30> |
| 19 | 24 | <19,20,27,2,3,4,16,17,18,19,23,24> |
| 19 | 30 | <19,20,27,2,3,4,28,29,30> |
| 26 | 19 | <26,27,2,3,4,16,17,18,19> |
| 26 | 24 | <26, 27,2,3,4,16,17,18,19,23,24> |
| 26 | 30 | <26,27,2,3,4,28,29,30> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| new TokenStream(“1\*1”), new Error() | <1,2, 3, 4, 16,17,18,19>,  <1,2,3,4,28,29,30>,  <6,7,2,3,4,28,29,30> | 1 |
| new TokenStream(“1%2”), new Error() | <1,2,3,4,16,17,18,19,23,24>,  <1, 2, 3, 4, 16,17,18,19>,  <19,20,27,2,3,4,16,17,18,19,23,24>, <19,20,27,2,3,4,28,29,30>，<26,27,2,3,4,28,29,30> | 1 |
| new TokenStream(“1\*2%3”), new Error() | <6,7,2,3,4,16,17,18,19>, <6,7,2,3,4,16,17,18,19,23,24>, | 2 |
| new TokenStream(“1+1”), new Error() | <1,2,3,4,28,29,30> | 1 |
| new TokenStream(“6/3%3”), new Error() | <14,15,2,3,4,16,17,18,19>，  <14, 15, 2,3,4,16,17,18,19,23,24> | 2 |
| new TokenStream(“6/3”), new Error() | <14,15,2,3,4,28,29,30> | 2 |
| new TokenStream(“3%4%2”) | <26,27,2,3,4,16,17,18,19>,  <26, 27,2,3,4,16,17,18,19,23,24> | 1 |

#### single函数

#### 数据流图



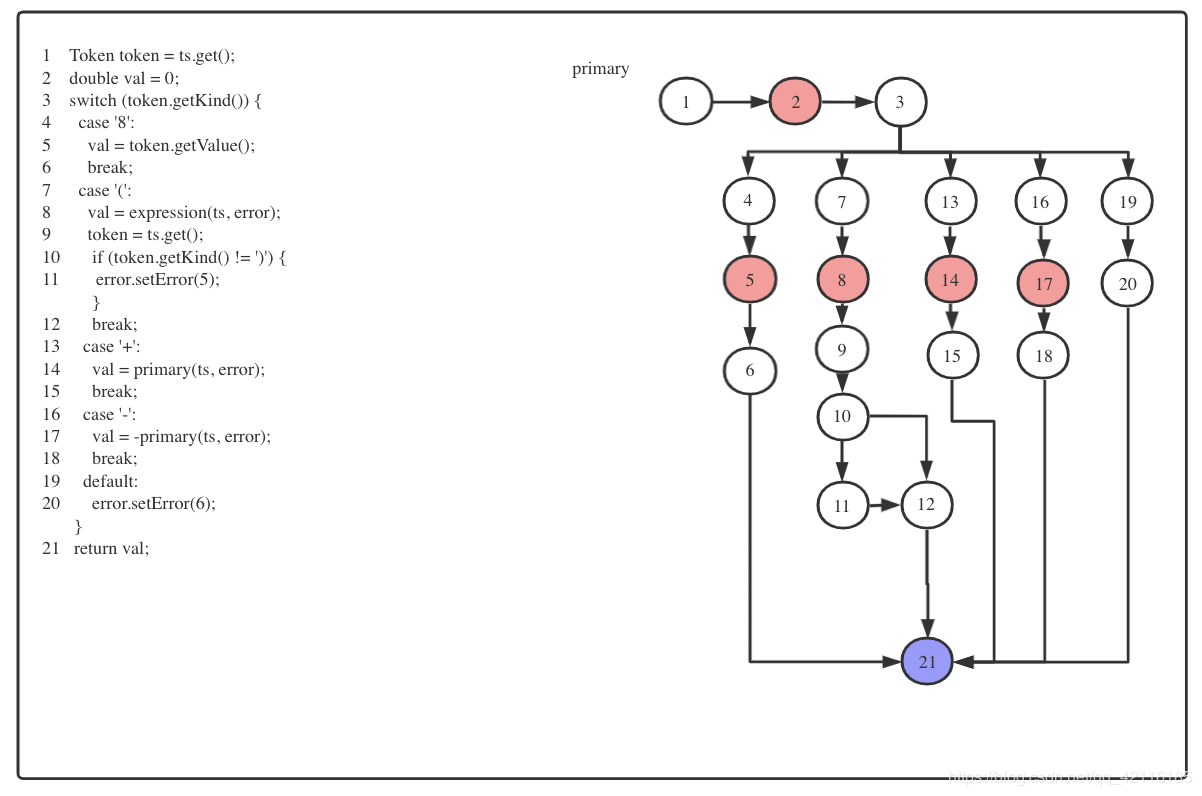
#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | 4 | <1,2,3,4> |
| 1 | 5 | <1,2,3,4,5> |
| 1 | 6 | <1,2,3,4,5,6> |
| 1 | 8 | <1,2,3,4,5,6,7,8> |
| 1 | 13 | <1,2,3,11,12,13> |
| 7 | 8 | <7,8> |

