

Golang



2020



Информация

Данная публикация стала возможной благодаря помощи американского народа, оказанной через Агентство США по международному развитию (USAID). Алиф Академия несёт ответственность за содержание публикации, которое не обязательно отражает позицию USAID или Правительства США.



ЦЕЛИ КУРСА



Цели курса

Цель курса достаточно простая - научить вас решать реальные задачи с использованием языка Golang (иногда мы его будем называть Go).

Это важно: не олимпиадные, не логические, не математические, а реальные, которые ставятся в промышленной разработке с использованием промышленных инструментов.



0 курсе

Курс состоит из 33 обязательных занятий. Мы будем проводить онлайн-вебинары каждую среду и пятницу в 18:30 (по Душанбе), а во вторник вечером будем выкладывать видеоурок с разбором ДЗ. По итогу онлайн-вебинара мы выкладываем PDF-материал.

В течении 24 часов мы выкладываем запись вебинара. Каждый вторник 12:00 дедлайн сдачи домашнего задания.

Все вопросы вы сможете задавать в телеграмм-чате https://t.me/online_academy_go

Если не успеете сдать в срок домашнее задания, тогда этот курс будет для вас закончен и вы сможете зарегистрироваться на запуск следующего через несколько месяцев.



GOLANG



Golang

Golang (иногда просто Go) - язык, созданный в компании Google для создания высокопроизводительных приложений и упрощения процесса разработки программного обеспечения.

В статье <https://talks.golang.org/2012/splash.article> описаны основные причины создания языка:

- разработка сложного ПО требовала слишком высокой квалификации
- медленно внедрялись новые функции
- использовалось слишком много разных языков
- у разработчиков был слишком разный уровень



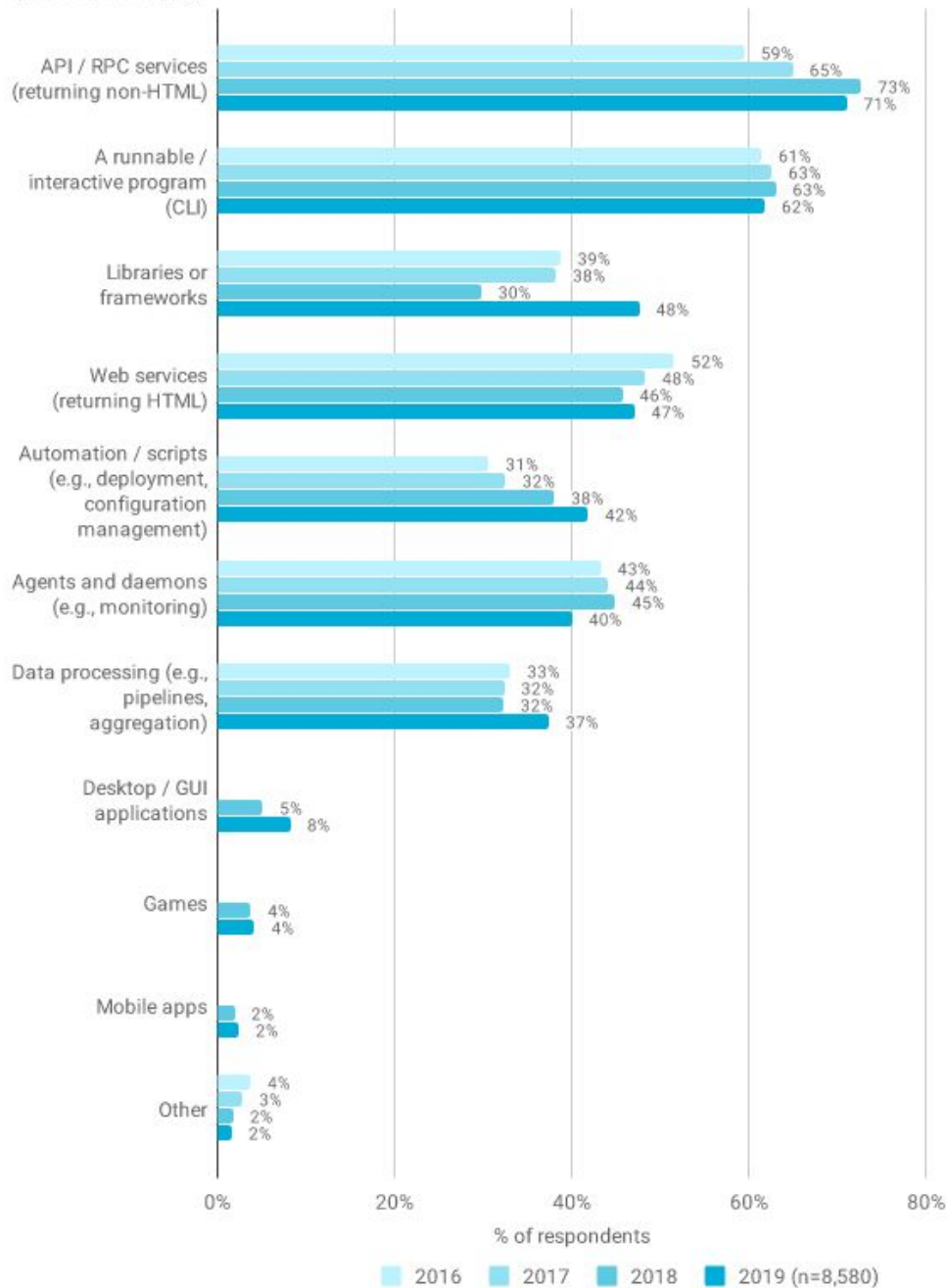
Golang

В итоге был разработан язык, который достаточно простой, чтобы выучить его за короткое время, и достаточно мощный, чтобы сразу начать разрабатывать промышленные приложения.



