Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Волгоградский государственный технический университет»

Утверждаю

Зав. кафедрой ПОАС, проф. д.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_ Дворянкин А. М.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

КАЧЕСТВО И НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Определить интерфейс функции при ее выделении из кода программы

Программа и методика испытаний

|  |  |
| --- | --- |
| СОГЛАСОВАНО:  Руководитель проекта:  Доцент кафедры ПОАС  \_\_\_\_\_\_\_\_\_\_\_\_\_ Сычев О. А.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | Разработчик:  Студент ПрИн-266  \_\_\_\_\_\_\_\_\_\_\_\_ Катышев А. М.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |
|  | Нормоконтролер:  Ассистент кафедры ПОАС  \_\_\_\_\_\_\_\_\_\_\_\_\_ Мамонтов Д. П.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |

Волгоград, 2017 г.

Содержание

[1 Объект испытаний 3](#_Toc481519691)

[2 Цель испытаний 3](#_Toc481519692)

[3 Требования к программе 3](#_Toc481519693)

[4 Требования к программной документации 3](#_Toc481519694)

[5 Средства и порядок испытаний 4](#_Toc481519695)

[6 Методы испытаний 4](#_Toc481519696)

[Приложение А. Тестирование функции extractProgramStructure 5](#_Toc481519697)

[Приложение Б. Тестирование функции performFunctionExtraction 48](#_Toc481519698)

# 1 Объект испытаний

Наименование программы: Function Refactor.

Программа Function Refactor предназначена для проведения рефакторинга «Выделение функции» данного участка кода внутри исходного кода программы.

# 2 Цель испытаний

Целью испытаний является проверка соответствия программы требованиям к функциональным характеристикам.

# 3 Требования к программе

Требования к программе приведены в пункте 3.4 технического задания.

# 4 Требования к программной документации

Состав программной документации должен включать в себя:

- в бумажной форме:

- техническое задание;

- технический проект в виде описания программы;

- программа и методика испытаний;

- руководство программиста;

- в электронной форме:

- копии всех документов бумажной формы;

- рабочая документация (генерируется doxygen);

- текст программы.

Вся документация должна быть представлена в соответствии с ГОСТ 19.

# 5 Средства и порядок испытаний

Для запуска тестов необходимо подключить библиотеку QTestLib к проекту программы.

# 6 Методы испытаний

Перечень проверок функции формирования структуры данных extractProgramStructure в виде аспектов тестирования, а также входные данные и соответствующие им ожидаемые результаты для функции extractProgramStructure представлены в Приложении A.

Перечень проверок функции, выполняющий главную задачу performFunctionExtraction в виде аспектов тестирования, а также входные данные и соответствующие им ожидаемые результаты для функции performFunctionExtraction представлены в приложении Б.

