Прочитайте эти заметки после появления у вас некоторого опыта программирования на C#. Вспомните, пришлось ли вам справиться с ошибками, описанными ниже.

**Простейшая классификация ошибок (errors):**

* **Синтаксические** (ошибки в операторах) – компилятор их обнаруживает и указывает возможную причину. Пока вы их не устраните, программа выполняться не будет.
* **Логические** (нарушение логики алгоритма решения задачи) – компилятор их пропускает, так как проверяет только синтаксис программы.
* Ошибки, возникающие **во время выполнения** программы (*Run-time error*). Такая ошибка генерирует исключение (*exception*), которое без обработки приводит к остановке программы с указанием возможной причины.

**Вот типичные ошибки, которые делает почти каждый программист.**

**Синтаксические ошибки:**

* Если вы забыли вставить или случайно закомментировали оператор подключения библиотеки (например, using System;), то для вас будут недоступны поля и методы класса Console и всех других классов. Хорошо, что компилятор подскажет вам эту ошибку: «Элемент «Console» не существует в текущем контексте».
* Пренебрежение точкой с запятой в конце оператора является синтаксической ошибкой, которая возникает тогда, когда компилятор не может распознать оператор. Обычно выдается сообщение «Требуется “;”».
* Если операции ==, !=, >=, <= содержат между своими символами пробелы, это ошибка.
* Смешивание операций проверки на равенство == с операцией присваивания приводит к ошибкам, как синтаксическим, так и логическим.
* Использование ключевого слова в качестве идентификатора переменной.
* Пропуск одной или обеих фигурных скобок, ограничивающих составной оператор.
* Запись ключевого слова ***while*** как While с символом W в верхнем регистре приведет к синтаксической ошибке. Заметим, что все ключевые слова содержат только символы нижнего регистра.
* Попытка использовать в операции инкремента не имя простой переменной, а выражение, например,  “++(d+2)”.
* Использование запятых вместо точек с запятой в заголовке цикла for.
* Вызов математических функций по имени (например, Sin(x)) без указания в качестве префикса имени класса Math. Нужно вызывать так: Math.Sin(x). То же касается и методов других классов: Console, Convert и тому подобных.

**Логические ошибки:**

* Постановка точки с запятой после правой круглой скобки, завершающей условие в операторе if, приводит к тому, что тело структуры if становится пустым. Подлинный оператор тела структуры if превращается в оператор, следующий за структурой if, и выполняется всегда.
* В теле структуры while не предусматривается действие, которое со временем приведет к тому, что условие станет ложным. Возникнет «зацикливание» программы, которое может продолжаться бесконечно долго.
* Если для счетчика или переменной суммы не заданы начальные значения, то результат работы вашей программ будет вероятно неправильным.
* Использование чисел с плавающей запятой в предположении, что они совершенно точные, может приводить к некорректным результатам.
* Использование неправильной операции отношения или использование неправильной конечной величины счетчика цикла в условиях структур for или while может привести к ошибке занижения (завышения) на единицу.
* Размещение точки с запятой сразу после правой закрывающей скобки заголовка for делает оператор цикла бессмысленным.
* Пропуск пробела между ключевым словом case и целым значением, которое проверяется в цикле switch. Запись *case5:* (вместо *case 5:*) просто создаст неиспользуемую метку, но не будут совершены действия, когда управляющее выражение switch будет равно 5.

**Примеры Run-time ошибок:**

* Ошибка деления на нуль возникает во время выполнения программы. Приводит к остановке программы с сообщением «Попытка деления на нуль. DivideByZeroException не обработано». В этом случае ошибка считается «неисправимой» (необходимо изменение программы!).
* Пример «исправимой» ошибки: результатом умножения двух положительных целых чисел 1000000 и 1100 будет целое отрицательное число, равное -094967296 (исключение — «переполнение»), при этом программа продолжит свое выполнение после оператора умножения.

Так что подумайте, какой вид ошибок хуже, и как их не делать.