package main

папка внутри которой мы работает, все будующие файлы которые мы будем создавать будут находиться в этом пространстве имён

Problem:

PS D:\Education\Go\www> go build

go: cannot find main module, but found .git/config in D:\Education\Go

to create a module there, run:

cd .. && go mod init

Fix:

When you encounter the error message “go: cannot find main module, but found .git/config,” it usually indicates that Go is unable to locate the main module for your project. Here are some steps to resolve this issue:

1. **Initialize a Go Module**:
   * Navigate to your project directory using the terminal.
   * Run the following command to create a go.mod file in the root of your project:
   * go mod init github.com/example/go-app-name
   * Replace github.com/example/go-app-name with the appropriate module name for your project.

# PS D:\Education\Go\Book> go mod init main

1. **Update Your Go Module** (Optional):
   * If you’re inside a Git repository, it’s generally advisable to initialize the module within that repository. This allows Go modules to use information from your remote repository to determine the correct module name.
   * You can do this by running:
   * git init
   * git remote add origin <repository-url>
   * go mod init
   * Alternatively, you can specify the module name directly:
   * go mod init <module-name>

Replace <module-name> with the desired name for your module.

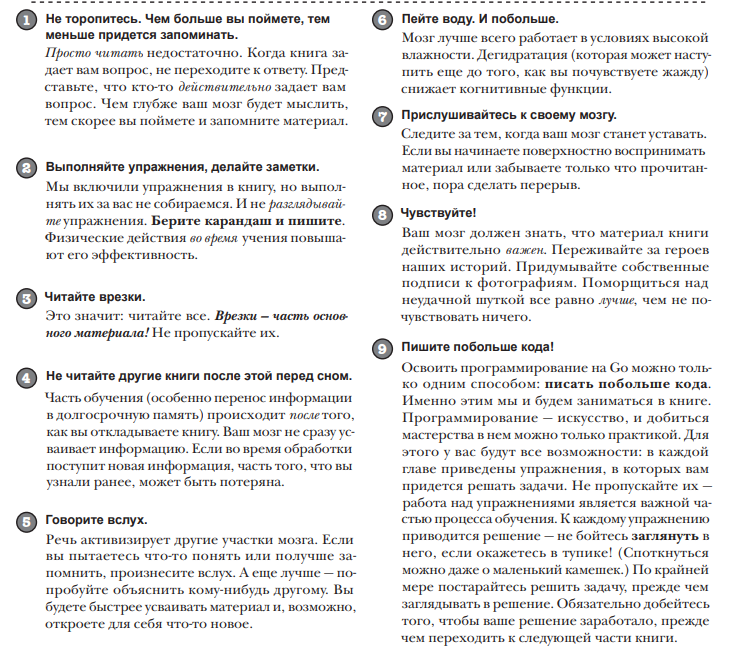
1. **Tidy Up Dependencies**:
   * After initializing the module, run the following command to tidy up your dependencies:
   * go mod tidy
2. **Build Your Program**:
   * Finally, build your program using:
   * go build your-program.go

Remember that using Go modules is generally recommended, especially as it simplifies dependency management and ensures consistency across projects. If you’re just starting out, consider naming your modules following the same structure you would use for a repository (e.g., github.com/example/go-app-name). This convention allows commands like go get to function correctly without manual intervention.

Hope this helps! 🚀

# #3 – Создание структур (модели данных)

Вместо классов в go используются структуры, и когда мы создаём некоторую структуру мы по сути создаём некий класс, на основе которого мы можем создавать некоторые объекты, и если у нас будет несколько структур то можно реализовывать различные концепции ООП, такие как наследование и инкапсуляцию



Когда вы распространяете свой код, другие разработчики ожидают, что он будет оформлен в стандартном формате Go. Это означает стандартное форматирование отступов, пробелов и т. д., чтобы другим людям было проще читать ваш код. Если в других языках программирования разработчикам приходилось вручную переформатировать свой код, чтобы он соответствовал стилевому руководству, в Go достаточно выполнить команду go fmt, которая автоматически сделает все за вас. (39 Стр)

Если строки обычно используются для представления последовательностей символов, то руны в языке Go представляют отдельные символы. Строковые литералы заключаются в двойные кавычки ("), а рунные литералы записываются в одиночных кавычках ('). (Cтр 46)

Если имя переменной, функции или типа начинается с буквы верхнего регистра, оно считается экспортируемым и может использоваться в других пакетах, кроме текущего. (Именно поэтому буква P в fmt.Println имеет верхний регистр: это нужно для того, чтобы его можно было использовать в main или любом другом пакете.) Если имя переменной/функции/типа начинается с буквы нижнего регистра, оно считается неэкспортируемым. Такие имена доступны только в текущем пакете.

Стр 62

Пакет представляет собой группу взаимосвязанных

функций и других блоков кода.

� Прежде чем использовать функции из пакета в файле

Go, необходимо импортировать этот пакет.

� Строка — последовательность байтов, обычно представляющих символы текста.

� Руна представляет отдельный символ текста.

� Два самых распространенных числовых типа Go — int

(для хранения целых чисел) и float64 (для хранения

чисел с плавающей точкой).

� Тип bool используется для хранения логических значений (true или false).

� Переменная представляет собой блок памяти для

хранения значения заданного типа.