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试路径 | 期望结果 |
| new TokenStream(“3!”) | <1,2,3,4>,  <1,2,3,4,5>,  <1,2,3,4,5,6>, <1,2,3,4,5,6,7,8>,  <7, 9> | 6 |
| new TokenStream(“3”) | <1,2,3,11,12,13> | 3 |

#### primary函数

#### 数据流图

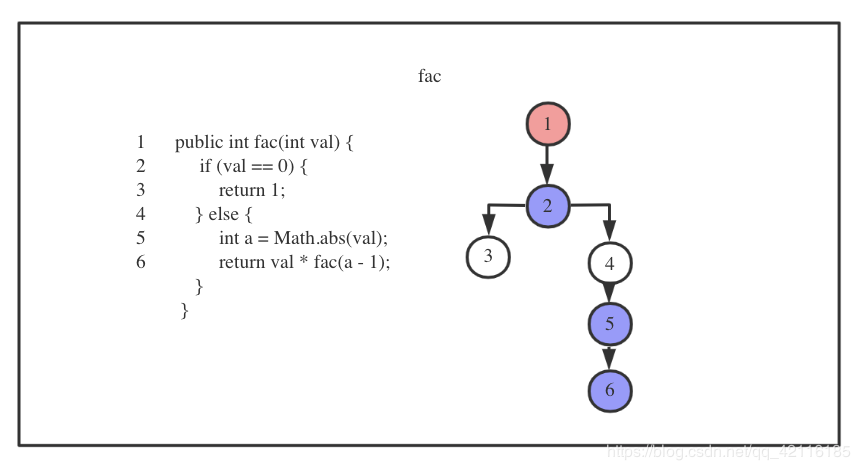


#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 2 | 21 | <2,3,7,8,9,10,11,12,21> |
| 5 | 21 | <5, 6, 21> |
| 8 | 21 | <8, 9, 10, 11, 12, 21> |
| 14 | 21 | <14, 15, 21> |
| 17 | 21 | <17, 18, 21> |

|  |  |  |
| --- | --- | --- |
| **测试用例** | **测试路径** | **期望结果** |
| new TokenStream(“5”), new Error() | <2,3,7,8,9,10,11,12,21>, <5, 6, 21> | 5 |
| new TokenStream("(1+1)"), new Error() | <8, 9, 10, 11, 12, 21> | 2 |
| new TokenStream("+1"), new Error() | <14, 15, 21> | 1 |
| new TokenStream("-1"), new Error | <17, 18, 21> | -1 |

#### fac函数



|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | 2 | 1,2 |
| 1 | 5 | 1,2,4,5 |
| 1 | 6 | 1,2,4,5,6 |

|  |  |  |
| --- | --- | --- |
| **测试用例** | **测试路径** | **期望结果** |
| fac(3) | <1,2>, <1,2,4,5>, <1,2,4,5,6> | 6 |

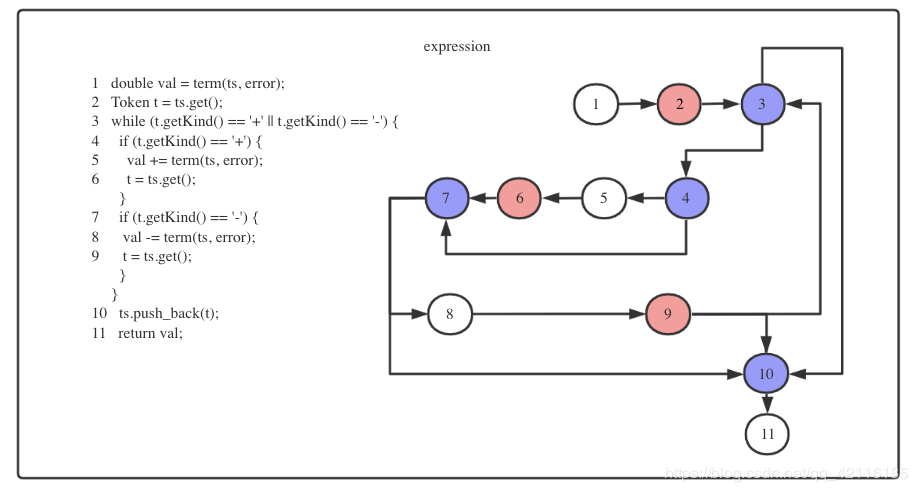
### 5.2.3 针对token变量

#### def/use图

|  |  |  |
| --- | --- | --- |
| **函数** | **定义节点** | **使用节点** |
| expression | 2，6，9 | 3，4，7，10 |
| term | 3 | 4, 29 |
| single | 2 | 3, 12 |
| primary | 1,9 | 3,5,10 |

