# Предисловие

> “Язык Go является языком программирования с открытым кодом, что делает его простым в создании, надежным и эффективным программным продуктом”. (С сайта Go: `golang.org`.)

Go был задуман в сентябре 2007 года Робертом Грисемером (Robert Griesemer), Робом Пайком (Rob Pike) и Кеном Томпсоном (Ken Thompson) из Google и анонсирован в ноябре 2009 года. Целью разработки было создание выразительного, высокоэффективного как при компиляции, так и при выполнении программ языка программирования, позволяющего легко и просто писать надежные высокоинтеллектуальные программы.

Go имеет поверхностное сходство с языком программирования C и обладает тем же духом инструментария для серьезных профессиональных программистов, предназначенного для достижения максимального эффекта с минимальными затратами. Но на самом деле Go — это нечто гораздо большее, чем просто современная версия языка программирования C. Он заимствует и приспосабливает для своих нужд хорошие идеи из многих других языков, избегая возможностей, которые могут привести к созданию сложного и ненадежного кода. Его способности к параллелизму новы и чрезвычайно эффективны, а подход к абстракции данных и объектно-ориентированному программированию непривычный, но необычайно гибкий. Как и все современные языки, Go обладает эффективным механизмом сбора мусора.

Go особенно хорошо подходит для инфраструктуры: построения инструментария и систем для работы других программистов. Однако, будучи в действительности языком общего назначения, он подходит для любого применения и становится все более популярным в качестве замены нетипизированных языков сценариев, обеспечивая компромисс между выразительностью и безопасностью. Программы Go обычно выполняются быстрее, чем программы, написанные на современных динамических языках, и не завершаются аварийно с неожиданными типами ошибок.

Go — это проект с открытым исходным кодом, так что исходные тексты его библиотек и инструментов, включая компилятор, находятся в открытом доступе. Свой вклад в язык, его библиотеки и инструментарий вносят многие программисты всего мира. Go работает на большом количестве Unix-подобных систем, таких как Linux, FreeBSD, OpenBSD, Mac OS X, а также на Plan 9 и Microsoft Windows; при этом программы, написанные для одной из этих сред, легко переносимы на другие.

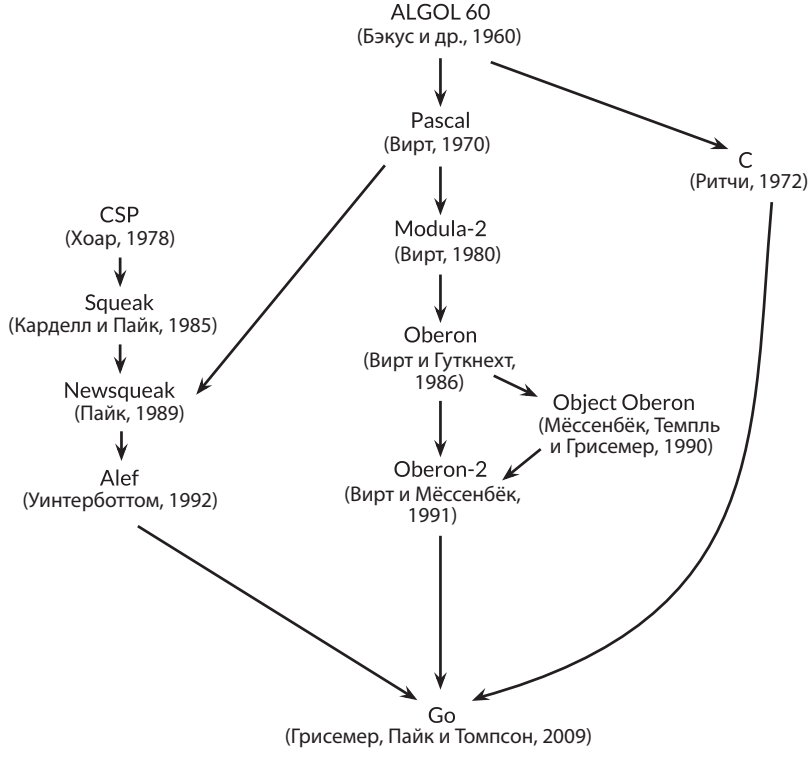
Эта книга призвана помочь вам начать работать с Go, причем с самого начала эффективно использовать все его особенности и богатые стандартные библиотеки для написания понятных, идиоматичных и эффективных программ.