I write the following in Go:

(select all that apply)



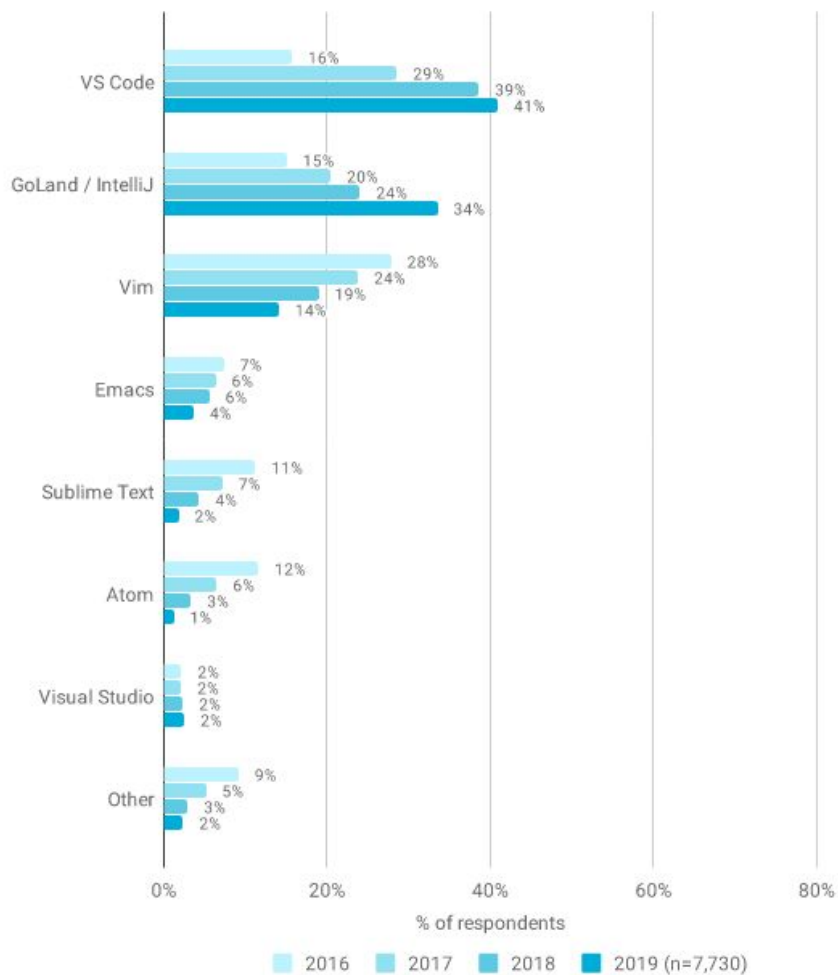
Что делают на Go

(blog.golang.org/survey2019-results):



Golang

Which editor do you most prefer to use when working with Go?



В каком редакторе пишут на Go

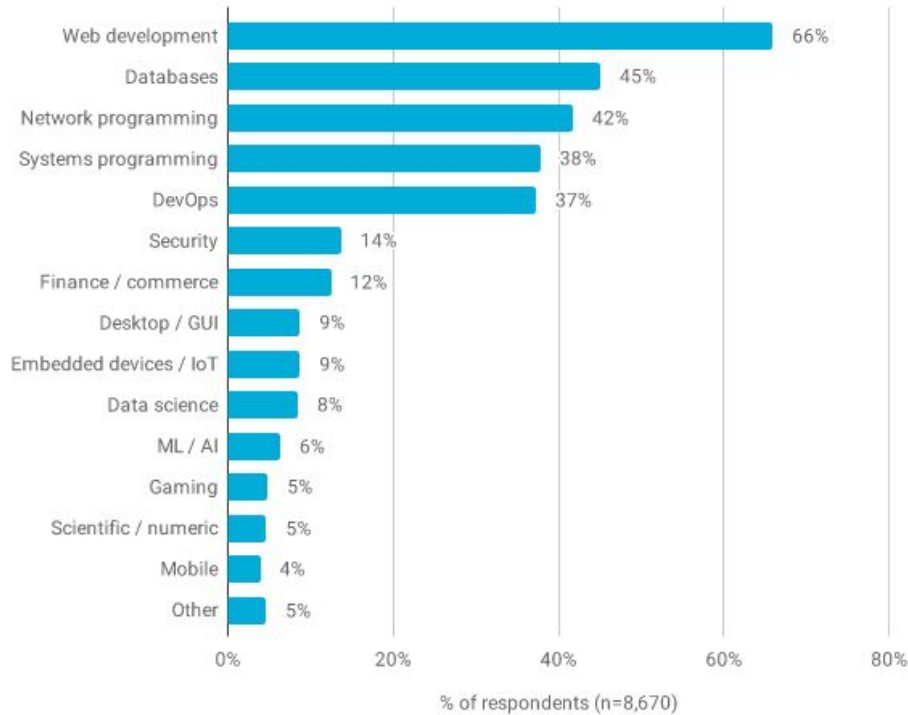
(blog.golang.org/survey2019-results):



Golang

I work with Go in the following areas:

(select all that apply)



В каких сферах работают на Go

(blog.golang.org/survey2019-results):



История версий

- 2007 - появление
- 2012 - version 1
- 2013 - version 1.1
- ...
- version 1.12 - 2019
- version 1.13 - 2019
- version 1.14 - 2020
- **version 1.15 - 2020** ← мы будем работать с этой версией



ИНСТРУМЕНТЫ



Инструменты

Для прохождения основной части курса вам понадобятся два инструмента: Golang, редактор VS Code и Git (для дополнительных лекций понадобятся также другие инструменты).

Кроме того, важны следующие три момента:

1. У вас должны быть права администратора на компьютере (чтобы вы могли устанавливать программы)
2. Ваш пользователь должен называться по-английски, без пробелов в имени (если это не так – переименуйте)
3. Создавайте все проекты где-нибудь на диске **C:**, например, в каталоге **projects** (следите за тем, чтобы в именах каталогов и файлов не было пробелов, не английских символов и т.д.)

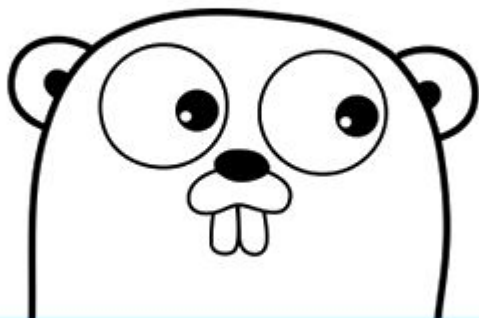
В любом случае, если у вас возникнут проблемы, пишите в канал курса.



Golang

Для работы с Go нужен сам Go, поэтому переходим на сайт <https://golang.org>, и нажимаем на кнопку Download Go:

Go is an open source programming language that makes it easy to build **simple, reliable, and efficient** software.



Download Go

Binary distributions available for
Linux, macOS, Windows, and more.