#### Expression函数

#### 数据流图



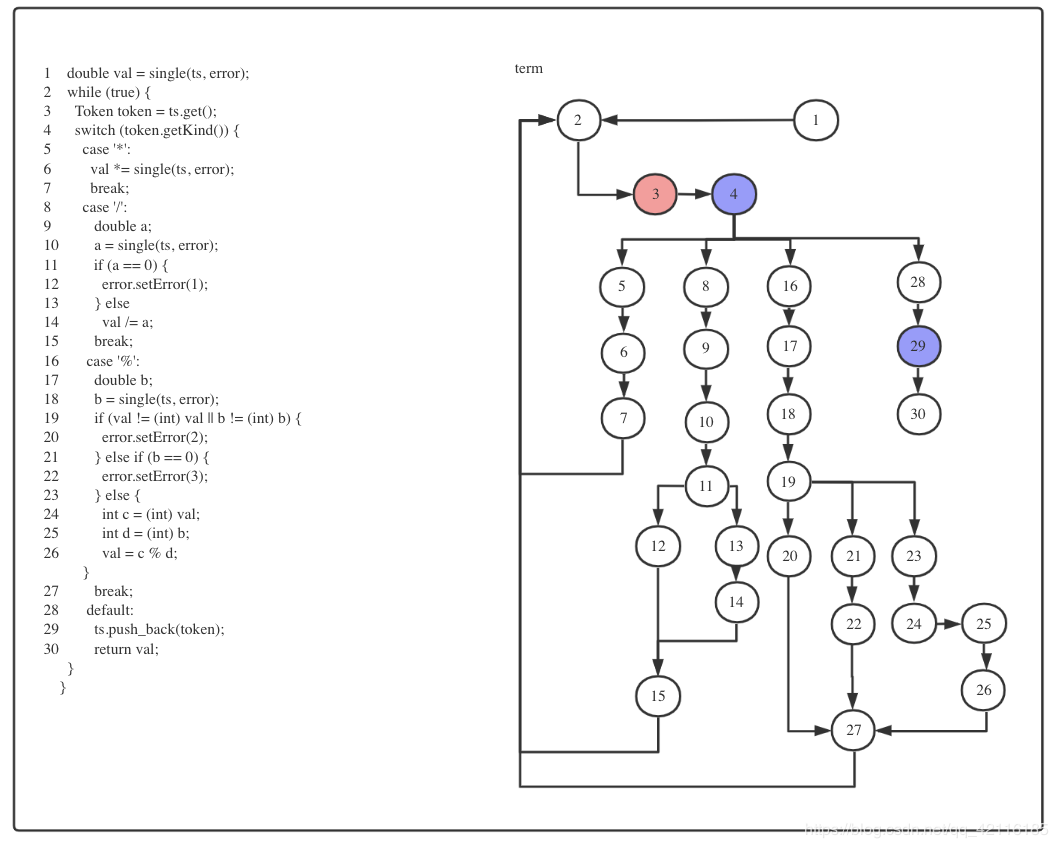
#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 2 | 3 | <2,3> |
| 2 | 4 | <2,3,4> |
| 2 | 7 | <2,3,4,5,6,7> |
| 2 | 10 | <2,3,10> |
| 6 | 7 | <6,7> |
| 6 | 10 | <6,7,8,9,10> |
| 9 | 10 | <9, 10> |

|  |  |  |
| --- | --- | --- |
| 测试样例 | 测试路径 | 期望结果 |
| new TokenStream(“1+1”), new Error() | <2,3>，<2,3,4>，<2,3,4,5,6,7>,<6,7>,<6,7,8,9,10> | 2 |
| new TokenStream(“1\*1”), new Error() | <2,3,10> | 1 |
| new TokenStream(“1-1”), new Error() | <9,10> | 1 |