# Происхождение Go

Подобно биологическим видам, успешные языки порождают потомство, которое наследует наилучшие особенности своих предков. Скрещивание при этом иногда приводит к удивительным результатам. Аналогом мутаций служит появление радикально новых идей. Как и в случае с живыми существами, глядя на такое влияние предков, можно многое сказать о том, почему язык получился именно таким, какой он есть, и для каких условий работы он приспособлен более всего.

Если вы просто хотите быстро выучить язык, этот раздел является для вас необязательным, но для глубокого понимания Go имеет смысл ознакомиться с его происхождением.

На рисунке ниже показано, какие языки повлияли на дизайн языка программирования Go.




Go часто описывают как “C-подобный язык” или “язык C XXI века”. От языка C Go унаследовал синтаксис выражений, конструкции управления потоком, базовые типы данных, передачу параметров в функции по значению, понятие указателей и, что важнее всего, направленность C на получение при компиляции эффективного машинного кода и естественное взаимодействие с абстракциями современных операционных систем.

Однако в генеалогическом древе Go есть и другие предки. Одно из сильнейших влияний на Go оказали языки программирования Никлауса Вирта (Niklaus Wirth), начиная с Pascal. Modula-2 привнесла концепцию пакетов; Oberon использует один файл для определения модуля и его реализации; Oberon-2 явился источником синтаксиса пакетов, импорта и объявлений (прежде всего, объявлений методов), которые он, в свою очередь, унаследовал от языка Object Oberon.

Еще одна линия предков Go, которая выделяет его среди прочих современных языков программирования, представляет собой последовательность малоизвестных исследовательских языков, разработанных в Bell Labs и основанных на концепции *взаимодействующих последовательных процессов* (communicating sequential processes — CSP) из статьи Тони Хоара (Tony Hoare) 1978 года, посвященной основам параллелизма.

В CSP программа представляет собой параллельное объединение процессов, не имеющих общего состояния; процессы взаимодействуют и синхронизируются с помощью каналов. Но CSP Хоара представлял собой формальный язык описания фундаментальных концепций параллелизма, а не язык программирования для написания выполнимых программ.

Роб Пайк (Rob Pike) и другие начали экспериментировать с реализациями CSP в виде фактических языков программирования. Первый из них назывался “Squeak” (“язык для общения с мышью”), который являлся языком для обработки событий мыши и клавиатуры со статически созданными каналами. За ним последовал Newsqueak, в котором C-образные инструкции и синтаксис выражений сочетались с записью типов в стиле Pascal. Это был чисто функциональный язык со сборкой мусора, направленный, как и его предшественник, на управление клавиатурой, мышью и оконными событиями. Каналы в нем стали полноправными участниками языка, динамически создаваемыми и хранимыми в переменных.

Операционная система Plan 9 развила эти идеи в языке Alef. Alef попытался сделать Newsqueak жизнеспособным системным языком программирования, но параллелизм без сборки мусора требовал слишком больших усилий.

Ряд конструкций в Go демонстрирует влияние генов непрямых предков; например, `iota` происходит из APL, а лексическая область видимости с вложенными функциями — из Scheme (и большинства последующих за ним языков). Здесь мы находим и признаки мутации: инновационные элементы Go предоставляют динамические массивы с эффективным произвольным доступом, но при этом разрешают сложные размещения, напоминающие связанные списки. Инструкция `defer` также представляет собой новинку Go.

# Проект Go

Все языки программирования тем или иным образом отражают философию их создателей, что часто приводит к включению в язык программирования их реакции на слабости и недостатки более ранних языков. Go не является исключением. Проект

Go родился из разочарования в Google несколькими программными системами, страдающими от “взрыва сложности” (эта проблема отнюдь не уникальна для Google).

Как заметил Роб Пайк (Rob Pike), “сложность мультипликативна”: устранение проблемы путем усложнения одной части системы медленно, но верно добавляет сложность в другие части. Постоянное требование внесения новых функций, настроек и конфигураций очень быстро заставляет отказаться от простоты, несмотря на то что в долгосрочной перспективе простота является ключом к хорошему программному обеспечению.

