# Лабораторная работа 1

Версия 0.8

*Использование утилиты gcov для оценки полноты тестовых данных*

Утилита *gcov* используется (в том числе) для анализа покрытия кода тестами. Информация, которая получается в результате работы этой утилиты, позволяет ответить на следующие вопросы:

- как часто выполняется каждая строка кода;
- какие строки кода не выполняются;
- какой процент строк кода покрыт тестами (для каждого файла);
- какой процент ветвлений покрыт тестами (для каждого файла).

*gcov* обрабатывает программы, которые были получены только с помощью компилятора *gcc*. При сборке программы желательно выключить оптимизацию с помощью ключа “-O0” и добавить следующие ключи “-fprofile-arcs” и “-ftest-coverage”.

*Замечание*
В современных версиях компиляторов ключи “-fprofile-arcs” и “-ftest-coverage” могут быть заменены ключом “--coverage”

Два последних ключа вынуждают компилятор добавить в исполняемый файл программы дополнительный код, который во время ее выполнения собирает статистическую информацию и сохраняется ее в служебные файлы. *gcov* использует эти файлы для создания «аннотированного» листинга исходного кода программы, который содержит информацию о частоте выполнения каждой строки.

Рассмотрим шаги использования утилиты *gcov* на примере исследования тестовых данных для следующей программы.

Пример 1. Файл sq.c

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float a, b, c, d;

    printf("Enter a, b, c: ");
    if (scanf("%f%f%f", &a, &b, &c) == 3)
    {
        if (a != 0.0)
        {
            d = b * b - 4 * a * c;

            if (d < 0.0)
            {
                printf("There are no real roots\n");
            }
            else if (d > 0)
            {
                printf("x1 = %f, x2 = %f\n", (-b - sqrt(d)) / (2 * a),
                                             (-b + sqrt(d)) / (2 * a));
            }
            else
            {
```

```
                printf("x1 = x2 = %f\n", -b / (2 * a));
            }
        }
        else
        {
            printf("Equation is not square\n");
        }
    }
    else
    {
        printf("I/O error\n");
    }

    return 0;
}
```

Для компиляции программы нужно выполнить следующую команду

```
$ gcc -std=c99 -Wall -Werror -O0 -fprofile-arcs -ftest-coverage sq.c -o sq.exe
```

В случае успешного выполнения компиляции в папке появится не только исполняемый файл sq.exe, но и файл sq.gcno. Этот файл содержит информацию необходимую для работы утилиты *gcov* (например, граф базовых блоков и др.).

Программа запускается как обычно. Например, с такими входными данными

```
$ ./sq.exe
Enter a, b, c: 1 2 1
x1 = x2 = -1.000000
```

После запуска программы в папке, в которой расположен исполняемый файл, должен появится файл с расширение “gcda” (в нашем случае это sq.gcda). В этом файле «накапливается» статистика выполнения кода для каждого файла.

После выполнения программы хотя бы раз можно поучить «аннотированный» листинг. Делается это с помощью утилиты *gcov* следующим образом

```
$ gcov sq.c
File 'sq.c'
Lines executed:64.29% of 14
Creating 'sq.c.gcov'
```

*Замечание*
Если программа состоит из нескольких файл, «аннотированный» листинг нужно получить для каждого.

Утилита gcov выдаст на экран указанную информацию и создаст «аннотированный» листинг sq.c.gcov. Возможно, что «аннотированный» листинг будет включать в себя несколько файлов.

Аннотированный листинг для файла sc.c

```
    -:    0:Source:sq.c
    -:    0:Graph:sq.gcno
    -:    0:Data:sq.gcda
    -:    0:Runs:1
    -:    0:Programs:1
    -:    1:#include <stdio.h>
    -:    2:#include <math.h>
    -:    3:
    1:    4:int main(void)
```

```
    -:    5:{
    -:    6:    float a, b, c, d;
    -:    7:
    1:    8:    printf("Enter a, b, c: ");
    1:    9:    if (scanf("%f%f%f", &a, &b, &c) == 3)
    -:   10:    {
    1:   11:        if (a != 0.0)
    -:   12:        {
    1:   13:            d = b * b - 4 * a * c;
    -:   14:
    1:   15:            if (d < 0.0)
    -:   16:            {
#####:   17:                printf("There are no real roots\n");
    -:   18:            }
    1:   19:            else if (d > 0)
    -:   20:            {
#####:   21:                printf("x1 = %f, x2 = %f\n", (-b - sqrt(d)) / (2 * a),
#####:   22:                                             (-b + sqrt(d)) / (2 * a));
    -:   23:            }
    -:   24:            else
    -:   25:            {
    1:   26:                printf("x1 = x2 = %f\n", -b / (2 * a));
    -:   27:            }
    -:   28:        }
    -:   29:        else
    -:   30:        {
#####:   31:            printf("Equation is not square\n");
    -:   32:        }
    -:   33:    }
    -:   34:    else
    -:   35:    {
#####:   36:        printf("I/O error\n");
    -:   37:    }
    -:   38:
    1:   39:    return 0;
    -:   40:}
```

Изучим содержание «аннотированного» листинга. Он начинается «преамбулой», каждая строка которой имеет вид:

```
-:0:<tag>:<value>
```

Из «преамбулы» нас будет интересовать количество запусков программы:

```
-:    0:Runs:1
```

У остальных строк «аннотированного» листинга формат следующий:

```
<количество выполнений строки>:<номер строки>:<строка исходного кода>
```

Для строк, которые не содержат код, количество выполнений обозначается символом “-”, а для строк, которые не выполнялись ни разу, - “#####”.

Повторяя запуск программы с различными входными данными и анализируя «аннотированный» листинг после каждого запуска, можно добиться полного покрытия кода тестами.

*Замечание*
Следует помнить, что даже 100% покрытие кода тестами не означает, что в программе нет ошибок!

Полезные ключи утилиты *gcov*

| Ключ | Назначение |
|---|---|
| -a | Дополнит «аннотированный» листинг информацией о выполнении всех базовых блоков каждой строки программы, а не только основного. |
| -b | Дополнит «аннотированный» листинг информацией статистикой о выполнении условных операторов. |
| -f | Выведет суммарную статистику по каждой функции программы. |

*Вопросы для самопроверки*

1. Напишите небольшую программу, которая содержит ошибку. Подготовьте для нее такие тестовые данные, которые с одной стороны обеспечивают полное покрытие кода, с другой стороны не выявляют ошибку, которая есть в программе. В отчете необходимо привести прокомментированную программу, тестовые данные и описание ошибки.
2. Изучите дополнительные ключи утилиты *gcov*.
    a. Придумайте пример, когда может быть полезен ключ “-a”. Пример поместите в отчет.
    b. Какая дополнительная информация появляется в «аннотированном» листинге при использовании ключа “-b”? Можно ли ее каким-то образом использовать для облегчения работы над тестовыми данными? Если да, опишите как.