1



Golang

На открывшейся странице выберите свою ОС (мы всё будем рассматривать на примере Windows, если у вас другая ОС - пишите в группу):

Featured downloads

Microsoft Windows

Windows 7 or later, Intel 64-bit processor

[go1.15.windows-amd64.msi](#) (115MB)

Apple macOS

macOS 10.12 or later, Intel 64-bit processor

[go1.15.darwin-amd64.pkg](#) (117MB)

Linux

Linux 2.6.23 or later, Intel 64-bit processor

[go1.15.linux-amd64.tar.gz](#) (116MB)



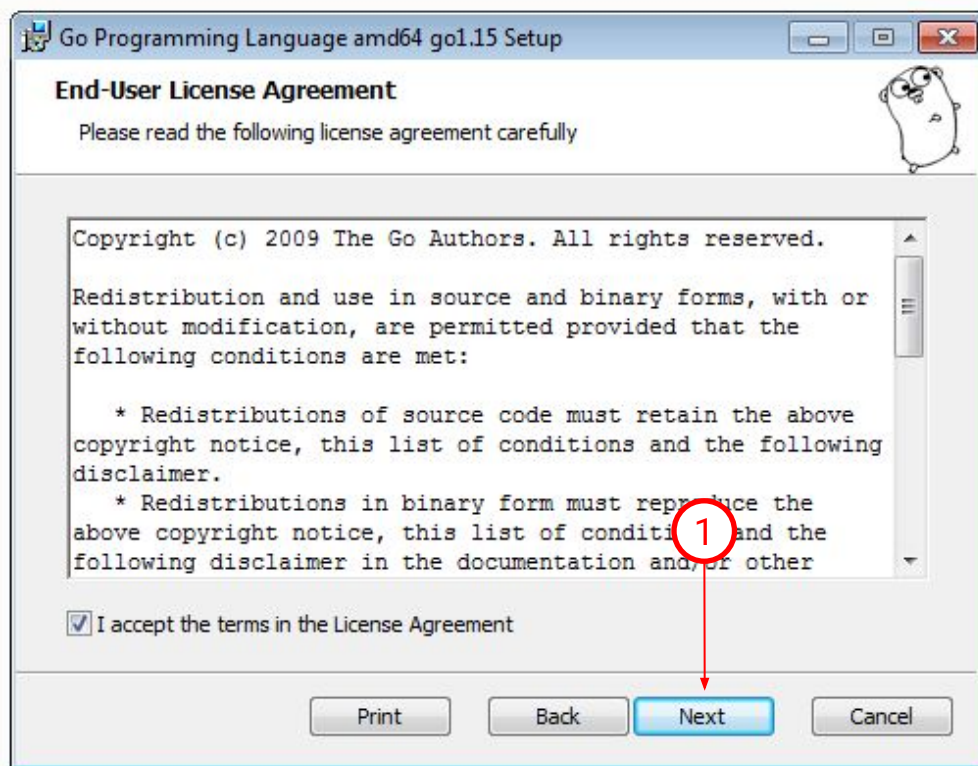
Golang

Запустите файл на установку и нажмите кнопку "Next":



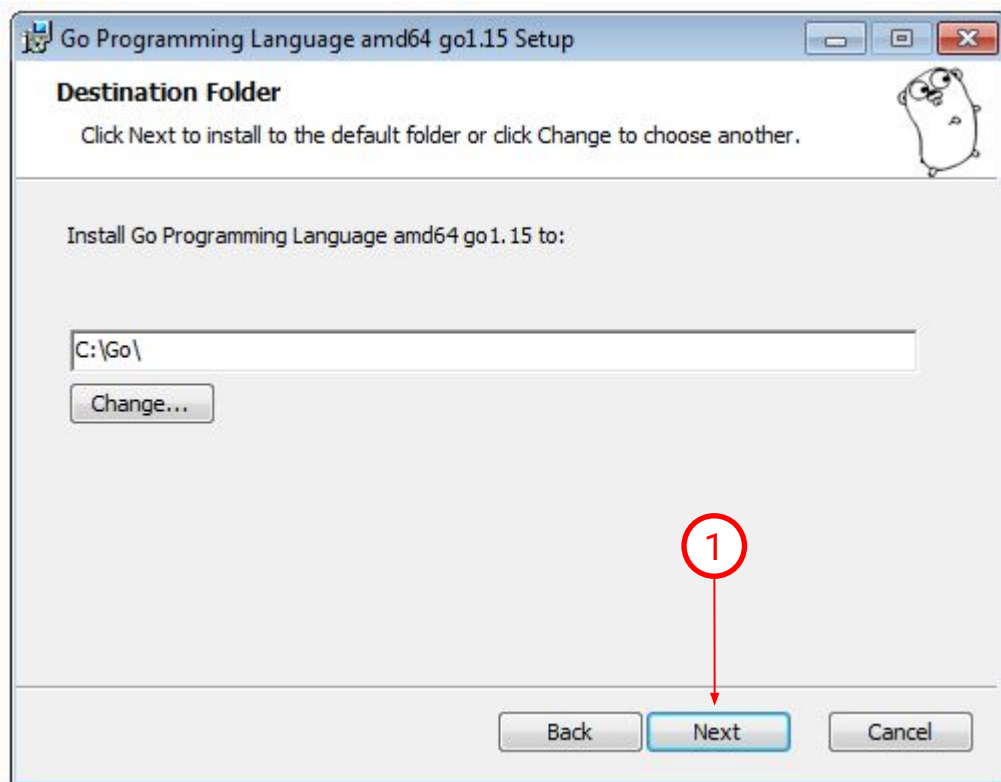
Golang

Прочитайте и примите условия лицензионного соглашения, после чего нажмите на кнопку "Next":