Простота требует большего количества работы в начале проекта по определению самой сути идеи и большей дисциплины во время жизненного цикла проекта, которая поможет отличать хорошие изменения от плохих. При достаточных усилиях хорошие изменения, в отличие от плохих, могут быть приняты без ущерба для того, что Фред Брукс (Fred Brooks) назвал “концептуальной целостностью” проекта. Плохие же изменения всего лишь разменивают простоту на удобство. Только с помощью простоты дизайна система может в процессе роста оставаться устойчивой, безопасной и последовательной. Проект Go включает в себя сам язык, его инструментарий, стандартные библиотеки и последнее (по списку, но не по значению) — культуру радикальной простоты. Будучи одним из современных высокоуровневых языков, Go обладает преимуществом ретроспективного анализа других языков, и это преимущество использовано в полной мере: в Go имеются сборка мусора, система пакетов, полноценные функции, лексическая область видимости, интерфейс системных вызовов и неизменяемые строки, текст в которых кодируется с использованием кодировки UTF-8. Однако язык программирования Go имеет сравнительно немного возможностей, и вряд ли в него будут добавлены новые. Например, в нем отсутствуют неявные числовые преобразования, нет конструкторов и деструкторов, перегрузки операторов, значений параметров по умолчанию; нет наследования, обобщенных типов, исключений; отсутствуют макросы, аннотации функций и локальная память потока. Язык программирования Go является зрелым и стабильным и гарантирует обратную совместимость: старые программы на Go можно компилировать и запускать с помощью новых версий компиляторов и стандартных библиотек.

Go имеет достаточную систему типов, чтобы избежать большинства ошибок программистов в динамических языках, но эта система типов гораздо более ограниченная, чем в других строго типизированных языках программирования. Это приводит к изолированным очагам “нетипизированного программирования”’ в пределах более широких схем типов. Так что программисты на Go не прибегают к длинным конструкциям, которыми программисты на C++ или Haskell пытаются выразить свойства безопасности своих программ на основе типов. На практике Go дает программистам преимущества безопасности и производительности времени выполнения относительно строгой системы типов без излишней сложности и накладных расходов.

Go поощряет понимание дизайна современных компьютерных систем, в частности — важность локализации. Его встроенные типы данных и большинство библиотечных структур данных созданы для естественной работы без явной инициализации или неявных конструкторов, так что в коде скрывается относительно мало распределений и записей памяти. Составные типы Go (структуры и массивы) хранят свои

элементы непосредственно, требуя меньшего количества памяти и ее распределений, а также меньшего количества косвенных обращений с помощью указателей по сравнению с языками, использующими косвенные поля. А поскольку современные компьютеры являются параллельными вычислительными машинами, Go обладает возможностями параллельности, основанными, как упоминалось ранее, на CSP. Стеки переменного размера легких потоков (или *go-подпрограмм* (goroutines)) Go изначально достаточно малы, чтобы создание одной go-подпрограммы было дешевым, а создание миллиона — практичным.

Стандартная библиотека Go, часто описываемая фразой "все включено", предоставляет строительные блоки и API для ввода-вывода, работы с текстом и графикой, криптографические функции, функции для работы с сетью и для создания распределенных приложений. Библиотека поддерживает множество стандартных форматов файлов и протоколов. Библиотеки и инструменты интенсивно используют соглашения по снижению потребностей в настройке, упрощая тем самым логику программ; таким образом, различные программы Go становятся более похожими одна на другую и тем самым — более простыми в изучении. Проекты создаются с помощью всего лишь одного инструмента `go` и используют только имена файлов и идентификаторов и иногда — специальные комментарии для определения всех библиотек, выполнимых файлов, тестов, примеров, документации и прочего в проектах; исходный текст Go содержит всю необходимую спецификацию построения проекта.

## Структура книги

Мы предполагаем, что читатель программирует на одном или нескольких современных языках программирования, компилирующих языках наподобие C, C++ и Java или динамических, таких как Python, Ruby и JavaScript. Таким образом, мы не стараемся излагать материал так, как будто имеем дело с новичками в программировании. Внешне синтаксис будет вам знаком, так как будет содержать переменные и константы, выражения, управление потоком и функции.

Глава 1 представляет собой руководство по базовым конструкциям Go, содержащее массу небольших программ для решения ежедневных задач наподобие чтения и записи файлов, форматированного вывода результатов, соединений "клиент/сервер" в Интернете и т.п.

В главе 2 описаны структурные элементы программы Go — объявления, переменные, новые типы, пакеты и файлы, области видимости. В главе 3 рассмотрены основные типы данных — числа, логические значения, строки и константы. В главе 4 изучаются составные типы, т.е. типы, построенные из более простых типов с помощью таких механизмов, как массивы, отображения, структуры, а также *срезы* (slices) — нетрадиционное представление динамических списков в Go. Глава 5 посвящена функциям, обработке ошибок, а также инструкциям `panic`, `recover` и `defer`.

Таким образом, главы 1–5 представляют собой реальную основу, то, что является частью любого императивного языка. Синтаксис и стиль Go иногда отличаются от привычных для других языков программирования, но большинство программистов

быстро к этому привыкают. В остальных главах внимание сосредоточено на темах, в которых подход Go менее привычен: методы, интерфейсы, параллелизм, пакеты, тестирование и рефлексия.

