

Peter the Great
Saint-Petersburg Polytechnic University



Тема работы

Golang *"Hello world"*



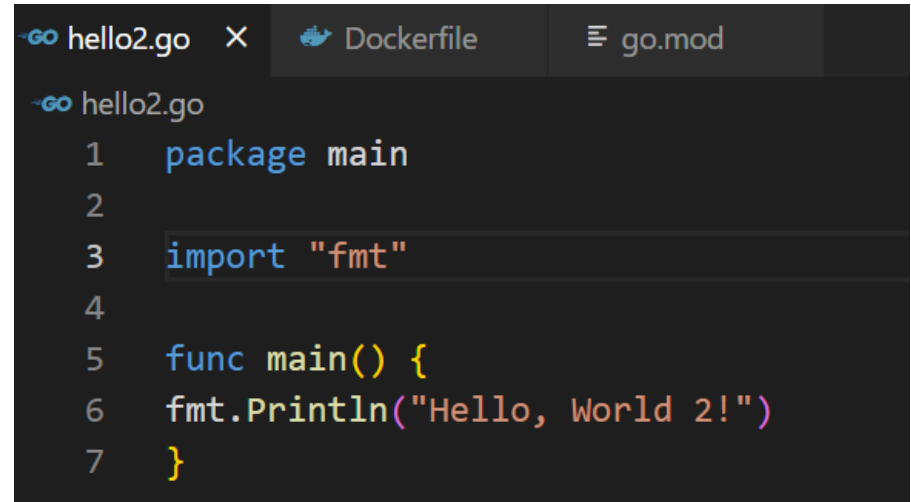
Исполнители: учащиеся группы
5030102/00201

Руководитель: Иванов Денис Юрьевич

12.09.23

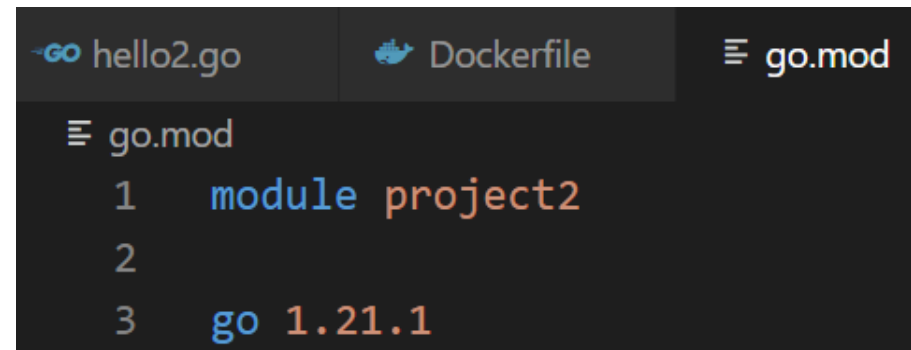
Написание кода

- ▶ Расширение Go для VS Code
 - ▶ [Go with Visual Studio Code](#)
- ▶ Код программы "Hello, World!"
 - ▶ hello.go - файл с кодом
 - ▶ go.mod - содержит метаданные о модуле
 - ▶ \$ go mod init main - для создания .mod файла