# Приложение А

Тестирование функции extractProgramStructure

Таблица 1. Аспекты тестирования функции extractProgramStructure

| № | Аспект тестирования | Возможные случаи |
| --- | --- | --- |
| 1 | Структура кода на верхнем уровне | - пустая программа;  - одна функция main;  - функция main с объявлением глобальных переменных;  - функция main с объявлением других функций. |
| 2 | Количество функций на верхнем уровне | - функций нет;  - одна функция main;  - много функций (3–4 объявления функций). |
| 3 | Количество объявленных глобальных переменных | - нет глобальных переменных;  - одна глобальная переменная;  - много глобальных переменных. |
| 4 | Количество объявленных глобальных структур | - нет описания структур;  - одна глобальная структура;  - много глобальных структур. |
| 5 | Тип переменной | - простой тип переменной;  - переменная с указанием квалификатора типа const;  - переменная-указатель;  - переменная-указатель с указанием квалификатора типа const. |
| 6 | Состав функции main | - пустая функция (без операторов);  - функция с одни оператором;  - функция с множеством операторов (3–5 вложенных операторов). |
| 7 | Количество аргументов функции | - функция без аргументов;  - функция с одним аргументом;  - функция с двумя аргументами;  - функция с множеством аргументов (3–5 аргументов). |
| 8 | Структура оператора if-else | - состоит из одного блока if;  - состоит из блоков if и else;  - состоит из блоков if, else и одного блока if-else;  - состоит из блоков if, else и нескольких блоков if-else (2–3 блока). |
| 9 | Состав блока if | - содержит один оператор внутри блока;  - содержит несколько операторов внутри блока. |
| 10 | Состав блока else | - содержит один оператор внутри блока;  - содержит несколько операторов внутри блока. |
| 11 | Состав блока if-else | - содержит один оператор внутри блока;  - содержит несколько операторов внутри блока. |
| 12 | Состав цикла for | - пустой цикл;  - содержит один оператор внутри цикла;  - содержит несколько операторов внутри цикла. |
| 13 | Инициализация цикла for | - цикл без инициализации;  - инициализация цикла объявлением новой переменной;  - инициализация цикла выражением. |
| 14 | Условие выполнения цикла for | - цикл не содержит условия выполнения;  - цикл содержит условие выполнения. |
| 15 | Шаг цикла for | - цикл не содержит описания шага;  - цикл содержит описания шага. |
| 16 | Структура цикла while | - является простым циклом while;  - является циклом do-while. |
| 17 | Состав цикла while | - пустой цикл;  - содержит один оператор внутри цикла;  - содержит несколько операторов внутри цикла. |
| 18 | Структура выражения | - простое выражение (типа a=b);  - сложное выражение с одним вложением составного выражения (типа a=b+1);  - сложное выражение со множественным вложением составного выражения (типа ++a=b%3+(b++)\*(a–4) ). |
| 19 | Описание структуры | - структура не содержит ни одного поля;  - структура содержит одно поле;  - структура содержит множество полей (2–4 поля). |

Входным параметром функции extractProgramStructure является дерево разбора текста программы в формате xml. Для удобства в данном приложении будут приведены входные данные в виде текста кода программы, чтобы не загромождать документацию. Выходные данные будут представлены схемами в виде деревьев, вложенности внутри которых будут изображаться стрелками.

Тест 1. Пустая программа

Входные данные:

""

Выходные данные:

codeDeclaration: NULL

Тест 2. Программа с пустой функцией main

Входные данные:

void main()

{

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody | NULL |

Тест 3. Программа с объявлением глобальных переменных

Входные данные:

int a;

void main()

{

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 7] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 4. Программа с объявлением множества функций

Входные данные:

int func1()

{

return 1;

}

int func2()

{

return 2;

}

void main()

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | int |
| identifier | func1 |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

function:

|  |  |
| --- | --- |
| returnType | int |
| identifier | func2 |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

return:

|  |  |
| --- | --- |
| value | 2 |

return:

|  |  |
| --- | --- |
| value | 1 |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [10; 13] |

Тест 5. Программа с объявлением множества глобальных переменных

Входные данные:

int a;

float b = 0;

void main()

{

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | float |
| identifier | b |
| isInitialized | true |
| value | "0" |
| visibilityArea | [1; 7] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 7] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 6. Программа с объявлением одной глобальной структуры

Входные данные:

int a;

struct str

{

int b;

};

void main()

{

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

struct:

|  |  |
| --- | --- |
| identifier | str |
| member | variable |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 7] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [3; 4] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 7. Программа с объявлением нескольких глобальных структур

Входные данные:

struct str1

{

int a;

};

struct str2

{

int b;

};

void main()

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 3] |

struct:

|  |  |
| --- | --- |
| identifier | str1 |
| member | variable |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [6; 7] |

struct:

|  |  |
| --- | --- |
| identifier | str2 |
| member | variable |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [10; 13] |

Тест 8. Программа с простой переменной

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

=

5

a

0

a

>

Тест 9. Программа с константной переменной

Входные данные:

void main()

{

int a;

const float b = 5;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | isConst:true  type:float |
| identifier | b |
| isInitialized | true |
| value | "5" |
| visibilityArea | [3; 6] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 6] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 10. Программа с переменной-указателем

Входные данные:

void main()

{

int a;

float\* b = NULL;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | isPointer:true  type:float |
| identifier | b |
| isInitialized | true |
| value | "5" |
| visibilityArea | [3; 6] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 6] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 11. Программа с константной переменной-указателем