Go демонстрирует нестандартный подход к объектно-ориентированному программированию: в нем нет ни иерархий классов, ни каких-либо классов; методы могут быть связаны с любым типом, а не только со структурами; поведение сложных объектов создается из более простых не путем наследования, а с помощью композиции; наконец взаимосвязь между конкретными типами и абстрактными типами (*интерфейсами*) является неявной, так что конкретный тип может удовлетворять интерфейсу так, что его разработчик не будет об этом знать. Методы описаны в главе 6, а интерфейсы — в главе 7.

Go обеспечивает эффективный и удобный механизм параллелизма, используя *go-подпрограммы* (goroutines) и каналы и основываясь на упомянутых выше идеях взаимодействующих последовательных процессов. Широко известные возможности параллелизма Go являются темой главы 8. В главе 9 рассмотрены более традиционные аспекты параллелизма на основе совместно используемых переменных.

В главе 10 описаны пакеты, представляющие собой механизм для организации библиотек. В этой главе также показано, как эффективно использовать инструмент `go`, который обеспечивает компиляцию, тестирование, форматирование программы, документирование и другие задачи, все — единой командой.

Глава 11 посвящена тестированию, где Go использует особенно простой подход, избегая абстрактных схем и предпочитая простые библиотеки и инструменты. Библиотеки тестирования предоставляют основу, на которой при необходимости могут быть построены более сложные абстракции.

В главе 12 рассматривается рефлексия — возможность получения программой информации о собственном представлении во время выполнения. Рефлексия является мощным инструментом, но использовать его следует с осторожностью. В этой главе речь идет о том, как найти правильный компромисс, на примере применения рефлексии при реализации некоторых важных библиотек Go. Глава 13 касается деталей низкоуровневого программирования: как использовать пакет `unsafe` для обхода системы типов Go (и когда это уместно).

В каждой главе имеется ряд упражнений, которые можно использовать для проверки своего понимания Go и для изучения расширений и альтернатив примерам из книги.

Все примеры кода, кроме самых тривиальных, в книге доступны для загрузки из репозитория `git` по адресу `gopl.io`. Каждый пример идентифицируется своим путем импорта пакетов и может быть легко получен с помощью команды `go get`. Вам нужно выбрать каталог, который будет использоваться в качестве рабочего пространства Go, и установить переменную среды `GOPATH` так, чтобы она указывала на этот каталог. Инструмент `go` при необходимости сам создаст этот каталог. Например:

```
$ export GOPATH=$HOME/gobook    # Каталог рабочего пространства
$ go get gopl.io/ch1/helloworld # Выборка, построение, установка
$ $GOPATH/bin/helloworld        # Запуск
Hello, World
```

Для выполнения примеров нужна версия Go не ниже 1.5.

```
$ go version
go version go1.5 linux/amd64
```

Если ваша версия не отвечает указанному требованию, следуйте инструкциям, расположенным по адресу `https://golang.org/doc/install`.

## Дополнительная информация

Основным источником дополнительной информации о Go является официальный веб-сайт `https://golang.org`. Он предоставляет доступ к документации, включая официальную спецификацию языка Go, к стандартным пакетам и т.п. Здесь, кроме того, есть учебные пособия о том, как писать на Go и писать хорошо, а также широкий спектр текстовых и видеоресурсов. Все это будет ценным дополнением к данной книге. В блоге Go по адресу `blog.golang.org` публикуются некоторые из лучших материалов о Go, включая статьи о состоянии языка, планы на будущее, доклады на конференциях и углубленное объяснение широкого спектра тем, связанных с Go.

Одна из самых полезных возможностей онлайн-доступа к Go (и, к сожалению, напрочь отсутствующая в бумажной книге) — возможность запускать программы Go с веб-страниц, на которых эти программы описаны. Эта функциональность обеспечивается сайтом Go Playground по адресу `play.golang.org` и может быть встроена в другие страницы, как, например, начальная страница `golang.org` или страница документации, обслуживаемая инструментом `godoc`.

Go Playground позволяет проводить простые эксперименты с короткими программами для проверки понимания синтаксиса, семантики или библиотечных пакетов. Постоянный URL позволяет обмениваться фрагментами кода Go с другими программистами, сообщать об ошибках или вносить свои предложения. Авторы используют этот сайт по многу раз в день.