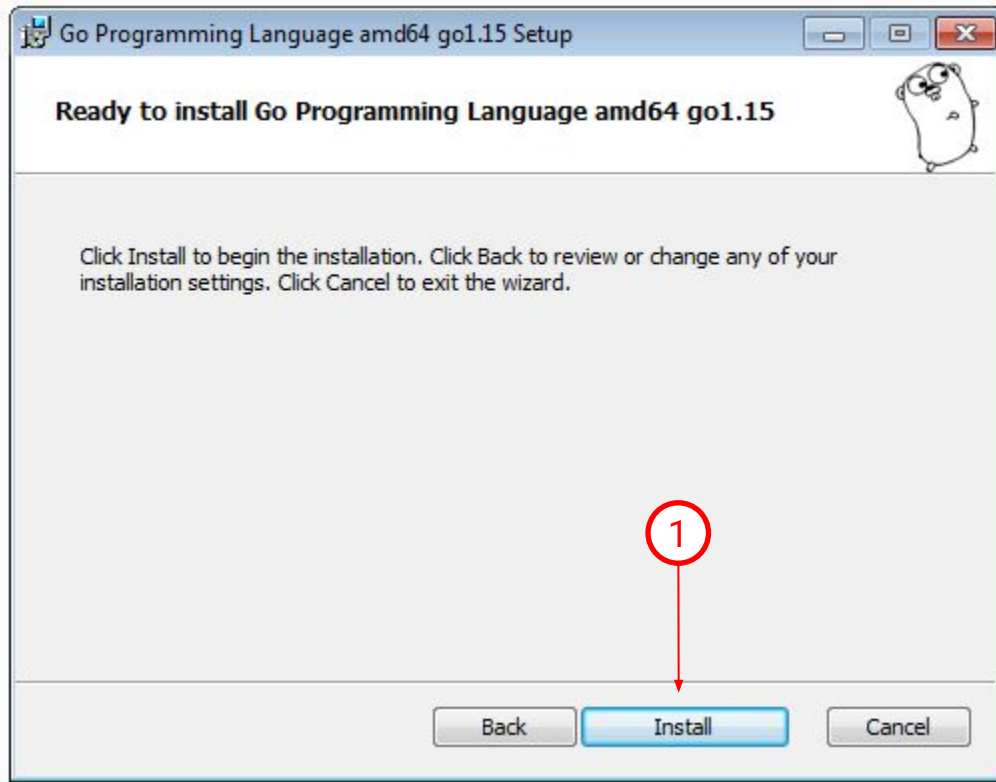
Golang

Оставьте значение по умолчанию и нажмите на кнопку "Next":



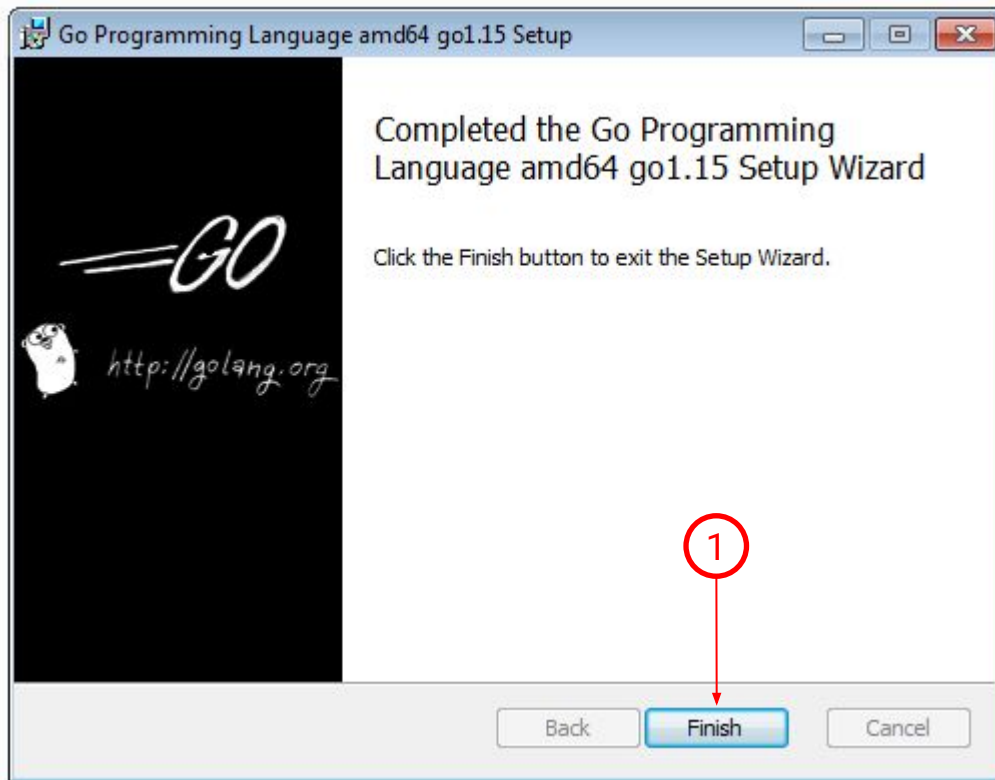
Golang

Подтвердите установку, нажав на кнопку "Install":



Golang

Дождитесь завершения установки и нажмите кнопку "Finish":



Git

Для установки Git перейдите по адресу <https://git-scm.com/downloads> и выберите установочный файл для вашей операционной системы (далее – ОС). Обычно, ОС определяется автоматически и вам нужно лишь нажать на кнопку скачивания:



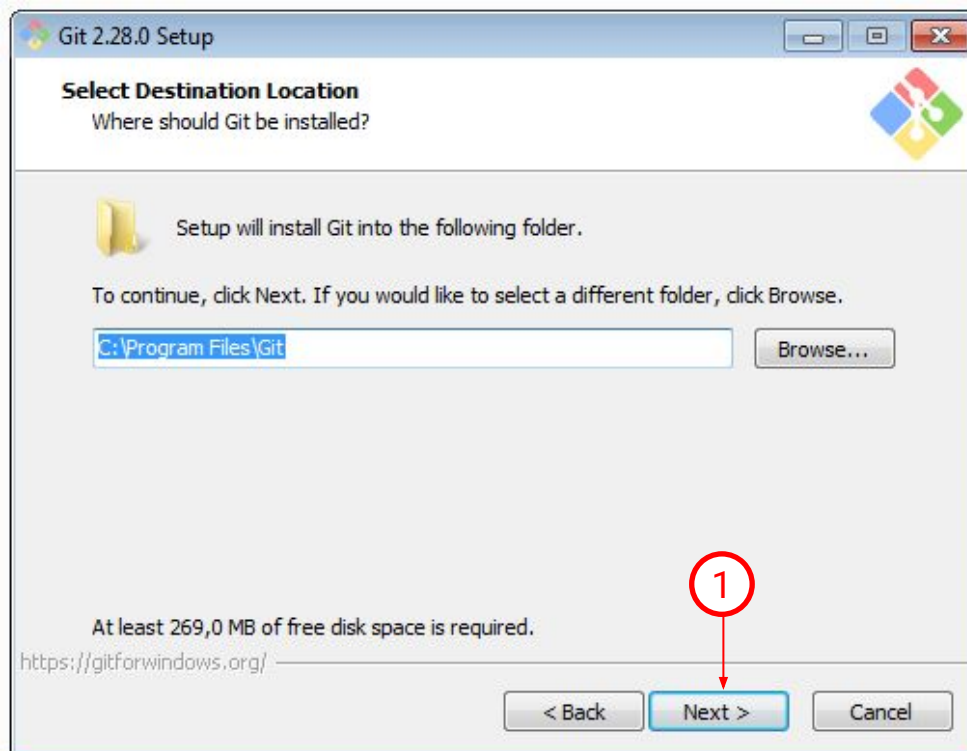
Git

Дождитесь загрузки файла и откройте его, прочитайте и согласитесь с условиями лицензии (нажмите кнопку "Next"):



