1 Go: просто научиться, но сложно освоить

В этой главе:

- ✓ Что делает Go эффективным, масштабируемым и производительным языком
- ✓ Почему языку Go просто научиться, но овладеть им по-настоящему сложно
- ✓ Общее описание распространенных типов ошибок, допускаемых разработчиками

Ошибаться свойственно всем. Как сказал Альберт Эйнштейн:

Тот, кто никогда не совершал ошибок, тот никогда не пробовал что-то новое.

В конце концов, важно не количество совершенных ошибок, а наша способность учиться на них. Это утверждение относится и к программированию. Мастерство, которое мы приобретаем, — это не волшебство. Мы делаем множество ошибок и учимся на них. Это основная мысль книги. Мы рассмотрим и изучим 100 распространенных ошибок, которые допускаются во многих сферах использования языка Go, и это поможет вам стать более опытным программистом.

В главе 1 мы кратко расскажем, почему Go с годами стал одним из основных и стандартных инструментов работы. Мы обсудим, почему, несмотря на то что Go считается простым в изучении, овладение его нюансами может быть весьма сложным. Наконец, познакомимся с основными понятиями из этой книги.

1.1. GO: ОСНОВНЫЕ МОМЕНТЫ

Если вы читаете нашу книгу, то, скорее всего, уже «подсели» на Go. Поэтому в этом разделе будет только краткий обзор, призванный напомнить, что делает Go таким мощным языком.

Отрасль разработки программного обеспечения (ПО) за последние десятилетия значительно изменилась. Большинство современных систем больше не создается одним человеком. Все они — результат работы команд, состоящих из многих программистов, а иногда даже из сотен, если не тысяч. Написанный программный код должен быть читабельным, выразительным, удобным в сопровождении, чтобы обеспечивать надежную работу системы на протяжении многих лет. С другой стороны, в нашем быстро меняющемся мире максимальное повышение гибкости и сокращение времени выхода на рынок очень важны для большинства компаний. Программирование тоже должно следовать этой тенденции, поэтому компании стремятся к тому, чтобы программисты работали максимально продуктивно при чтении, написании и сопровождении кода.

В ответ на эти вызовы и требования в 2007 году компания Google создала язык Go. С тех пор многие организации приняли его для использования в различных областях программирования: в API, автоматизации, базах данных, интерфейсах командной строки и т. д. Сегодня многие считают Go одним из основных языков для разработки облачных систем.

Что касается функциональности, то в Go нет наследования типов, исключений, макросов, частичных функций, поддержки ленивых вычислений или неизменяемости, перегрузки операторов, сопоставления шаблонов и т. д. Почему? Вот что об этом говорит официальный FAQ по Go (https://go.dev/doc/faq):

> *Почему в Go нет какой-то функции X? Ваша любимая функция может отсутствовать, поскольку не вписывается в логику или структуру языка, влияет на скорость компиляции или ясность дизайна кода либо просто потому, что сделала бы фундаментальную модель системы слишком сложной.*

Оценка качества языка программирования на основании количества функций в нем, вероятно, некорректна. По крайней мере для Go эта метрика не главная.

При оценке адекватности использования языка в масштабе какой-то организации используют несколько важных характеристик. К ним относятся:

- *Стабильность*. Несмотря на то что в Go вносятся частые изменения (направленные на улучшение самого языка и устранение уязвимостей с точки зрения безопасности), он остается достаточно стабильным языком. Некоторые считают это качество одной из лучших особенностей языка.
- *Выразительность*. Мы можем определить выразительность языка по тому, насколько написание и чтение кода отвечает представлениям о естественности и интуитивной понятности. Уменьшенное количество ключевых слов и ограниченные способы решения общих проблем делают Go выразительным языком для больших кодовых баз.
- *Компиляция*. Что может быть более раздражающим для разработчиков, чем долгое ожидание сборки для тестирования приложения? Стремление к быстрой компиляции всегда было сознательной целью разработчиков языка. А это основа высокой производительности.
- *Безопасность*. Go — надежный язык со статической типизацией. Следовательно, у него есть строгие правила времени компиляции, которые в большинстве случаев обеспечивают безопасность типов.

Go был создан с нуля с очень полезными функциями: с примитивами конкурентности, горутинами и каналами. Ему особо не нужно полагаться на внешние библиотеки для создания эффективных конкурентных приложений. Наблюдение за тем, насколько важна конкурентность в наши дни, также показывает, почему Go сейчас самый подходящий язык и будет оставаться им в обозримом будущем.

Некоторые считают Go простым языком, и отчасти это правда. Например, новичок может разобраться с его основными возможностями менее чем за один день. Возникает вопрос: зачем же изучать книгу, посвященную систематизации ошибок в Go, если он так прост?

1.2. ПРОСТО НЕ ОЗНАЧАЕТ ЛЕГКО

Между понятиями «просто» и «легко» есть тонкая разница. «Простой» применительно к технологии означает несложный для изучения или понимания. «Легкость» означает возможность добиваться чего угодно без особых усилий. Go прост в изучении, но не всегда легок в освоении.