Входные данные:

void main()

{

int a;

const float\* b = NULL;

if (a > 0)

a = 5;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | isConst:true  isPointer:true  type:float |
| identifier | b |
| isInitialized | true |
| value | "NULL" |
| visibilityArea | [3; 6] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 6] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

=

>

5

a

0

a

Тест 12. Функция main содержит много операторов

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

a++;

a+=8;

}

Выходные данные:

codeDeclaration

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 7] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

+=

++

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

a

8

a

>

=

a

0

a

5

Тест 13. Функция main имеет один входной параметр

Входные данные:

void main(int b)

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | variable |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

>

=

0

a

5

a

Тест 14. Функция main имеет два входных параметра

Входные данные:

void main(int b, float c)

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

variable:

|  |  |
| --- | --- |
| variableType | float |
| identifier | c |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | variable |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

>

=

0

a

5

a

Тест 15. Функция main имеет много входных параметров

Входные данные:

void main(int b, float c, char d)

{

int a;

if (a > 0)

a = 5;

}

Выходные данные:

variable:

|  |  |
| --- | --- |
| variableType | float |
| identifier | c |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | variable |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | char |
| identifier | d |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 5] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

>

=

0

a

5

a

Тест 16. Структура условного оператора содержит блоки if и else

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else

a = 6;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 7] |

=

>

5

a

0

a

=

a

6

Тест 17. Структура условного оператора содержит блоки if, else и один блок else-if

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

a = 1;

else

a = 6;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

=

>

5

a

0

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 9] |

1

a

=

0

a

==

=

a

6

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

Тест 18. Структура условного оператора содержит блоки if, else и два блока else-if

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

a = 1;

else if (a == 1)

a = 2;

else

a = 6;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

=

>

5

a

0

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

=

a

6

a

1

a

==

2

=

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

1

a

=

0

a

==

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

Тест 19. Состав блока if содержит несколько операторов

Входные данные:

void main()

{

int a;

if (a > 0)

{

a = 5;

a += 8;

a = a\*3;

}

else if (a == 0)

a = 1;

else

a = 6;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 13] |

\*

a

=

a

+=

8

=

5

a

>

0

a

a

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

1

a

=

0

a

==

=

a

6

Тест 20. Состав блока else содержит несколько операторов

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

a = 1;

else

{

a = 6;

a += 8;

a = a\*3;

}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 13] |

=

5

a

>

0

a

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | block-Statement |

1

a

=

0

a

==

=

a

6

\*

a

=

a

+=

8

a

3

Тест 21. Состав блока else-if содержит несколько операторов

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

{

a = 1;

a += 8;

a = a\*3;

}

else

a = 6;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 13] |

=

5

a

>

0

a

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock |  |

1

a

=

0

a

==

=

a

6

\*

a

=

a

+=

8

a

3

Тест 22. Программа с простым циклом for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

a+=8;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

a

++

<

a

5

a

=

8

a

100

Тест 23. Программа с простым циклом for без инициализации

Входные данные:

void main()

{

int a = 0;

for (; a < 100; a++)

a+=8;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization | NULL |
| condition |  |
| step |  |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

a

++

<

a

8

a

100

Тест 24. Программа с простым циклом for без условного выражения

Входные данные:

void main()

{

int a = 0;

for (a = 5;; a++)

a+=8;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition | NULL |
| step |  |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

a

++

5

a

=

8

a

Тест 25. Программа с простым циклом for без шага

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100;)

a+=8;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step | NULL |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

<

a

5

a

=

8

a

100

Тест 26. Программа с пустым циклом for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

a

++

<

a

5

a

=

100

Тест 27. Программа с циклом for c множеством операторов

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

Тест 28. Программа с простым циклом while

Входные данные:

void main()

{

int a = 0;

while (a < 0)

a+=8;

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | false |
| condition |  |
| whileBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

<

a

0

8

a

Тест 29. Программа с простым циклом do-while

Входные данные:

void main()

{

int a = 0;

do

a+=8;

while (a < 0);

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | true |
| condition |  |
| whileBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 6] |

+=

8

a

<

a

0

Тест 30. Программа с пустым циклом while

Входные данные:

void main()