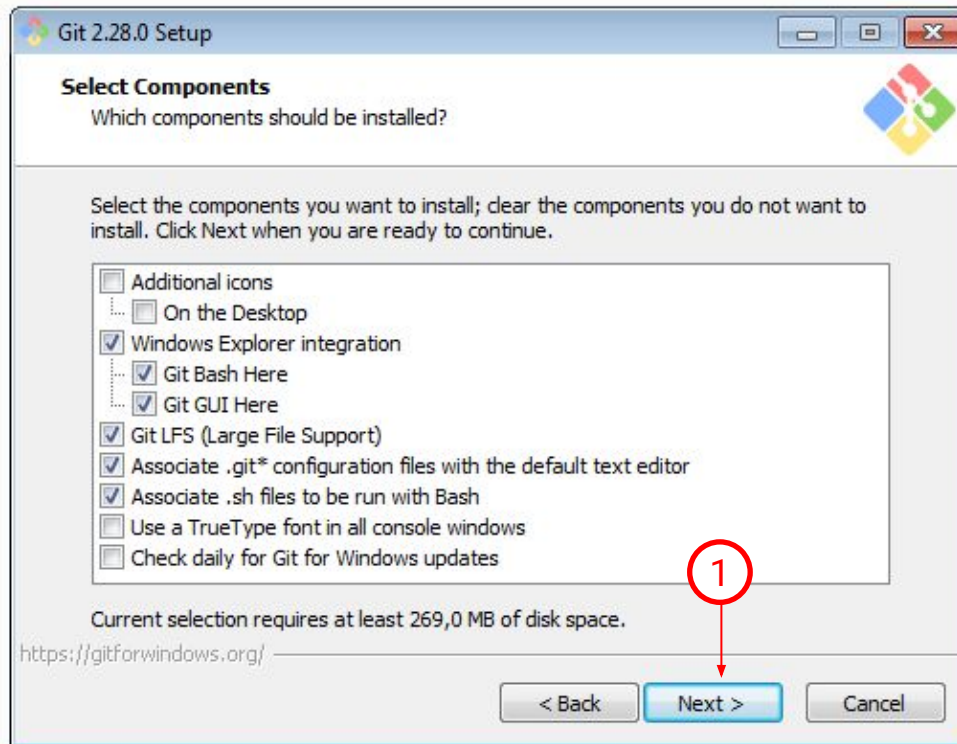
Git

Подтвердите установку в указанный каталог:



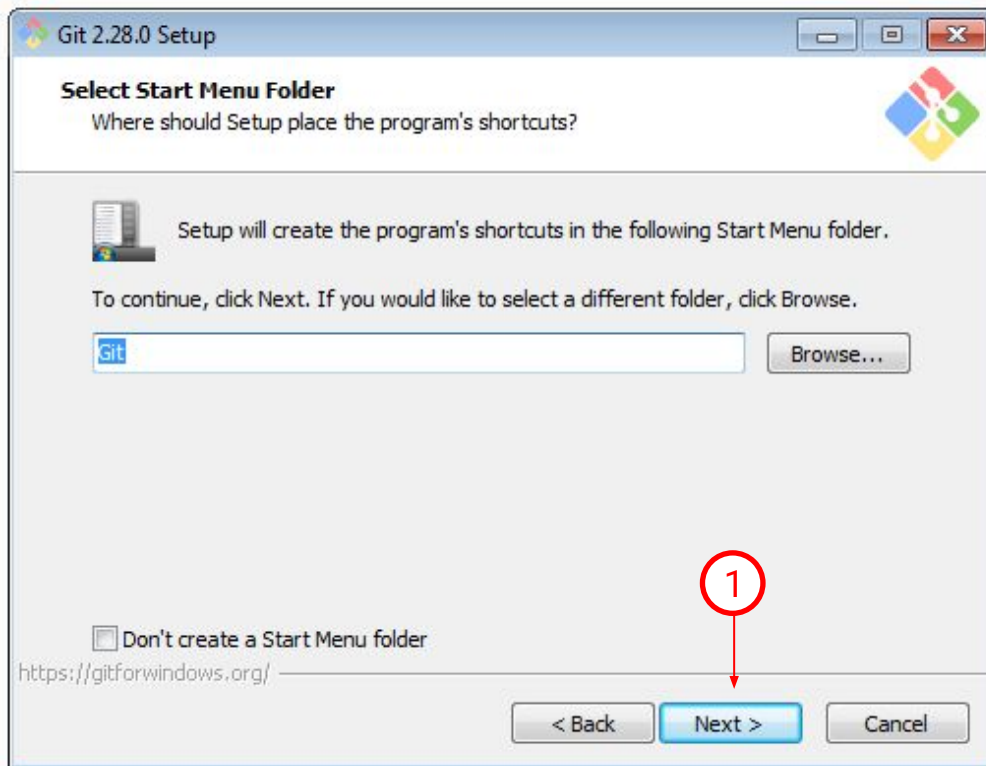
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