Возьмем, к примеру, конкурентность. В 2019 году было опубликовано исследование, посвященное ошибкам конкурентности: «Понимание реальных ошибок

конкурентности в Go»[1]. Это исследование было первым систематическим анализом ошибок конкурентности. Оно опиралось на данные нескольких популярных репозиториев Go — Docker, gRPC и Kubernetes. Один из самых важных выводов заключается в том, что большинство блокирующих ошибок вызвано неточным использованием парадигмы передачи сообщений (message passing) по каналам, несмотря на убеждение, что передача сообщений легче обрабатывается и менее подвержена ошибкам, чем разделяемая память.

Какой должна быть реакция на такой вывод? Должны ли мы считать, что разработчики языка ошибались насчет передачи сообщений? Должны ли мы пересмотреть использование конкурентности в нашем проекте? Конечно нет.

Это не вопрос противопоставления передачи сообщений разделяемой памяти и выявления из них «победителя». Но разработчики Go должны хорошо понимать, как использовать конкурентность, каково ее влияние на современные процессоры, когда следует предпочесть один подход другому и как избежать при этом попадания в типичные ловушки. Этот пример подчеркивает, что хотя каналы и горутины могут быть простыми для изучения, на практике это совсем не просто.

Понятие «просто не значит легко» можно обобщить на многие аспекты Go, а не только на конкурентность. И чтобы стать опытными Go-разработчиками, нужно хорошо разбираться во всех его аспектах. А это требует времени, усилий и ошибок.

Цель книги — помочь ускорить наш путь к мастерству, рассмотрев 100 ошибок в Go.

1.3. 100 ОШИБОК В GO

Почему следует прочитать эту книгу? Почему бы вместо этого не углубить знания с помощью «обычной» книги, которая достаточно подробно рассматривает разные темы?

В статье, опубликованной в 2011 году, нейробиологи доказали, что столкновение с ошибками — это лучшие моменты для развития способностей нашего мозга[2].

---

[1] T. Tu, X. Liu, et al. (с соавторами), *Understanding Real-World Concurrency Bugs in Go*, работа была представлена на ASPLOS 2019, April 13–17, 2019.

[2] J. S. Moser, H. S. Schroder, с соавторами, “Mind Your Errors: Evidence for a Neural Mechanism Linking Growth Mindset to Adaptive Posterror Adjustments,” Psychological Science, vol. 22, no. 12, pp. 1484–1489, Dec. 2011.

Все мы проходили через процесс обучения на какой-то ошибке, вспоминая этот случай через месяцы или даже годы, когда с ним был связан какой-то контекст. В статье Джанет Меткалф (Janet Metcalfe) говорится, что это происходит потому, что ошибки оказывают стимулирующее воздействие[1]. Суть в том, что мы можем помнить не только саму ошибку, но и ее контекст. И поэтому обучение на ошибках так эффективно.

Чтобы усилить этот эффект, в книге каждая рассматриваемая типичная ошибка подкреплена примерами из реальной практики. Эта книга не только о теории, она поможет избежать ошибок и принимать взвешенные, осознанные решения.

Скажи мне, и я забуду. Научи меня, и я запомню. Вовлеки меня, и я научусь.

Неизвестный автор

Здесь представлены семь основных категорий ошибок, которые можно классифицировать как:

- баги;
- излишнюю сложность;
- плохую читаемость;
- неоптимальную или неидиоматическую организацию;
- отсутствие удобства в API;
- неоптимизированный код;
- недостаточную производительность.

Далее я дам краткое описание каждой категории ошибок.

1.3.1. Баги

Первый и, возможно, самый очевидный тип — это ошибки в исходном коде. В 2020 году исследование, проведенное Synopsys, оценило стоимость багов в ПО только в США более чем в 2 триллиона долларов[2].

---

[1] J. Metcalfe, “Learning from Errors,” Annual Review of Psychology, vol. 68, pp. 465–489, Jan. 2017.

[2] Synopsys, “The Cost of Poor Software Quality in the US: A 2020 Report.” 2020. https://news.synopsys.com/2021-01-06-Synopsys-Sponsored-CISQ-Research-Estimates-Cost-of-Poor-Software-Quality-in-the-US-2-08-Trillion-in-2020.

Баги могут приводить и к трагическим последствиям. Вспомним случай с аппаратом для лучевой терапии Therac-25 производства компании Atomic Energy of Canada Limited (AECL). Из-за состояния гонки машина дала своим пациентам дозы облучения, которые в сотни раз превышали ожидаемые, что привело к смерти трех пациентов. Этот пример показывает, что баги могут повлечь за собой не только денежные потери. И мы, как разработчики, должны помнить, насколько важна наша работа.