{

int a = 0;

while (a < 0)

{}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | false |
| condition |  |
| whileBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

<

a

0

Тест 31. Программа с циклом while с множеством операторов

Входные данные:

void main()

{

int a = 0;

while (a < 0)

{

a+=8;

if (a < 50)

a–=5;

a\*=3;

a = a – 7;

}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | false |
| condition |  |
| whileBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

<

a

0

+=

8

a

5

a

–=

50

a

<

7

–

a

a

=

3

a

\*=

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

Тест 32. Программа со сложным выражением

Входные данные:

void main()

{

int a;

int b;

++a=b%3+(b++)\*(a–4);

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | false |
| value | "" |
| visibilityArea | [3; 5] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

-

4

a

++

a

3

b

%

\*

a

++

+

=

# Приложение Б

Тестирование функции performFunctionExtraction

Таблица 2. Аспекты тестирования функции performFunctionExtraction

| № | Аспект тестирования | Возможные случаи |
| --- | --- | --- |
| 1 | Состав выделенного фрагмента | - выделенный фрагмент не содержит операторов;  - выделенный фрагмент содержит один оператор;  - выделенный фрагмент содержит несколько операторов (2–4 оператора). |
| 2 | Структура выделенного фрагмента | - выделенный фрагмент является выражением;  - выделенный фрагмент является условным оператором if-else;  - выделенный фрагмент является циклом for;  - выделенный фрагмент является циклом while; |
| 3 | Местоположение выделенного фрагмента | - выделенный фрагмент находится внутри функции;  - находится внутри условного оператора;  - находится внутри цикла. |
| 4 | Выделенный фрагмент как часть условного оператора if-else | - выделенный фрагмент является условием блока if;  - выделенный фрагмент является условием блока else-if;  - выделенный фрагмент является частью блока if, else-if, else. |
| 5 | Выделенный фрагмент как часть цикла | - выделенный фрагмент является инициализацией цикла;  - выделенный фрагмент является условием продолжения цикла;  - выделенный фрагмент является шагом цикла;  - выделенный фрагмент является частью тела цикла. |
| 6 | Выделенный фрагмент как часть тела цикла | - выделенный фрагмент является телом цикла целиком;  - выделенный фрагмент является частью тела цикла, перед которым и после которого имеются другие операторы;  - выделенный фрагмент является частью условного оператора, который является частью тела цикла;  - выделенный фрагмент является частью тела цикла, который вложен в другой цикл. |
| 7 | Выделенный фрагмент как часть выражения | - выделенный фрагмент является полным оператором всего выражения;  - выделенный фрагмент является частью целого выражения. |
| 8 | Выделенный фрагмент содержит переопределения типа typedef | - выделенный фрагмент состоит из вызова переменной пользовательского типа, не включая определение typedef;  - выделенный фрагмент состоит из вызова переменной пользовательского типа, включая определение typedef. |
| 9 | Выделенный фрагмент содержит оператор break | - выделенный фрагмент содержит только оператор break;  - выделенный фрагмент содержит оператор break внутри условного оператора if-else;  - выделенный фрагмент содержит оператор break внутри цикла. |
| 10 | Выделенный фрагмент содержит оператор return | - оператор return как часть следования операторов;  - выделенный фрагмент является условным оператором if-else, где оператор return является его частью. |
| 11 | Выделенный фрагмент содержит операции над структурами | - структура является входным параметром выделяемой функции;  - структура является возвращаемым значением выделяемой функции. |
| 12 | Выделенный фрагмент содержит операции над массивами | - массив является входным параметром выделяемой функции;  - массив является выходным значением выделяемой функции. |

Входными данными функции performFunctionExtraction являются дерево разбора текста программы в структурированном формате, границы выделенного фрагмента и имя новой выделяемой функции. Выходными данными функции performFunctionExtraction являются дерево разбора текста программы в структурированном формате после проведения рефакторинга и признак успешности его проведения как возвращаемое значение.

Представление программы до и после проведения рефакторинга будет представлено схемами в виде деревьев, вложенности внутри которых будут изображаться стрелками (см. приложение А), а также к каждому дереву разбора программы будет прилагаться соответствующий код для наглядности.

