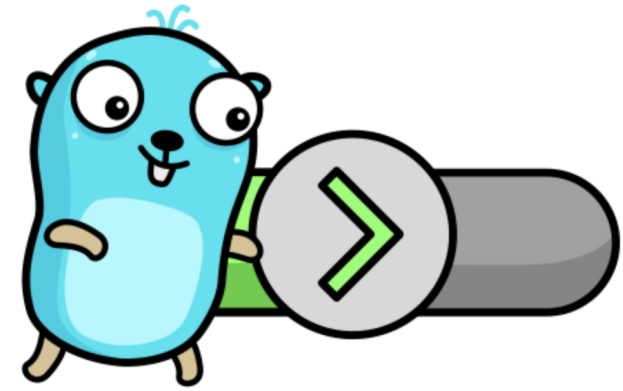
Вступительное слово

Категорически приветствуем!

В этом уроке мы поговорим о том, какие инструменты понадобятся для прохождения курса, где общаться и куда жаловаться :)

Вперёд!



Чат курса

Общаемся в комментариях к урокам и в Telegram: https://t.me/goinpractice.

Можно задавать любые вопросы (а лучше давать ответы) по материалу, заданиям и по языку в целом.
Не забываем пользоваться правилом nometa.

Welcome!

Репозиторий курса

Примеры из теоретической части, а также заготовки для домашек лежат в

https://github.com/www-golang-courses-ru/advanced-dealing-with-errors-in-go

Если вы незнакомы с git, то самое время познакомиться. В целом будет достаточно пары команд:

```
$ git clone
https://github.com/www-golang-courses-ru/advanced-dealing-with-errors
-in-go.git
Cloning into 'advanced-dealing-with-errors-in-go'...


$ cd advanced-dealing-with-errors-in-go
```

Форкать и загружать свои решения в GitHub особого смысла **не имеет**, так как поделиться решением, а также ознакомиться с решениями коллег и авторов можно будет с помощью платформы Stepik.

Как сдавать задачи?

Практически все задачи не имеют примеров ввода и вывода:

```
Sample Input:

Sample Output:

SUCCESS
```

Всё потому, что они тестируются не через STDIN-STDOUT, а через полноценные гошные тесты, эквивалентные тем, что вы видите в заготовке к задаче (пример).

Поэтому правило одно – сделайте так, чтобы ваша реализация проходила тесты в заготовке и скорее всего она будет принята и Stepik'ом.

Если в какой-то задаче вам не хватает определённых импортов – пишите в комментариях к задаче или сразу в чат – мы добавим их!

Модули построены так, что вам не будет доступен следующий модуль, пока вы не сдадите все задачи в текущем. Мы считаем практику неотъемлемой частью любого обучения и забивать на неё болт не выйдет. Если вам непонятно условие или вы имеете замечания по задаче, то не стесняйтесь писать нам или в комментариях на странице к задаче. Спасибо!

Компиляторы

Go >= 1.14.4

На момент завершения работы над курсом последней версией Go является **1.16.4**, но в целом для решения задач подойдёт любая версия не ниже **Go 1.14.4**.

Установить актуальную версию компилятора можно с помощью страницы https://golang.org/doc/install.

Также курс изредка касается версий 1.12, 1.13 и 1.14. Если вам хочется самим воспроизвести какие-то примеры на конкретной версии языка, то её можно доставить себе в систему, следуя данной инструкции.

Stepik будет запускать ваши программы с помощью команды

```
# go 1.14.4

$ go build -ldflags "-extldflags -static"
```

---

gcc >= 6.3.0

Для выполнения парочки первых домашек на Си вам понадобится gcc не ниже **6.3.0** с поддержкой диалекта C99.

Stepik будет запускать ваши программы с помощью команды

```
# gcc 6.3.0
$ gcc -std=c99 -pipe -O2 -static -o main
```

На MacOS `gcc` прилетает с **XCode Command Line Tools** и *неявно* ссылается на `Apple clang`. В целом не требуется специально ставить "настоящий" gcc, для домашек достаточно имеющегося. Пример вывода у авторов курса (Apple M1):

```
$ gcc --version
Configured with:
   --prefix=/Applications/Xcode.app/Contents/Developer/usr

--with-gxx-include-dir=/Library/Developer/CommandLineTools/SDKs/MacOS
X.sdk/usr/include/c++/4.2.1
Apple clang version 12.0.0 (clang-1200.0.32.29)
Target: arm64-apple-darwin20.4.0
```

```
Thread model: posix
```

```
InstalledDir:
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xc
toolchain/usr/bin
```

Среда разработки

В общем случае для выполнения домашних заданий достаточно простого текстового редактора и компилятора.