#### Term函数

#### 数据流图



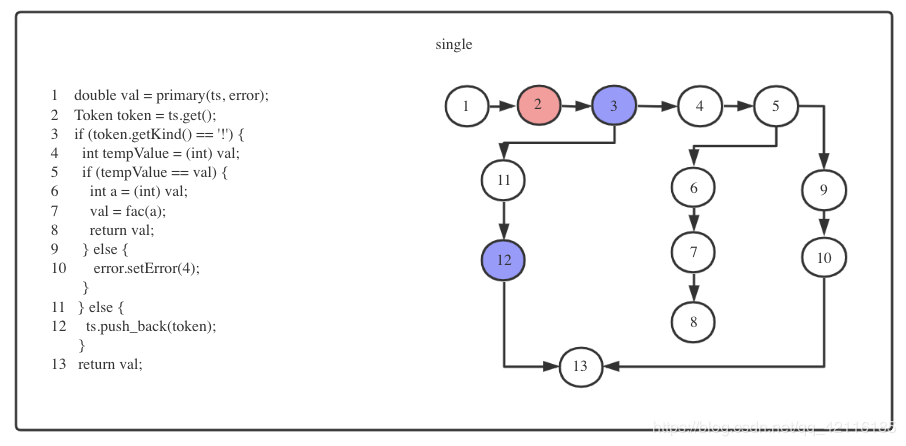
#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 3 | 4 | <3,4> |
| 3 | 29 | <3,4,28,29> |

|  |  |  |
| --- | --- | --- |
| **测试用例** | **测试路径** | **期望结果** |
| new TokenStream(“1”), new Error() | <3,4>,  <3,4,28,29> | 1 |

#### Single函数

#### 数据流图



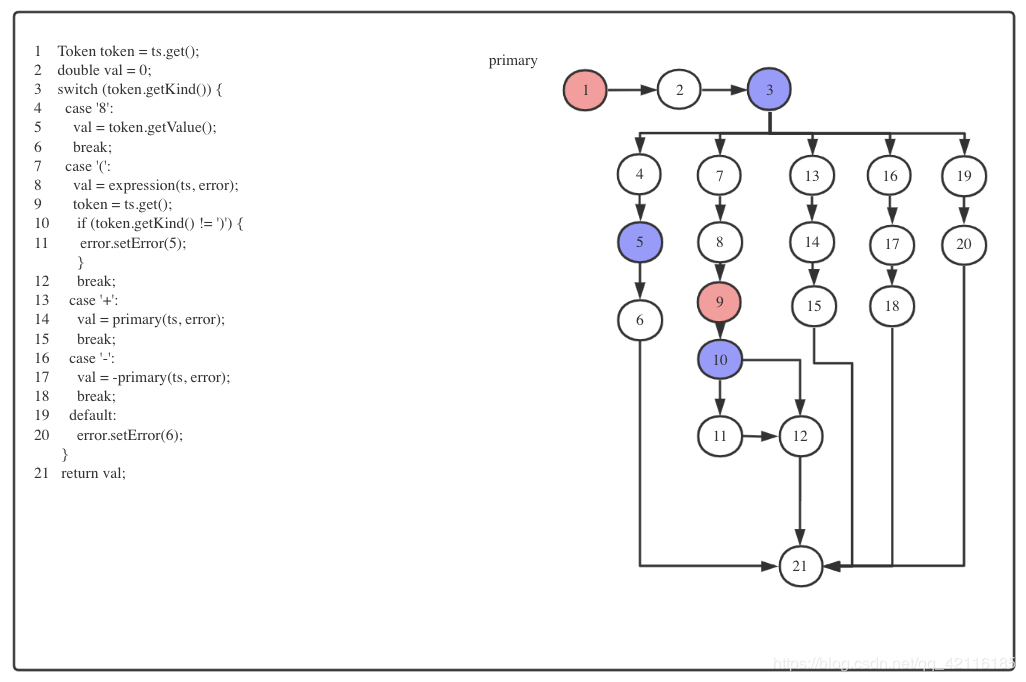
#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 2 | 3 | <2,3> |
| 2 | 12 | <2,3,11,12> |

|  |  |  |
| --- | --- | --- |
| **测试用例** | **测试路径** | **期望结果** |
| new TokenStream(“1”), new Error() | <2,3>,<2,3,11,12> | 1 |

#### Primary函数

#### 数据流图



#### 路径

|  |  |  |
| --- | --- | --- |
| **定义节点** | **使用节点** | **路径** |
| 1 | 3 | <1,2,3> |
| 1 | 5 | <1,2,3,4,5> |
| 1 | 10 | <1,2,3,7,8,9,10> |
| 9 | 10 | <9,10> |

|  |  |  |
| --- | --- | --- |
| **测试用例** | **测试路径** | **期望结果** |
| new TokenStream(“1”), new Error() | <1,2,3>，  <1,2,3,4,5> | 1 |
| new TokenStream("(1+1)"), new Error | <1,2,3,7,8,9,10>,  <9,10> | 2 |