Выделенный фрагмент будет выделен в тексте кода программы квадратными скобками (так будет показана нагляность выделения границ, которым соответствет входной параметр blockBounds).

fc

Вызов новой выделяемой функцией на схеме будет обозначаться .

Тест 1. Выделенный фрагмент не содержит операторов

Входные данные:

void main()

{

int a;

if (a > 0)

{

a = 5;

a += 8;

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

a

\*=

3

8

a

+=

=

5

a

0

a

>

blockBounds = [ <0,0>; <0;0> ]

functionName = "function";

Выходные данные:

Возвращаемое значение = false.

Тест 2. Выделенный фрагмент содержит один оператор

Входные данные:

void main()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

blockBounds = [ <5,2>; <5;8> ]

functionName = "function";

Выходные данные:

int function(int a)

{

a += 8;

return a;

}

void main()

{

int a;

if (a > 0)

{

a = function(a);

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | int |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

return:

|  |  |
| --- | --- |
| value | a |

+=

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

8

a

0

a

>

a

\*=

3

a

fc

Возвращаемое значение = true.

Тест 3. Выделенный фрагмент содержит несколько операторов

Входные данные:

void main()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

blockBounds = [ <5,2>; <6;8> ]

functionName = "function";

Выходные данные:

void function(int a)

{

a += 8;

a \*= 3;

}

void main()

{

int a = 5;

if (a > 0)

{

function(a);

}

}

codeDeclaration

8

a

+=

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

fc

a

0

a

>

Возвращаемое значение = true;

Тест 4. Выделенный фрагмент является условным оператором if-else

Входные данные:

void main()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

blockBounds = [ <3,1>; <7;1> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

void main()

{

int a = 5;

function();

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

fc

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

0

a

>

8

a

+=

a

\*=

3

Возвращаемое значение = true.

Тест 5. Выделенный фрагмент является циклом for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

a+=8;

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

a

++

<

a

5

a

=

8

a

100

blockBounds = [ <3,1>; <4;6> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 0;

for (a = 5; a < 100; a++)

a+=8;

}

void main()

{

function();

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

fc

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

a

++

<

a

5

a

=

8

a

100

Возвращаемое значение = true.

Тест 6. Выделенный фрагмент является циклом while

Входные данные:

void main()

{

int a = 0;

while (a < 0)

a+=8;

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | false |
| condition |  |
| whileBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

<

a

0

8

a

blockBounds = [ <3,1>; <4;6> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 0;

while (a < 0)

a+=8;

}

void main()

{

function();

}

codeDeclaration

fc

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

while:

|  |  |
| --- | --- |
| localVariables | NULL |
| isDoWhile | false |
| condition |  |
| whileBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 5] |

+=

<

a

0

8

a

Возвращаемое значение = true.

Тест 7. Выделенный фрагмент находится внутри блока else-if

Входные данные:

void main()

{

int a = 1;

if (a > 0)

a = 5;

else if (a == 0)

{

a += 8;

a = a\*3;

}

else

a = 6;

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "1" |
| visibilityArea | [2; 12] |

=

5

a

>

0

a

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock |  |

0

a

==

=

a

6

\*

a

=

a

+=

8

a

3

blockBounds = [ <7,2>; <8;9> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 1;

a += 8;

a = a\*3;

}

void main()

{

int a = 1;

if (a > 0)

a = 5;

else if (a == 0)

{

function();

}

else

a = 6;

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

a

\*

a

=

8

a

+=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "1" |
| visibilityArea | [2; 12] |

=

5

a

>

0

a

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock |  |

0

a

==

=

a

6

a

fc

Возвращаемое значение = true.

Тест 8. Выделенный фрагмент находится внутри блока else

Входные данные:

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

a = 1;

else

{

a = 6;

a += 8;

a = a\*3;

}

}

Выходные данные:

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 13] |

=

5

a

>

0

a

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | block-Statement |

1

a

=

0

a

==

=

a

6

\*

a

=

a

+=

8

a

3

blockBounds = [ <10,2>; <11,9> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 6;

a += 8;

a = a\*3;

}

void main()

{

int a;

if (a > 0)

a = 5;

else if (a == 0)

a = 1;

else

{

function();

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

a

\*

a

=

8

a

+=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "1" |
| visibilityArea | [8; 17] |

=

5

a

>

0

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "6" |
| visibilityArea | [2; 5] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | block-Statement |

0

a

==

=

a

1

a

fc

Возвращаемое значение = true.

