Целью работы является разработка метода статического поиска гонок в программах на языке Си. Для этого необходимо решить следующие задачи:

* Выполнить анализ методов поиска гонок в программах.
* Разработать метод статического поиска гонок при доступе к разделяемой памяти.
* Разработать алгоритм статического поиска гонок на основе предложенного метода.
* Разработать ПО, реализующее предлагаемый метод.
* Провести исследование разработанного метода.

Под ***состоянием гонки при доступе к разделяемой области памяти*** будем понимать ситуацию, когда несколько потоков одновременно совершают доступ к общей области памяти, и хотя бы один из них выполняет операцию записи в неё. На слайде показан пример, в котором возможно возникновение гонок при доступе к глобальной переменной count, т.к. доступ к ней не защищен никакими средствами взаимоисключения.

Существует 3 основные группы методов поиска гонок в программах:

* Формальная верификация, основная идея которой заключается в установлении соответствия между программой и требованиями к программе, описывающими цель разработки.
* Динамические методы, основанные на изучении потока событий, генерируемых программой во время выполнения. Недостаток - состояние гонки может быть зафиксировано, только если оно возникло в проверяемом варианте исполнения программы. Достоинство – если состояние гонки было зафиксировано, то оно, скорее всего, является таковым.
* Статические методы, основанные на анализе исходного кода программы. Достоинство - теоретическая возможность анализа всех возможных путей выполнения программы. Недостаток – много ошибок 2-го рода. Основная идея статического поиска гонок заключается в том, чтобы удостовериться, что для каждой общей области памяти существует, по крайней мере, одна блокировка, которая захватывается во всех потоках при доступе к этой области памяти. Основные подходы: аннотирование кода, которое выполняется за счёт добавления в исходный код программы специальных конструкций, содержащих информацию об объектах взаимоисключения и обеспечивающих защиту доступа к переменным и вызовам функций, и анализ потока выполнения, осуществляемый на основе анализа всевозможных путей выполнения программы.