Будучи надстройкой над Go Playground, Go Tour по адресу `tour.golang.org` представляет собой последовательность около 75 коротких интерактивных уроков, посвященных основным идеям и конструкциям языка программирования Go и упорядоченных в виде обучающего курса по языку Go.

Основным недостатком Go Playground и Go Tour является то, что они позволяют импортировать только стандартные библиотеки, и к тому же многие библиотечные функции — например, сетевые — ограничены по практическим соображениям и для большей безопасности. Кроме того, для компиляции и выполнения каждой программы они требуют доступа к Интернету. Таким образом, для более сложных экспериментов вам придется запускать программы Go на своем компьютере. К счастью, процесс загрузки прост, и, как отмечалось выше, Go работает в любой современной операционной системе. Так что загрузка Go с сайта `golang.org` и установка на компьютере не должна занять более нескольких минут, после чего вы сможете начать написание и запуск программ Go самостоятельно, не будучи привязанными к конкретному сайту.

Поскольку Go представляет собой проект с открытым кодом, вы можете прочесть код любого типа или функции в стандартной библиотеке, доступной онлайн по адресу `https://golang.org/pkg`; этот же код является частью загружаемого дистрибутива. Используйте это, чтобы понять, как работает та или иная возможность или функция, чтобы получить подробные ответы на свои вопросы или просто узнать, как выглядит хороший код на Go, написанный экспертами.

## Благодарности

Роб Пайк (Rob Pike) и Расс Кокс (Russ Cox), одни из лидеров команды Go, неоднократно тщательно читали рукопись; их комментарии касались всего — от выбора слов до общей структуры и организации книги — и были поистине бесценны. При подготовке японского перевода Ёсики Сибата (Yoshiki Shibata) вышел далеко за рамки служебного долга; его дотошный взгляд обнаружил ошибки в коде и многочисленные несоответствия в английском тексте. Мы высоко ценим тщательные обзоры и критические замечания по данной рукописи, сделанные Брайаном Гётцем (Brian Goetz), Кори Косак (Corey Kosak), Арнольдом Роббинсом (Arnold Robbins), Джошем Бличером Снайдером (Josh Bleecher Snyder) и Питером Вайнбергером (Peter Weinberger).

Мы в долгу перед Самиром Аджмани (Sameer Ajmani), Иттаи Балабаном (Ittai Balaban), Дэвидом Кроушоу (David Crawshaw), Билли Донохью (Billy Donohue), Джонатаном Файнбергом (Jonathan Feinberg), Эндрю Жеррандом (Andrew Gerrand), Робертом Грисемером (Robert Griesemer), Джоном Линдерманом (John Linderman), Минуксом Ма (Minux Ma), Брайаном Миллсом (Bryan Mills), Балой Натаражан (Bala Natarajan), Космосом Николау (Cosmos Nicolaou), Полом Станифортом (Paul Staniforth), Нигелем Тао (Nigel Tao) и Говардом Трики (Howard Trickey) за множество полезных предложений. Мы также благодарим Дэвида Брейлсфорда (David Brailsford) и Рафа Левина (Raph Levien) за советы при верстке.

Наш редактор Грег Донч (Greg Doench) из Addison-Wesley был неизменно полезен — с момента получения этой книги в работу. Издательская команда — Джон Фуллер (John Fuller), Дайна Исли (Dayna Isley), Джули Нагил (Julie Nahil), Чути Прасертсих (Chuti Prasertsith) и Барбара Вуд (Barbara Wood) — оказалась выдающейся; мы не могли и надеяться на лучшее.

Алан Донован (Alan Donovan) благодарит Самира Аджмани (Sameer Ajmani), Криса Деметроу (Chris Demetriou), Уолта Драммонда (Walt Drummond) и Рейда Татджа (Reid Tatge) из Google за то, что они помогли ему найти время для написания книги; Стивена Донована (Stephen Donovan) — за его советы и своевременную поддержку; а больше всех — его жену Лейлу Каземи (Leila Kazemi) за ее энтузиазм и неизменную поддержку этого проекта, несмотря на то что он надолго отвлекал ее мужа от семьи.

Брайан Керниган (Brian Kernighan) глубоко признателен друзьям и коллегам за их терпение и выдержку, а в особенности — своей жене Мэг, которая поддерживала его как при написании книги, так и во всем другом.

Нью-Йорк
Октябрь 2015

# Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: `info@williamspublishing.com`
WWW: `http://www.williamspublishing.com`

Наши почтовые адреса:

в России: 127055, Москва, ул. Лесная, д. 43, стр. 1
в Украине: 03150, Киев, а/я 152