Тест 9. Выделенный фрагмент является инициализацией цикла for

Входные данные:

void main()

{

int a = 0;

for ( a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <3,7>; <3,11> ]

functionName = "function";

Выходные данные:

int function()

{

int a = 0;

a = 5;

return a;

}

void main()

{

int a = 0;

for ( a = function(); a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

5

a

=

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

return:

|  |  |
| --- | --- |
| value | a |

a

++

<

a

fc

a

=

100

a

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

Возвращаемое значение = true.

Тест 10. Выделенный фрагмент является условием цикла for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <3,13>; <3,20> ]

functionName = "function";

Выходные данные:

bool function(int a)

{

return a < 100;

}

void main()

{

int a = 0;

for ( a = 5; function(a); a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

return:

|  |  |
| --- | --- |
| value | < |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

a

100

fc

a

++

5

a

=

a

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

Возвращаемое значение = true.

Тест 11. Выделенный фрагмент является шагом цикла for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <3,22>; <3,24> ]

functionName = "function";

Выходные данные:

int function(int a)

{

a++;

return a;

}

void main()

{

int a = 0;

for (a = 5; a < 100; a=function(a))

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

++

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

a

return:

|  |  |
| --- | --- |
| value | a |

<

a

=

5

a

=

fc

100

a

a

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

Возвращаемое значение = true.

Тест 12. Выделенный фрагмент является всем телом цикла for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <5,3>; <9,11> ]

functionName = "function";

Выходные данные:

int function(int a)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

return a;

}

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a=function(a);

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

fc

a

=

<

a

++

5

a

=

100

a

a

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

return:

|  |  |
| --- | --- |
| value | a |

Возвращаемое значение = true.

Тест 13. Выделенный фрагмент является частью тела цикла for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <5,3>; <7,7> ]

functionName = "function";

Выходные данные:

int function(int a)

{

a+=8;

if (a > 50)

a–=5;

return a;

}

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a=function(a);

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

return:

|  |  |
| --- | --- |
| value | a |

++

a

<

5

a

=

100

a

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

a

=

fc

a

Возвращаемое значение = true.

Тест 14. Выделенный фрагмент является телом условного оператора, который вложен в цикл for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

–=

5

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

blockBounds = [ <7,3>; <7,7> ]

functionName = "function";

Выходные данные:

int function(int a)

{

a–=5;

return a;

}

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

a=function(a);

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

return:

|  |  |
| --- | --- |
| value | a |

++

a

<

5

a

=

100

a

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

–=

5

a

a

=

fc

a

Возвращаемое значение = true.

Тест 15. Выделенный фрагмент является частью тела цикла, который вложен в другой цикл for

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

for (; a > 50; )

if (a > 50)

a–=5;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

7

a

–

a

=

a

\*=

3

+=

8

a

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization | NULL |
| condition |  |
| step | NULL |
| forBlock |  |

50

a

>

a

–=

5

a

>

50

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

blockBounds = [ <7,3>; <8,8> ]

functionName = "function";

Выходные данные:

int function(int a)

{

if (a > 50)

a–=5;

return a;

}

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

for(; a > 50; )

a=function(a);

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

return:

|  |  |
| --- | --- |
| value | a |

++

a

<

5

a

=

100

a

7

a

–

a

=

a

\*=

3

+=

8

a

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization | NULL |
| condition |  |
| step | NULL |
| forBlock |  |

50

a

>

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

a

fc

a

=

a

–=

50

a

>

5

Возвращаемое значение = true.