� Если переменной не присвоено значение, то она содержит нулевое значение для своего типа. Примеры нулевых значений: 0 для переменных int или float64,

"" для строковых переменных.

� Объявление переменной можно совместить с присваиванием ей значения при помощи короткого объявления

переменной :=.

� К переменной, функции или типу можно обращаться из

кода других пакетов только в том случае, если ее имя

начинается с буквы верхнего регистра.

� Команда go fmt автоматически переформатирует

исходные файлы по стандартам Go. Всегда выполняйте команду go fmt для любого кода, который

вы собираетесь передавать другим разработчикам.

� Команда go build компилирует исходный код

Go в двоичный формат, который может выполняться

компьютером.

� Команда go run компилирует и выполняет программу

без сохранения в исполняемом файле в текущем каталоге

go fmt hello.go – форматирование кода

go build hello.go - компиляция исходного кода

x := 4 – объявление переменной int

y = float64(x) – преобразование типа значения

fmt.Println(reflect.TypeOf(y)) - вывод типа значения

Путь импортирования — всего лишь уникальная строка, которая идентифицирует пакет и используется в директиве import. После того как пакет будет импортирован, к нему можно обращаться по имени пакета. (90ст)

Циклы 96 page

В языке Go также поддерживаются операторы присваивания += и -=

Page 102

В Go предусмотрены два ключевых слова для управления циклом.

Первое — **continue** — осуществляет немедленный переход к следующей итерации цикла; при этом дальнейший код текущей итерации в блоке цикла пропускается.

Второе ключевое слово **break** приводит к немедленному выходу из цикла. Дальнейший код в блоке цикла не отрабатывается, другие итерации цикла не выполняются. Управление передается первой команде, следующей за циклом.

Page 108

* Метод — разновидность функций, связываемых со значениями конкретного типа.
*  Go интерпретирует все символы от // до конца строки
* как комментарий и игнорирует их.
*  Многострочные комментарии начинаются с /\* и завершаются \*/. Все символы между этими маркерами,
* включая символы новой строки, игнорируются.
*  Традиционно в начало любой программы включается
* комментарий, который объясняет, что делает программа.
*  В отличие от большинства языков программирования,
* Go допускает множественные возвращаемые значения
* из вызова функции или метода.
*  Одно из стандартных применений множественных
* возвращаемых значений — возвращение основного
* результата функции и второго значения, которое сообщает, произошла ли ошибка при вызове.
*  Чтобы проигнорировать значение без реального использования в программе, воспользуйтесь пустым
* идентификатором \_. Пустой идентификатор может
* использоваться вместо любой переменной в любой
* команде присваивания.
*  Постарайтесь не присваивать переменным имена, совпадающие с именами типов, функций или пакетов; это
* приведет к замещению (переопределению) элемента
* с тем же именем.
*  Функции, условные команды и циклы содержат блоки
* кода, заключенные в фигурные скобки {}.
*  Файлы и пакеты также образуют блоки, хотя содержащийся в них код и не заключается в фигурные скобки {}.
*  Область видимости переменной ограничивается
* блоком, в котором она определяется, а также всеми
* блоками, вложенными в этот блок.
*  Кроме имени, пакет может иметь путь импортирования, который должен указываться при импортировании.
*  Ключевое слово continue осуществляет переход
* к следующей итерации цикла.

 Ключевое слово break полностью прерывает выполнение цикла.

Форматирование вывода функциями Printf и Sprintf Page 115

Функция получает строку и вставляет в нее одно или несколько значений, отформатированных заданным способом. После этого функция выводит полученную строку.

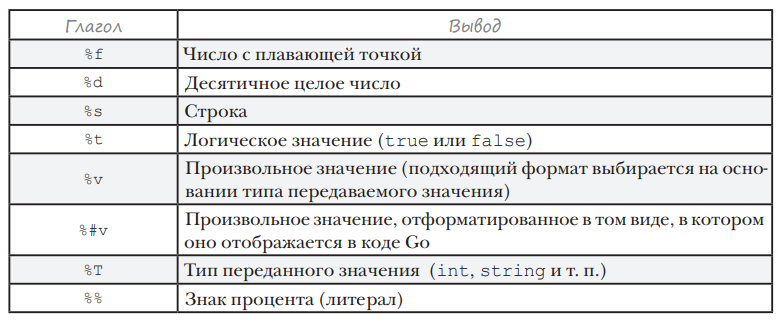
fmt.Printf("About one-third: %0.2f\n", 1.0/3.0)

About one-third: 0.33

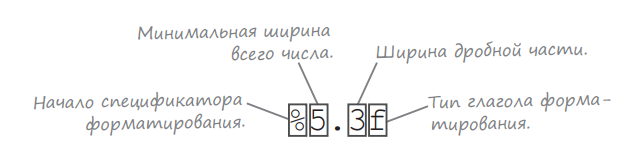
Функция Sprintf (также из пакета fmt) в целом похожа на Printf, но она возвращает отформатированную строку, а не выводит ее.

resultString := fmt.Sprintf("About one-third: %0.2f\n", 1.0/3.0) fmt.Printf(resultString)

About one-third: 0.33



Page 118



Page 471

Параметры и возвращаемое значение функции являются частью ее типа. Для переменной, в которой хранится функция, должны быть указаны параметры и возвращаемые значения этой функции. В переменной могут храниться только функции, у которых количество и тип параметров соответствуют заданному типу.