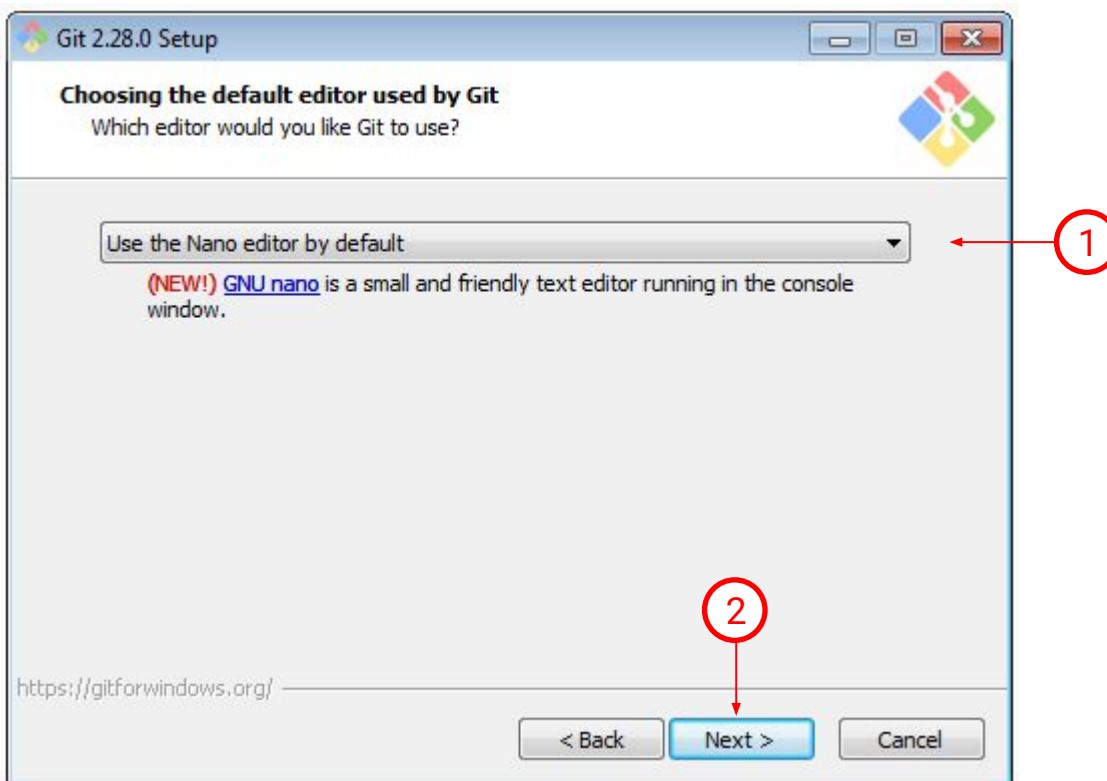
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



Git

В следующем окне выберите Nano вместо Vim (вы можете оставить Vim, если захотите научиться им пользоваться) и нажмите "Next".



Vim

Vim - это мощный текстовый редактор, позволяющий вам максимально эффективно работать с текстом (кодом в том числе). Он достаточно тяжёл для освоения, но если вы научитесь им пользоваться, то ваша скорость и эффективность работы с текстом и кодом вырастет в несколько раз (по факту - вы уже не сможете работать без него).

Плагины, обеспечивающие функциональность Vim есть для всех популярных редакторов кода - от VS Code до онлайн-редакторов.



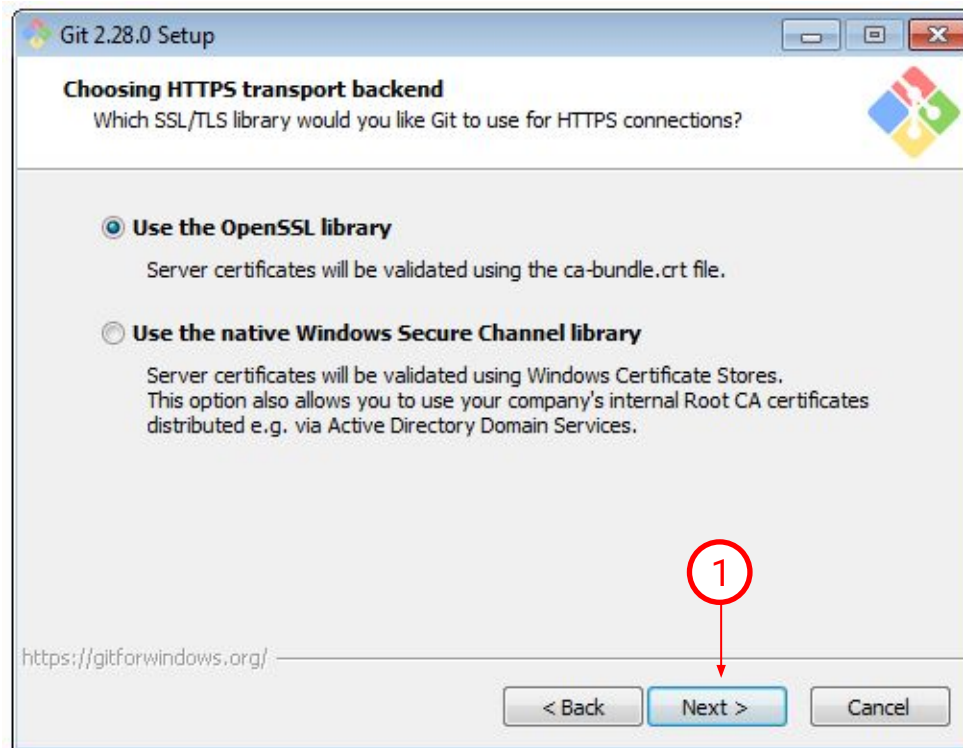
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



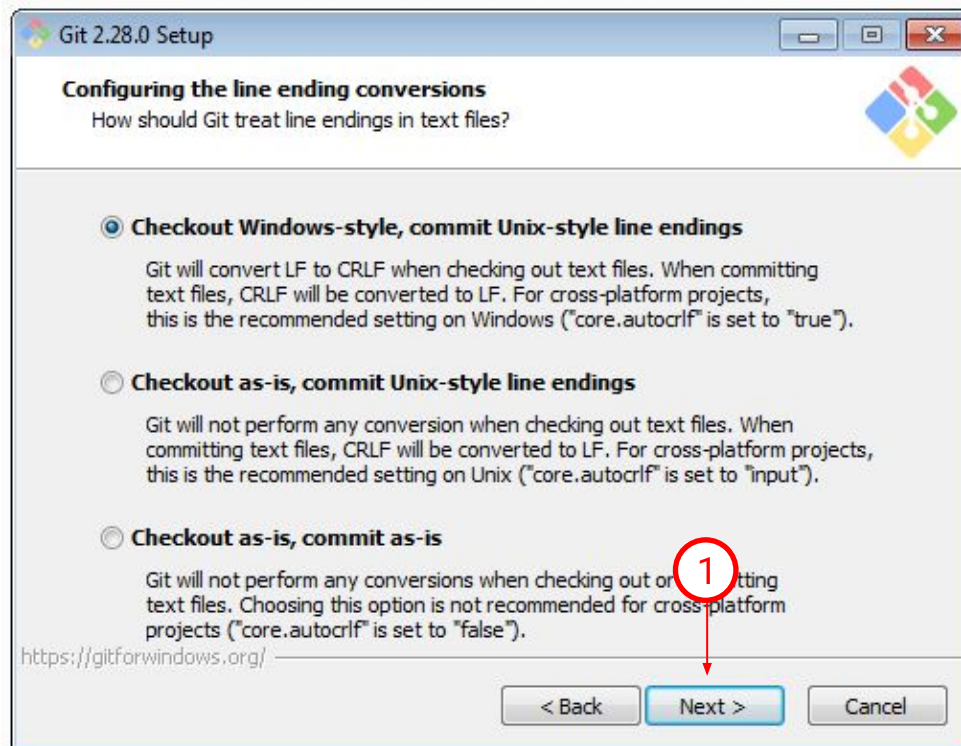
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