Тест 16. Выделенный фрагмент является частью одного выражения

void main()

{

int a = 3;

int b = 5;

++a=b%3+(b++)\*(a–4);

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | true |
| value | "5" |
| visibilityArea | [3; 5] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "3" |
| visibilityArea | [2; 5] |

4

a

a

–

++

3

b

\*

%

a

+

++

=

blockBounds = [ <4,9>; <4,20> ]

functionName = "function";

Выходные данные:

int function(int a)

{

return (b++)\*(a–4);

}

void main()

{

int a = 3;

int b = 5;

++a=b%3+function(a);

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | int |
| identifier | functin |
| arguments | variable |
| localVariables | variable |
| functionBody |  |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | b |
| isInitialized | true |
| value | "5" |
| visibilityArea | [3; 5] |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "3" |
| visibilityArea | [2; 5] |

3

b

fc

%

a

+

++

=

return:

|  |  |
| --- | --- |
| value | \* |

4

a

b

–

++

a

Возвращаемое значение = true.

Тест 17. Выделенный фрагмент содержит переопределение типа typedef

void main()

{

typedef int number;

number a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | number |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

blockBounds = [ <3,1>; <8;1> ]

functionName = "function";

Выходные данные:

void function()

{

typedef int number;

number a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

}

}

void main()

{

typedef int number;

function();

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | NULL |
| functionBody |  |

fc

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

0

a

>

8

a

+=

a

\*=

3

Возвращаемое значение = true.

Тест 18. Выделенный фрагмент содержит только один оператор break

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

break;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

break

blockBounds = [ <7,3>; <7,7> ]

functionName = "function";

Выходные данные:

Возвращаемое значение = false;

Тест 19. Выделенный фрагмент содержит оператор break, вложенный в условный оператор if-else

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

break;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

break

blockBounds = [ <6,3>; <7,7> ]

functionName = "function";

Выходные данные:

Возвращаемое значение = false;

Тест 20. Выделенный фрагмент содержит только оператор break, вложенный в цикл

Входные данные:

void main()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

break;

a\*=3;

a = a – 7;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 11] |

a

++

<

a

5

a

=

100

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

break

blockBounds = [ <3,3>; <10,3> ]

functionName = "function";

Выходные данные:

void function()

{

int a = 0;

for (a = 5; a < 100; a++)

{

a+=8;

if (a > 50)

break;

a\*=3;

a = a – 7;

}

}

void main()

{

function();

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

for:

|  |  |
| --- | --- |
| localVariables | NULL |
| initialization |  |
| condition |  |
| step |  |
| forBlock | block-Statement |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "0" |
| visibilityArea | [2; 11] |

++

a

<

5

a

=

fc

100

a

a

>

50

7

a

–

a

=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock |  |
| elseBlock | NULL |

+=

8

a

break

Тест 21. Выделенный фрагмент содержит return, как часть следования операторов

Входные данные:

void main()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

return;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

return

blockBounds = [ <5,2>; <7;7> ]

functionName = "function";

Выходные данные:

void function(int a)

{

a += 8;

a \*= 3;

return;

}

void main()

{

int a = 5;

if (a > 0)

{

function();

return;

}

}

codeDeclaration

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

return

8

a

+=

a

\*=

3

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

fc

0

a

>

return

a

Возвращаемое значение = true.

Тест 22. Выделенный фрагмент содержит return, как часть условного оператора

Входные данные:

void main()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

return;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

a

\*=

3

8

a

+=

0

a

>

return

blockBounds = [ <3,1>; <8;1> ]

functionName = "function";

Выходные данные:

bool function()

{

int a = 5;

if (a > 0)

{

a += 8;

a \*= 3;

return true;

}

return false;

}

void main()

{

int a = 5;

if (function())

{

return;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | NULL |
| localVariables | variable |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock |  |

fc

return

0

a

>

8

a

+=

a

\*=

3

return:

|  |  |
| --- | --- |
| value | false |

return:

|  |  |
| --- | --- |
| value | true |

Возвращаемое значение = true.

Тест 23. Выделенный фрагмент содержит элемент структуры

Входные данные:

struct str

{

int strA;

};

void main()

{

str s;

int a = 5;

if (a > 0)

{

s.strA = 8;

a \*= 3;

}

}

codeDeclaration

struct:

|  |  |
| --- | --- |
| identifier | str |
| member | variable |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | strA |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 3] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [7; 13] |

variable:

|  |  |
| --- | --- |
| variableType | str |
| identifier | s |
| isInitialized | false |
| value | "" |
| visibilityArea | [6; 13] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

a

\*=

3

8

.