The screenshot shows the VS Code editor with three tabs: 'hello2.go', 'Dockerfile', and 'go.mod'. The 'hello2.go' tab is active, displaying the following Go code:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hello, World 2!")
7 }
```

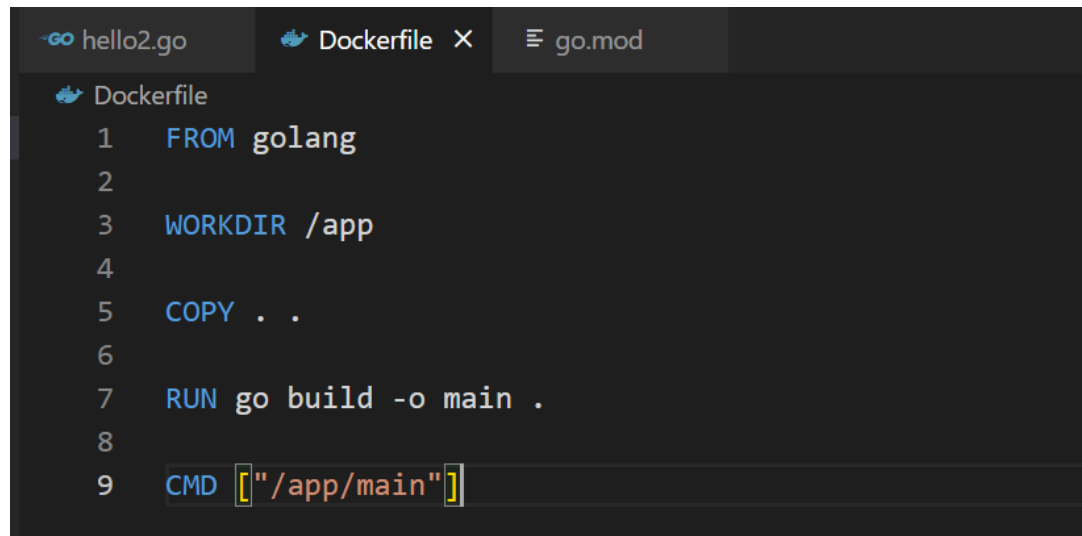


The screenshot shows the VS Code editor with the same three tabs. The 'go.mod' tab is active, displaying the following Go module definition:

```
1 module project2
2
3 go 1.21.1
```

Запуск в docker-контейнере (1/2)

- ▶ Загрузка docker-образа языка
 - ▶ `docker pull golang`
 - ▶ [golang - Official Image | Docker Hub](#)
- ▶ Создаем docker-файл
 - ▶ В образ включится последняя версия образа golang

A screenshot of a code editor with a dark theme. The editor has three tabs at the top: 'hello2.go', 'Dockerfile', and 'go.mod'. The 'Dockerfile' tab is active. The code in the Dockerfile is as follows:

```
1 FROM golang
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN go build -o main .
8
9 CMD ["/app/main"]
```

Запуск в docker-контейнере (2/2)

- ▶ Сборка и запуск docker-образ и контейнер
 - ▶ `docker build -t hello:v1 .`
 - ▶ `docker run --name hello_cont -d gomain:v1`
 - ▶ `docker logs -f hello_cont`

```
● PS C:\Users\misha\vs code projects\go_projects> docker run --name hello_cont -d mshnschnko/hello:v1
f97c0f1f7e860ed012013d89a06899966930da535844de4c79acdd93f894198c
● PS C:\Users\misha\vs code projects\go_projects> docker logs -f hello_cont
Hello, world!
```

Дебаг (1/3)

- ▶ Инструменты для дебага

- ▶ Delve

- ▶ [go-delve/delve: Delve is a debugger for the Go programming language. \(github.com\)](https://github.com/go-delve/delve)

- ▶ GDB

- ▶ [Debugging Go Code with GDB - The Go Programming Language](#)

Дебаг (2/3)

- ▶ Delve

- ▶ Установка

- ▶ `$ go install github.com/go-delve/delve/cmd/dlv@latest`

- ▶ Проверка корректности установки

- ▶ `$ dlv version`

- ▶ Запуск debug-режим

- ▶ `$ dlv debug`

- ▶ Выход из debug-режима

- ▶ `$ exit`

Дебаг (3/3)

- ▶ Основные команды
 - ▶ break (break hello.go:6)
 - ▶ Задание точки останова
 - ▶ continue
 - ▶ Переход к ближайшей точке останова
 - ▶ next
 - ▶ Переход на следующую строку
 - ▶ step
 - ▶ Шаг с заходом в функцию
 - ▶ print (print n)
 - ▶ Вывод значения переменной
 - ▶ display (display n)
 - ▶ Отслеживание значения переменной
 - ▶ Флаг -a добавит переменные в список отображения. Флаг -d удалит из списка

```
(dlv) continue
> main.main() C:/Users/misha/vs code pro
  1: package main
  2:
  3: import "fmt"
  4:
  5: func main() {
=>  6:     fmt.Println("Hello!")
    7:     fmt.Println("World!")
    8:     fmt.Println("Whats up?")
    9: }
(dlv) next
Hello!
> main.main() C:/Users/misha/vs code pro
  2:
  3: import "fmt"
  4:
  5: func main() {
  6:     fmt.Println("Hello!")
=>  7:     fmt.Println("World!")
    8:     fmt.Println("Whats up?")
    9: }
(dlv) █
```