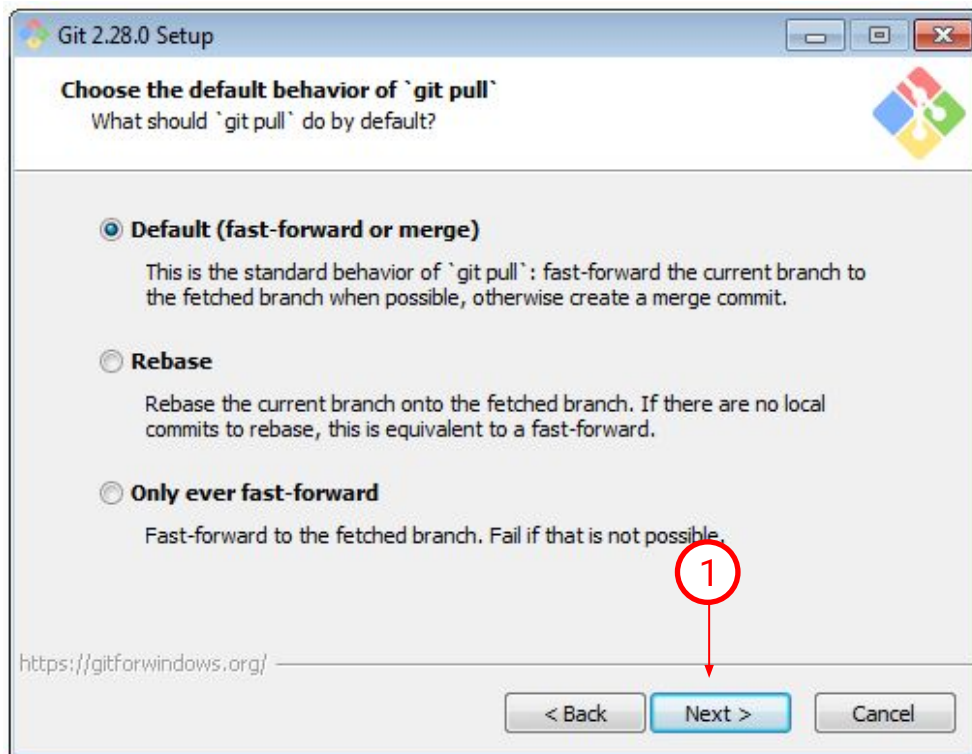
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



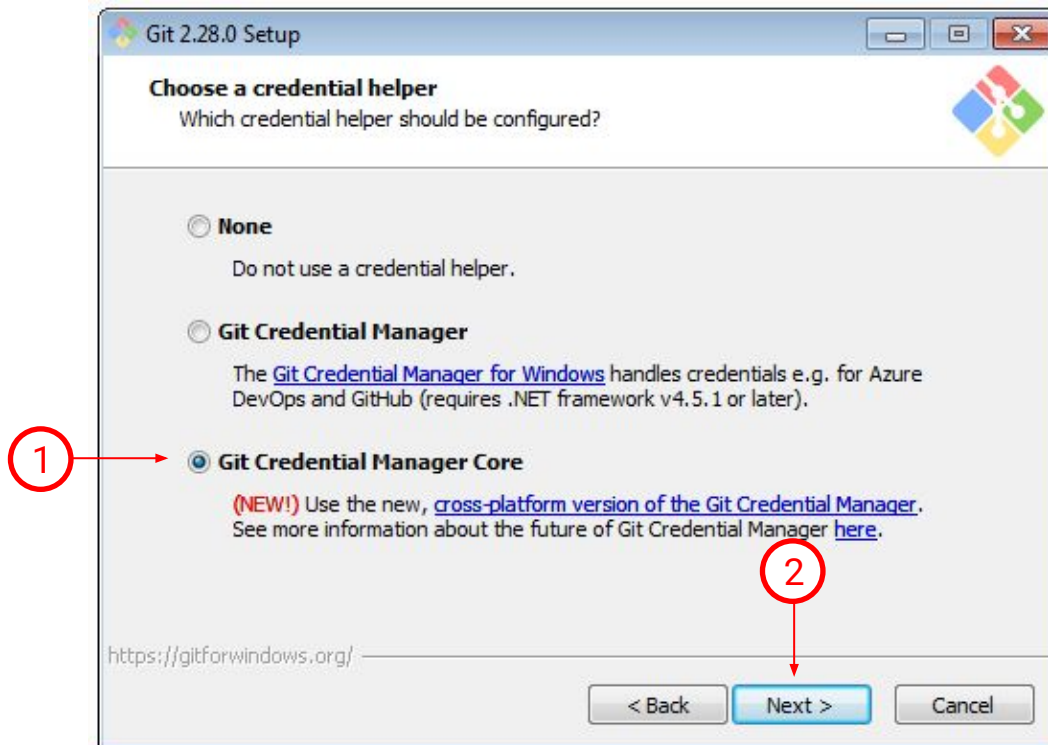
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