=

0

a

>

strA

s

blockBounds = [ <10,2>; <11;7> ]

functionName = "function";

Выходные данные:

struct str

{

int strA;

};

void function(str s, int a)

{

s.strA = 8;

a \*= 3;

}

void main()

{

str s;

int a = 5;

if (a > 0)

{

function(s, a);

}

}

codeDeclaration

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | strA |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 3] |

struct:

|  |  |
| --- | --- |
| identifier | str |
| member | variable |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

8

=

a

\*=

3

s

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

fc

s

0

a

>

a

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

variable:

|  |  |
| --- | --- |
| variableType | str |
| identifier | s |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

.

strA

Возвращаемое значение = true.

Тест 24. Выделенный фрагмент содержит элемент структуры, которая будет являться возвращаемым значением

Входные данные:

struct str

{

int strA;

};

void main()

{

str s, s1;

int a = 5;

if (a > 0)

{

s.strA = 8;

a \*= 3;

}

s1 = s;

}

codeDeclaration

struct:

|  |  |
| --- | --- |
| identifier | str |
| member | variable |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | strA |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 3] |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [7; 13] |

variable:

|  |  |
| --- | --- |
| variableType | str |
| identifier | s |
| isInitialized | false |
| value | "" |
| visibilityArea | [6; 13] |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

a

\*=

3

8

.

=

0

a

>

strA

s

s1

=

variable:

|  |  |
| --- | --- |
| variableType | str |
| identifier | s1 |
| isInitialized | false |
| value | "" |
| visibilityArea | [6; 13] |

s

blockBounds = [ <10,2>; <11;7> ]

functionName = "function";

Выходные данные:

struct str

{

int strA;

};

str function(str s, int a)

{

s.strA = 8;

a \*= 3;

return s;

}

void main()

{

str s, s1;

int a = 5;

if (a > 0)

{

s = function(s, a);

}

s1 = s;

}

codeDeclaration

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | strA |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 3] |

struct:

|  |  |
| --- | --- |
| identifier | str |
| member | variable |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

function:

|  |  |
| --- | --- |
| returnType | str |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

8

=

a

\*=

3

strA

s

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

>

s

=

a

s

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

variable:

|  |  |
| --- | --- |
| variableType | str |
| identifier | s |
| isInitialized | false |
| value | "" |
| visibilityArea | [0; 4] |

0

a

fc

.

return:

|  |  |
| --- | --- |
| value | s |

Возвращаемое значение = true.

Тест 25. Выделенный фрагмент содержит элемент массива

Входные данные:

void main()

{

int a[5];

a[3] = 5;

if (a[3] > 0)

{

a[3] = 8;

a[3] \*= 3;

}

}

codeDeclaration

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 8] |
| massDimension | "5" |

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

5

=

0

>

\*=

3

8

+=

a[3]

a[3]

a[3]

a[3]

blockBounds = [ <5,2>; <6;8> ]

functionName = "function";

Выходные данные:

void function(int a[5])

{

a[3] = 8;

a[3] \*= 3;

}

void main()

{

int a[5];

a[3] = 5;

if (a[3] > 0)

{

function(a);

}

}

codeDeclaration

8

+=

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | function |
| arguments | variable |
| localVariables | NULL |
| functionBody | block-Statement |

function:

|  |  |
| --- | --- |
| returnType | void |
| identifier | main |
| arguments | NULL |
| localVariables | variable |
| functionBody |  |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | false |
| value | "" |
| visibilityArea | [2; 8] |
| massDimension | "5" |

variable:

|  |  |
| --- | --- |
| variableType | int |
| identifier | a |
| isInitialized | true |
| value | "5" |
| visibilityArea | [2; 8] |

\*=

if:

|  |  |
| --- | --- |
| localVariables | NULL |
| condition |  |
| ifThenBlock | block-Statement |
| elseBlock | NULL |

3

a[3]

a[3]

0

>

fc

a

5

a[3]

=

a[3]

Возвращаемое значение = true.