Методичка

Учебник по программированию на Go

Введение. Введение в язык Go

Go - это язык программирования, разработанный в 2007 году в Google. Он предназначен для написания эффективных, масштабируемых и надежных программ.

Go был создан как ответ на необходимость в языке, который удовлетворял бы потребности современного программирования, такие как параллелизм и распределенные системы. Он был разработан с учетом производительности, простоты и удобства использования.

Go имеет синтаксис, похожий на язык программирования C, но с более высоким уровнем абстракции. Он также включает в себя некоторые конструкции, характерные для функционального программирования.

Основные преимущества языка Go:

- Высокая скорость выполнения программ
- Удобство использования и простота синтаксиса
- Встроенная поддержка параллелизма и распределенных систем
- Большое сообщество разработчиков и множество библиотек
- Высокая надежность и безопасность

В данном учебнике мы рассмотрим основы программирования на языке Go и научимся создавать эффективные и надежные программы.

Одной из главных причин выбора языка Go является его высокая эффективность и быстродействие, что делает его идеальным выбором для написания масштабируемых и надежных программ с высокой производительностью. Благодаря встроенной поддержке параллелизма, Go также является идеальным языком для написания программ, которые должны работать в распределенных и параллельных средах, таких как облачные вычисления или микросервисные архитектуры.

Кроме того, Go имеет простой и понятный синтаксис, который позволяет разработчикам быстро и легко создавать и поддерживать код. Большая часть

стандартной библиотеки Go написана на самом языке, что делает ее легкой для понимания и использования.

Go также обладает высокой надежностью и безопасностью благодаря строгой типизации и проверке ошибок во время компиляции. Это снижает количество ошибок в коде и повышает надежность программ.

Наконец, Go имеет большое сообщество разработчиков и множество библиотек, которые делают его еще более привлекательным для использования в различных проектах.

Многие крупные компании используют Go для разработки своих продуктов и сервисов. Например, Twitch использует Go для разработки своей платформы онлайн-трансляций, а SoundCloud использует его для обработки аудиофайлов. OZON и WildBerries используют Go для разработки своих интернет-магазинов, а СберБанк использует его для разработки своих банковских приложений.

Использование Go позволяет этим компаниям создавать быстрые, надежные и безопасные продукты, которые способны обрабатывать большие объемы данных и обеспечивать высокую производительность. Кроме того, использование Go позволяет компаниям сокращать время разработки и снижать затраты на техническую поддержку, благодаря простоте языка и легкости поддержки кода.

Таким образом, использование Go позволяет компаниям быть более конкурентоспособными на рынке, предоставляя им возможность быстро и эффективно разрабатывать новые продукты и сервисы, которые отвечают требованиям современных пользователей.

Go - это популярный язык программирования, разработанный Google, который широко используется в индустрии благодаря своей простоте и производительности. В этой главе мы рассмотрим, как установить Go на ваш компьютер с операционными системами Linux, macOS и Windows.

Установка Go

Установка на Linux

▼ Рекомендуемая установка

- Зайдите в терминал
- Вбейте указанную команду

```
sudo add-apt-repository ppa:longsleep/golang-backports
```

- Вбейте указанную команду

```
sudo apt update
```

- Вбейте указанную команду

```
sudo apt install golang-go
```

▼ Стандартная

- Зайдите на страницу загрузки Go по адресу
- Найдите архив с бинарными файлами для вашей версии Linux (обычно это файл .tar.gz) и скачайте его
- Откройте терминал и перейдите в каталог, в котором был скачан архив, используя указанную команду

```
cd <Сюда впишите путь до вашей директории>
```

- Извлеките архив, выполнив следующую команду, где [версия] - номер скаченной версии Go

```
tar -C /usr/local -xzf go[версия].linux-amd64.tar.gz
```

- Добавьте путь к исполняемым файлам Go в переменную окружения `PATH`, выполнив следующую команду

```
echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.profile
```

- Загрузите новые настройки в текущий терминал, выполнив команду

```
source ~/.profile
```