Я рассмотрю множество случаев, которые могут привести к различным багам, включая гонки данных, утечки, логические ошибки и др. Хотя точные тесты и должны обнаруживать такие ошибки как можно раньше, иногда мы можем пропускать их из-за различных факторов, например из-за нехватки времени или их сложности. И разработчику важно убедиться, что для устранения таких багов сделано все возможное.

1.3.2. Излишняя сложность

Следующая категория ошибок связана с излишней сложностью. Значительная часть сложности ПО вызвана тем, что разработчики стремятся думать о своем воображаемом будущем. Вместо того чтобы решать конкретные задачи прямо сейчас, может возникнуть соблазн создать «эволюционирующее» ПО, которое будет пригодным для любого будущего варианта использования. В большинстве случаев это приводит к тому, что объем недостатков превышает число преимуществ, что делает код сложным для понимания и анализа.

Возвращаясь к Go, можно вспомнить множество примеров того, как у разработчиков возникает соблазн разработать абстрактные функции для будущего, например интерфейсы или дженерики. В этой книге обсуждаются примеры, когда следует проявлять особую осторожность, чтобы не переусложнить код.

1.3.3. Плохая читаемость

Как написал Роберт Мартин (Robert Martin) в книге «Clean Code: A Handbook of Agile Software Craftsmanship»[1], соотношение времени, затрачиваемого на чтение и написание кода, значительно превышает 10 : 1. Большинство из нас начинали программировать в собственных проектах, где удобочитаемость не так важна. Но сегодняшняя разработка ПО — это программирование во временно́м

[1] Мартин Р. «Чистый код: создание, анализ и рефакторинг». Санкт-Петербург, издательство «Питер».

измерении: нужно убедиться, что с приложением все еще можно работать и поддерживать его спустя месяцы, годы или, возможно, даже десятилетия после релиза.

При программировании на Go можно наделать много ошибок, которые затруднят читаемость кода. Среди таких ошибок может быть и вложенный код, и представления типов данных, а иногда и использование неименованных результирующих параметров. На протяжении этой книги мы будем учиться писать читаемый код и заботиться о его будущих читателях (в частности, о себе).

1.3.4. Неоптимальная или неидиоматическая организация

Другой тип ошибки — это неоптимальная или неидиоматическая организация кода и проекта. Такие проблемы могут затруднить анализ и дальнейшую поддержку проекта. В этой книге рассмотрены некоторые из распространенных ошибок такого рода. Например, мы увидим, как структурировать проект и обращаться с пакетами утилит или функциями инициализации. Рассмотрение этих ошибок поможет организовать код и проекты более эффективно и идиоматично.

1.3.5. Отсутствие удобства в API

Распространенные ошибки, снижающие удобство API для наших потребителей, — это еще один тип. Если API неудобен для пользователя, он будет менее выразительным и, следовательно, более трудным для понимания и более подверженным дальнейшим ошибкам.

Такие ошибки встречаются во многих ситуациях и могут заключаться в чрезмерном использовании типа `any`, в использовании неправильных порождающих паттернов при работе с опциями или в слепом применении стандартных методов объектно-ориентированного программирования, что влияет на удобство использования API. Мы рассмотрим распространенные ошибки, мешающие передавать в распоряжение наших пользователей удобные для них API.

1.3.6. Неоптимизированный код

Код, оптимизированный в недостаточной степени, — еще один тип ошибок разработчиков. Их можно сделать по разным причинам, например из-за непонимания особенностей языка или даже из-за отсутствия фундаментальных знаний.

Недостаточная производительность — одно из наиболее очевидных последствий этой ошибки, но не единственное.

Оптимизация кода полезна и для точности. Например, в этой книге представлены некоторые распространенные методы, обеспечивающие высокую точность операций с плавающей точкой. Мы также рассмотрим множество случаев, которые могут негативно сказаться на производительности кода, например, из-за недостаточного распараллеливания задач, незнания того, как уменьшать использование ресурсов памяти, или влияния выравнивания данных. Поговорим о вопросах оптимизации под разными углами.

1.3.7. Недостаточная производительность

В большинстве случаев мы задаемся вопросом: какой язык лучше всего выбрать для конкретного нового проекта? Ответ: тот, с которым мы работаем наиболее продуктивно. Для достижения мастерства очень важно знать, как работает язык, и использовать его по максимуму.

Мы рассмотрим конкретные примеры, которые помогут стать продуктивными при работе на Go. Например, написание эффективных тестов для обеспечения работоспособности кода, использование стандартной библиотеки для повышения эффективности, а также извлечение максимальной пользы из инструментов профилирования и линтеров. Пришло время разобраться в этих 100 распространенных ошибках Go!

ИТОГИ

- Go — это современный язык программирования, который позволяет повысить производительность разработчиков, что сегодня крайне важно для большинства компаний.
- Go прост в изучении, но нелегок в освоении. Поэтому важно углубить свои знания, чтобы использовать его наиболее эффективно.
- Обучение на разборе ошибок и на конкретных примерах — это мощный способ овладеть языком. Книга на примерах разбора 100 распространенных ошибок ускорит путь к профессиональному мастерству.