Но мы крайне не рекомендуем разрабатывать в неполноценных IDE (nano, ~~vim~~, Sublime Text, ~~Emacs~~ и т.д.), особенно, если они плохо настроены. Часто это влечёт к примитивным ошибкам, связанным с отсутствием подсказок от среды, хоть и при должной сноровке ускоряет скорость разработки.

В течение курса мы неоднократно увидим, как здорово иметь IDE под рукой.

На данный момент наиболее популярными ср**е**дами являются:

- GoLand - платная, но очень мощная приблуда от JetBrains. К сожалению, получить лицензионный ключ от Stepik можно только для открытых курсов.
- Visual Studio Code - популярная, благодаря своей бесплатности, IDE. Расширение для Go официально поддерживается командой Google. Использует gopls - официальный language server, который вы, в принципе, можете запустить самостоятельно и внедрить в привычный вам редактор (инструкции здесь: поддерживаются VSCode, Vim/ Neovim, Emacs, Atom, Sublime Text, Acme).

Больше про редакторы и IDE – в официальной документации.

Напишите в комментариях, чем пользуетесь и что посоветовали бы коллегам!

github.com/golang/go

На курсе мы часто обращаемся к исходному коду стандартной библиотеки и самого компилятора Go.

Так как вы установили Go, его исходники (а не только бинарник компилятора) уже можно найти в системе:

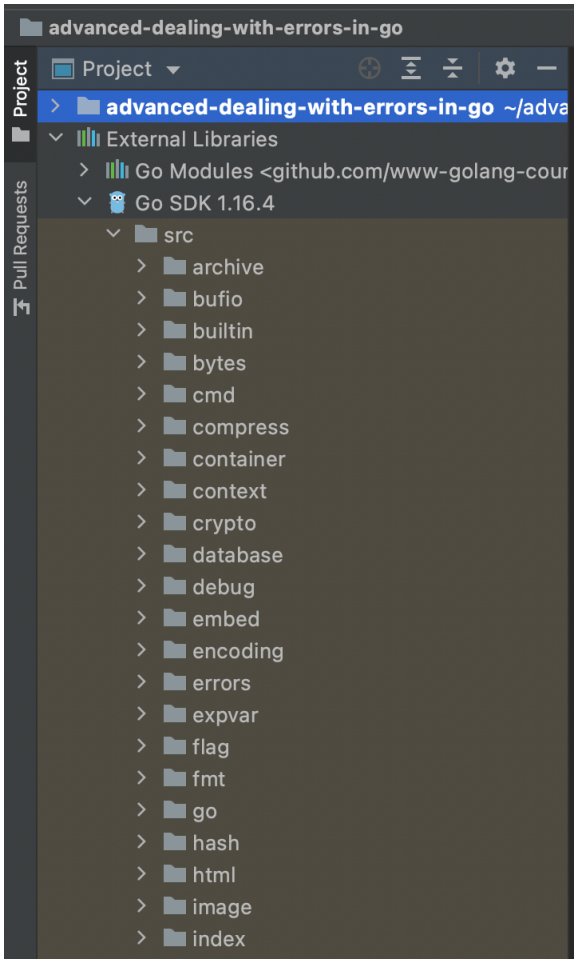
```
$ go env GOROOT
/usr/local/go

$ ls -1a $(go env GOROOT)
.
..
AUTHORS
CONTRIBUTING.md
CONTRIBUTORS
LICENSE
PATENTS
README.md
SECURITY.md
VERSION
api
bin
doc
favicon.ico
lib
```

```
misc
pkg
robots.txt
src

test
```

Например, в них можно копаться через GoLand:



Или можно выкачать нужную версию сорцов в удобное для себя место через https://github.com/golang/go.

Дополнительно не забываем про:

- спецификацию языка Go;

- документацию стандартной библиотеки языка Go;
- подсказки по написанию эффективного кода на Go;
- ответы на часто задаваемые вопросы про Go.

Заключение

Теперь, когда всё выкачено, установлено и настроено, можно переходить к работе с ошибками в Go.

Начнём с истоков, а именно, познакомимся с **концепцией обработки ошибок в Си**.