- Проверьте установку, выполнив указанную команду, в ответ вы должны увидеть номер установленной версии Go

```
go version
```

Установка на macOS

▼ Если есть brew

- Зайдите в терминал
- Вбейте указанную команду

```
brew update
```

- Вбейте указанную команду

```
brew install go
```

- Проверьте установку, выполнив указанную команду, в ответ вы должны увидеть номер установленной версии Go.

```
go version
```

▼ Если нет brew

- Зайдите в терминал
- Вбейте указанную команду

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
```

- Вбейте указанную команду

```
brew update
```

- Вбейте указанную команду

```
brew install go
```

- Проверьте установку, выполнив указанную команду, в ответ вы должны увидеть номер установленной версии Go.

```
go version
```

Установка на Windows

▼ Стандартная

Зайдите на страницу загрузки Go по адресу.

Найдите установщик для Windows (обычно это файл .msi) и скачайте его.

Запустите скачанный файл и следуйте инструкциям установщика.

Теперь у вас установлен Go на вашем компьютере, и вы готовы начать разрабатывать программы. В следующих главах мы изучим основы языка Go и научимся создавать простые программы.

Установка IDE

Установка на Linux

▼ Рекомендуемая установка

- Зайдите в терминал
- Вбейте указанную команду

```
sudo snap install --classic code
```

Установка на macOS

▼ Рекомендуемая установка

- Зайдите на сайт https://code.visualstudio.com/download
- Скачайте версию для MacOS

Установка на Windows

▼ Рекомендуемая установка

- Зайдите на сайт https://code.visualstudio.com/download
- Скачайте версию для Windows

После установки, нужно зайти в раздел плагинов VSCode и скачать плагин для работы с Go.

Глава 1. Основы языка Go

Раздел 1. "Hello, World!" на Go

Начнем с написания простой программы на Go, которая выводит на экран строку "Hello, World!". Вот как это делается:

```
package main // объявление пакета

import "fmt" // импорт библиотеки

func main() { // объявление функции main
    fmt.Println("Hello, World!") // вывод строки на экран
}
```

Теперь давайте разберем каждую строку кода по отдельности:

- `package main` - это объявление пакета. В Go каждый файл должен находиться в каком-то пакете. Пакет `main` используется для запуска программы и должен содержать функцию `main()`.
- `import "fmt"` - это импорт библиотеки `fmt`. Библиотека `fmt` используется для вывода данных в терминал.
- `func main()` - это объявление функции `main()`. Функция `main()` - это точка входа в программу. Вся программа начинается с вызова функции `main()`.

- `fmt.Println("Hello, World!")` - это строка кода, которая выводит на экран строку "Hello, World!". Функция `Println` из библиотеки `fmt` используется для вывода строки на экран и добавления перевода строки в конце.
- В этом примере `func` - это ключевое слово, которое используется для объявления функции. `main` - это имя функции. После имени функции идут круглые скобки `()`. Они могут содержать параметры, но в данном примере они пустые, поскольку функция не принимает никаких параметров. Тело функции начинается с открывающей фигурной скобки `{` и заканчивается закрывающей фигурной скобкой `}`. Все, что находится внутри этих скобок, является телом функции. В данном случае тело функции содержит только одну инструкцию: `fmt.Println("Hello, World!")`, которая выводит строку "Hello, World!" в консоль при вызове функции.

Теперь, когда мы разобрали каждую строку, давайте посмотрим, как мы можем написать этот блок кода:

1. Откройте новый файл с расширением `.go` в вашем текстовом редакторе.
2. Введите следующие строчки кода и в конце сохраните файл:

```
package main

import "fmt"

func main() {
        fmt.Println("Hello, World!")
}
```

Теперь, когда мы сохранили файл, мы можем запустить нашу программу.

Для этого нам надо будет воспользоваться терминалом!

Терминал (или командная строка) - это текстовый интерфейс, через который пользователи могут взаимодействовать с операционной системой компьютера. Это приложение, которое принимает команды, введенные с клавиатуры, и передает их операционной системе для выполнения.

В разных операционных системах терминал может называться по-разному. Например, в Windows это "Командная строка", в macOS и Linux - "Терминал".

Терминал предоставляет пользователям возможность выполнять различные задачи, такие как управление файлами и каталогами, запуск и остановка служб, настройка системных параметров и многое другое. Некоторые задачи могут быть выполнены быстрее или эффективнее с помощью командной строки, чем через графический интерфейс.

В то время как большинство пользователей предпочитают использовать графический интерфейс для взаимодействия с их компьютером, терминал остается важным инструментом для системных администраторов, разработчиков и других продвинутых пользователей.

Для того, чтобы запустить программу вам нужно открыть терминал и перейти в каталог, в котором находится ваш файл. Затем выполните следующую команду:

```
go run имя_файла.go
```

Где `имя_файла.go` - это имя вашего файла с расширением `.go`.

Если вы все сделали правильно, то на экране появится следующая строка:

```
Hello, World!
```

Теперь вы знаете, как написать простую программу на Go и как ее запустить. В следующих разделах мы изучим основные элементы языка Go и научимся создавать более сложные программы.

Шпаргалка

- Каждый файл должен находиться в определенном пакете.
- Для запуска программы должен использоваться пакет `main` , в котором обязательно должна быть функция точки входа в программу `main().`
- В языке программирования Go за вывод строчки на экран отвечает функция `Println` из пакета `fmt.` Если нужно вывести какое-то строковое значение, его нужно передать в саму функцию. Это делает с помощью

двойных кавычек. Так компилятор будет понимать, что ему на вход передали строковое значение. Пример - `fmt.Println("Hello, World!")`.

Раздел 2. Переменные в Go

Переменные - это именованные области памяти, которые используются для хранения данных в программе. В Go, перед использованием переменной, ее нужно объявить. Объявление переменной состоит из имени переменной и типа данных, который она будет хранить.

Типы данных в Go

Тип данных - это способ определения, какой тип значений может хранить переменная. Например, переменная типа `int` может хранить только целые числа.

В Go есть несколько встроенных типов данных, таких как:

- `bool` - логический тип, который может принимать значения `true` или `false`.
- `int` - целочисленный тип, который может хранить целые числа.
- `float64` - тип с плавающей точкой, который может хранить дробные числа.
- `string` - тип для хранения строковых значений.

Типы данных имеют свои ограничения. Например, переменная типа `int` не может хранить дробные числа. Если вам нужно хранить дробные числа, вы можете использовать тип `float64`.

В Go есть и другие типы данных, такие как `byte`, `rune`, `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, `uint64`, `uintptr`, `complex64`, `complex128` и `error`. Однако, мы обсудим их подробнее позже.

Пример с коробкой для фруктов

Представьте, что у вас есть коробка для фруктов. Она может содержать только определенные типы фруктов, такие как яблоки, апельсины и бананы. Каждый фрукт имеет свои уникальные характеристики, такие как цвет, размер и форма.

Точно так же, в Go переменные могут хранить только определенные типы значений. Например, переменная типа `int` может хранить только целые

числа, а переменная типа `string` может хранить только строки.

Типы данных в Go позволяют программистам более точно определять, какие типы значений могут храниться в переменных и как они могут быть использованы в программе. Это помогает избежать ошибок при работе с данными и делает программу более надежной и стабильной.

Объявление переменных

Вот пример объявления переменной типа `int`:

```
var x int
```

В этом примере мы объявляем переменную `x` типа `int`. Перед использованием переменной `x`, нам нужно присвоить ей значение, которое соответствует типу данных.

Вот пример присвоения значения переменной `x`:

```
x = 10
```

Мы присваиваем переменной `x` значение `10`, которое является целым числом.

Мы также можем объявлять и присваивать значение переменной в одной строке:

```
var x int = 10
```

В этом примере мы объявляем переменную `x` типа `int` и назначаем ей значение `10`.

Использование переменных

После объявления переменной мы можем использовать ее в программе. Например, мы можем вывести значение переменной `x` на экран:

```
fmt.Println(x)
```

В этом примере мы используем функцию `Println` из библиотеки `fmt`, чтобы вывести значение переменной `x` на экран.

Примеры использования переменных

Вот несколько примеров использования переменных в Go:

```
// Объявление переменной типа string и присваивание ей значен
ия
var name string = "John"

// Объявление переменной типа int и присваивание ей значения
var age int = 30

// Объявление переменной типа float64 и присваивание ей значе
ния
var height float64 = 1.75

// Объявление переменной типа bool и присваивание ей значения
var isMarried bool = true

// Использование переменных в программе
fmt.Println("Name:", name)
fmt.Println("Age:", age)
fmt.Println("Height:", height)
fmt.Println("Married:", isMarried)
```

В этом примере мы объявляем переменные `name`, `age`, `height` и `isMarried` и присваиваем им значения. Затем мы используем эти переменные в функции `Println` из библиотеки `fmt`, чтобы вывести значения переменных на экран.

Переменные - это именованные области памяти, которые используются для хранения данных в программе. Их использование позволяет сохранять и манипулировать данными в течение жизненного цикла программы.

Переменные могут быть использованы в различных контекстах, например, для хранения пользовательских данных, временных значений или констант.

Они также могут быть использованы для передачи значений между различными функциями в программе.

Переменные должны быть объявлены перед использованием, указывая их тип, имя и, возможно, начальное значение. В Go, переменные могут быть объявлены с помощью ключевого слова `var` . Если значение вы не укажите, то в этом случае тип получит “стандартное значение”, так называемое “дефолтное”, в зависимости от типа он отличается. К примеру, для числовых типов дефолтное значение будет 0.

Кроме того, в Go есть возможность использовать короткие объявления переменных, которые позволяют объявить и присвоить значение переменной в одной строке, нужно помнить, что в этом случае тип проставляется автоматически:

```
x := 10
```

В этом примере мы используем короткое объявление переменной для создания переменной `x` типа `int` и присваивания ей значения `10`.

Использование переменных в программе позволяет сохранять и обрабатывать данные в более удобной форме, делая программу более читаемой и поддерживаемой.

Давайте также подробно обсудим тип данных `bool` . В Go используется для хранения логических значений и может иметь только два возможных значения: `true` и `false`. Этот тип данных обычно используется для выполнения логических операций в программах, таких как проверка условий и принятие решений на основе этих условий.

Например, допустим, у нас есть программа, которая проверяет, равно ли число нулю. Мы можем использовать тип данных `bool`, чтобы хранить результат проверки:

```
var isZero bool = 4 == 0
```

В этом примере мы объявляем переменную `isZero` типа `bool` и присваиваем ей значение `false`, потому что результат `4` не равен `0`.

Тип данных `bool` также может использоваться для выполнения логических операций, таких как `&&` (логическое "И"), `||` (логическое "ИЛИ") и `!` (логическое отрицание). Например:

```
var x bool = true
var y bool = false

// Логическое "И"
var z1 bool = x && y // false

// Логическое "ИЛИ"
var z2 bool = x || y // true

// Логическое отрицание
var z3 bool = !x // false
```

В этом примере мы объявляем переменные `x` и `y` типа `bool` и выполняем логические операции с помощью операторов `&&`, `||` и `!`. Результаты операций сохраняются в переменных `z1`, `z2` и `z3`.

В целом, тип данных `bool` используется в программах для выполнения логических операций и принятия решений на основе этих операций. Это позволяет программистам создавать более умные и гибкие программы, которые могут адаптироваться к изменяющимся условиям и требованиям.

Пример использования `bool` в программе:

```
var isSunny bool = true

if isSunny {
    fmt.Println("It's a sunny day!")
} else {
    fmt.Println("It's a cloudy day.")
}
```

В этом примере мы объявляем переменную `isSunny` типа `bool` и присваиваем ей значение `true`. Затем мы используем условный оператор `if` для проверки

значения переменной `isSunny`. Если значение равно `true`, то мы выводим сообщение "It's a sunny day!". Если значение равно `false`, то мы выводим сообщение "It's a cloudy day.".

Тип данных `bool` играет важную роль в создании более интеллектуальных программ и позволяет программистам более точно определять логические операции и условия для выполнения различных действий в программе.

Конвертация типов данных

Конвертация типов данных или приведение типов данных - это процесс изменения типа значения переменной. Это может быть полезно, когда мы хотим использовать значение переменной в определенном контексте или выполнить определенные операции. Например, нам нужно вычислить сумму двух переменных, но типы этих переменных несовместимы между собой. Если мы напрямую попытаемся сложить переменную с типом int с переменной типа float64 мы получим ошибку компиляции.

`invalid operation: x + y (mismatched types float64 and int)`

Компилятор Go не смог сложить переменные из-за несоответствия типов.

В Go можно выполнять конвертацию типов данных с помощью оператора приведения типа `T(v)`, где `T` - тип, к которому нужно выполнить приведение, а `v` - значение, которое нужно преобразовать.

Например, если у нас есть переменная `x` типа `float64` со значением `10.5`, мы можем выполнить приведение этого значения к типу `int` следующим образом:

```
var x float64 = 10.5
var y int = int(x)
```

В этом примере мы объявляем переменную `x` типа `float64` со значением `10.5`. Затем мы объявляем переменную `y` типа `int` и выполняем приведение значения `x` к типу `int` с помощью оператора приведения типа `int(x)`.

Если мы попытаемся выполнить приведение значения, которое не может быть конвертировано в тип, то мы получим ошибку компиляции. Например,

если мы попытаемся выполнить приведение строки к типу `int`, то мы получим ошибку компиляции:

```
var s string = "Hello, world!"
var i int = int(s) // Ошибка компиляции: cannot convert "Hell
o, world!" to int
```

Кроме того, в Go есть возможность при приведении типов данных использовать функции, которые возвращают два значения. В этом случае первое значение - это преобразованное значение, а второе значение - это флаг успешности операции.

Например, если мы попытаемся выполнить приведение строки к типу `int` с помощью функции `strconv.Atoi`, мы можем получить преобразованное значение и флаг успешности:

```
import "strconv"

var s string = "10"
var i, err = strconv.Atoi(s)

if err != nil {
    // Обработка ошибки
} else {
    fmt.Println(i)
}
```

В этом примере мы импортируем пакет `strconv`, который содержит функцию `Atoi`, позволяющую выполнить приведение строки к типу `int`. Затем мы объявляем переменную `s` типа `string` со значением `10`. Мы используем функцию `strconv.Atoi` для преобразования строки `s` в значение типа `int`. Функция `Atoi` возвращает два значения: преобразованное значение и флаг успешности операции. Мы сохраняем эти значения в переменные `i` и `err`. Затем мы проверяем флаг успешности и выводим преобразованное значение на экран, если операция была успешной. Подробнее про if и обработку ошибок мы поговорим позже.

В целом, конвертация типов данных является важной частью программирования на Go и позволяет программистам работать с данными различных типов в более гибкой и эффективной форме.

Математические операторы

В Go есть несколько операторов, которые можно использовать для увеличения или уменьшения значения переменной на определенное число или для выполнения операции получения остатка от деления.

Оператор `++` используется для увеличения значения переменной на 1:

```
var x int = 5
x++
fmt.Println(x) // Вывод: 6
```

В этом примере мы объявляем переменную `x` типа `int` со значением `5`. Затем мы используем оператор `++` для увеличения значения переменной `x` на 1. В результате переменная `x` теперь равна `6`.

Аналогичным образом, оператор `--` используется для уменьшения значения переменной на 1:

```
var x int = 5
x--
fmt.Println(x) // Вывод: 4
```

В этом примере мы используем оператор `--` для уменьшения значения переменной `x` на 1. В результате переменная `x` теперь равна `4`.

Операторы `++` и `--` могут быть использованы как в префиксной, так и в постфиксной форме:

```
var x int = 5

// Префиксная форма
x = ++x // x равно 6
```

```
// Постфиксная форма
x = x++ // x равно 6
```

Обратите внимание, что результаты операций могут отличаться в зависимости от того, используется ли оператор в префиксной или постфиксной форме.

Оператор `%` используется для получения остатка от деления двух чисел:

```
var x int = 10
var y int = 3
var z int = x % y
fmt.Println(z) // Вывод: 1
```

В этом примере мы объявляем переменные `x` и `y` типа `int` со значениями `10` и `3` соответственно. Затем мы используем оператор `%` для получения остатка от деления `x` на `y`. В результате переменная `z` теперь равна `1`.

В целом, операторы `++`, `--` и `%` являются важной частью программирования на Go и позволяют программистам работать с переменными различных типов и выполнить различные математические операции.

Шпаргалка

1. Переменные - именованные области памяти, созданные для управления данными в течении жизненного цикла программы.
2. Встроенные типы данных в Go - `bool, int, float64, string.`
3. Для работы с переменными нужно выполнить несколько этапов - объявить и инициализировать. Объявление переменной - процесс, когда мы указываем тип переменной. Инициализация переменной - присвоение ей значения. Для простоты - оба этих шага могут быть скомбинированы в один, когда мы в одной строчке кода объявляем переменную и сразу же присваиваем ей значение.
   Например,
   var x int = 10
4. Конкатенация - операция склеивания строк. Выполняется с помощью знака “+”.

Так при конкатенации “микро” + “космос” мы получим “микрокосмос”.

Домашнее задание

1. **Объявление переменных**: Создайте переменные различных типов (int, float64, string, bool) и присвойте им значения. Выведите на экран значения этих переменных.
2. **Математические операции**: Создайте две переменные типа int и присвойте им значения. Напишите программу, которая выводит результат сложения, вычитания, умножения и деления этих переменных.
3. **Использование bool**: Объявите две переменные типа bool и присвойте им значения. Напишите программу, которая выводит результат логической операции "И" (&&) и "ИЛИ" (||) этих переменных.
4. **Конвертация типов**: Создайте переменную типа string, содержащую число. Преобразуйте эту строку в int и выполните некоторые математические операции.
5. **Объявление переменных**: Исследуйте разницу между объявлением переменных с помощью var и с помощью короткого объявления :=. Создайте примеры для обоих случаев.
6. **Математические операции с float**: Объявите две переменные типа float64, присвойте им значения и выполните различные математические операции. Обратите внимание на точность результатов.
7. **Работа с bool (продвинутый уровень)**: Создайте переменные типа bool и присвойте им разные значения. Напишите программу, которая выводит результат логической операции "отрицание" (!) для каждой из переменных.
8. **Конвертация типов с float**: Используя преобразование типов, преобразуйте float64 в int и наоборот. Обратите внимание на то, что происходит с десятичной частью числа при преобразовании.
9. **Задача на понимание переменных**: Объявите переменную, присвойте ей значение, затем присвойте этой же переменной новое значение.

Выведите значение переменной до и после изменения. Объясните полученный результат.

10. **Комплексное задание (⭐ сложное)**: Создайте программу, которая принимает на вход два значения разных типов (например, string и int), преобразует одно из значений в другой тип и производит математическую операцию. Результат должен выводиться на экран. Для выполнения этого задания вам потребуется самостоятельно изучить функцию Scan.
11. **Получение bool значения из операции сравнения:** Создайте две переменных типа int, задайте им произвольное значение, используйте операцию сравнения при создании переменной типа bool, выведите значение переменной bool на экран.
    Конвертируйте один из типов int в float64 и попробуйте поменять значение переменной типа bool используя операцию сравнения float64 и int. Объясните полученный результат.
12. **Конкатенация:** Создайте две переменные типа string, одну со значением “Hello”, другую со значением “World!”. Выведите результат слияния двух строк с помощью оператора “+”.
13. **Задача с настенными часами**: Создайте переменную d типа int со значением 137, это количество градусов на которое передвинулась часовая стрелка с начала суток, вычислите количество часов и минут на которых она находится. Выведите на экран результат в формате “*h* hours, *m* minutes”, где *h* и *m* часы и минуты соответственно.
    Попробуйте подставить разные значения градусов в пределах 0<d<360.