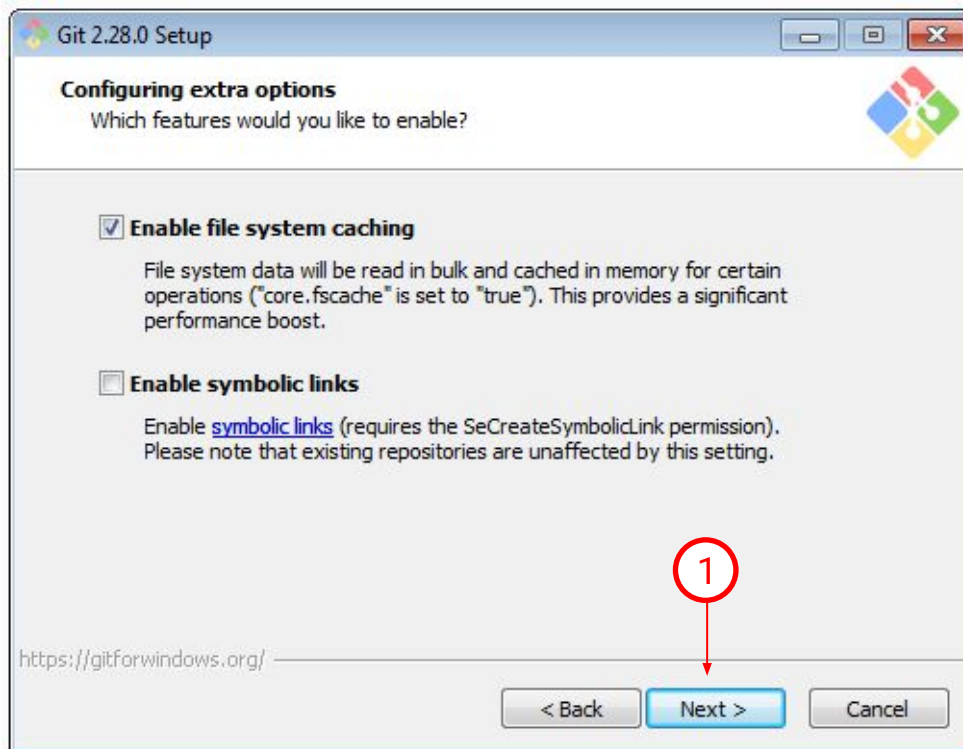
Git

Выберите Git Credential Manager Core и нажмите на кнопку "Next".



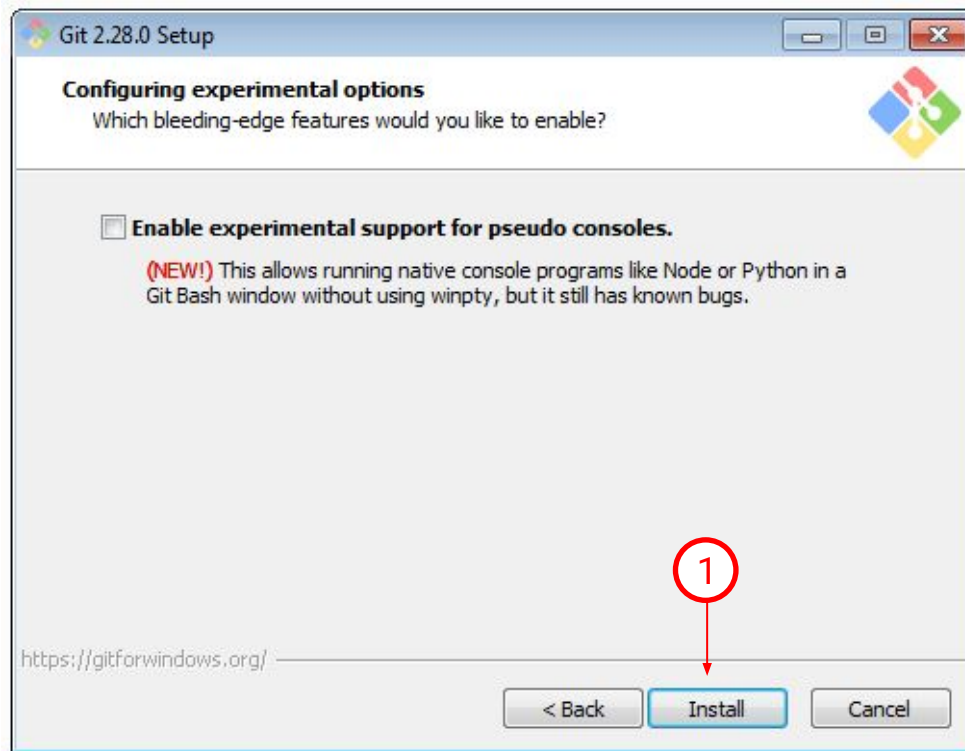
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Next":



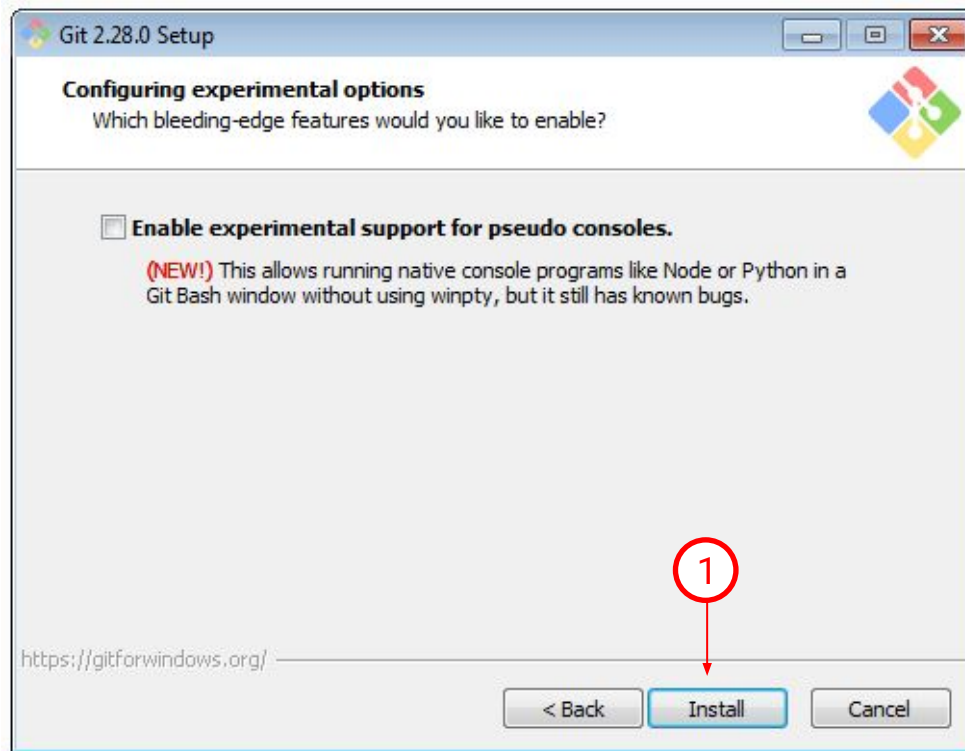
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Install":



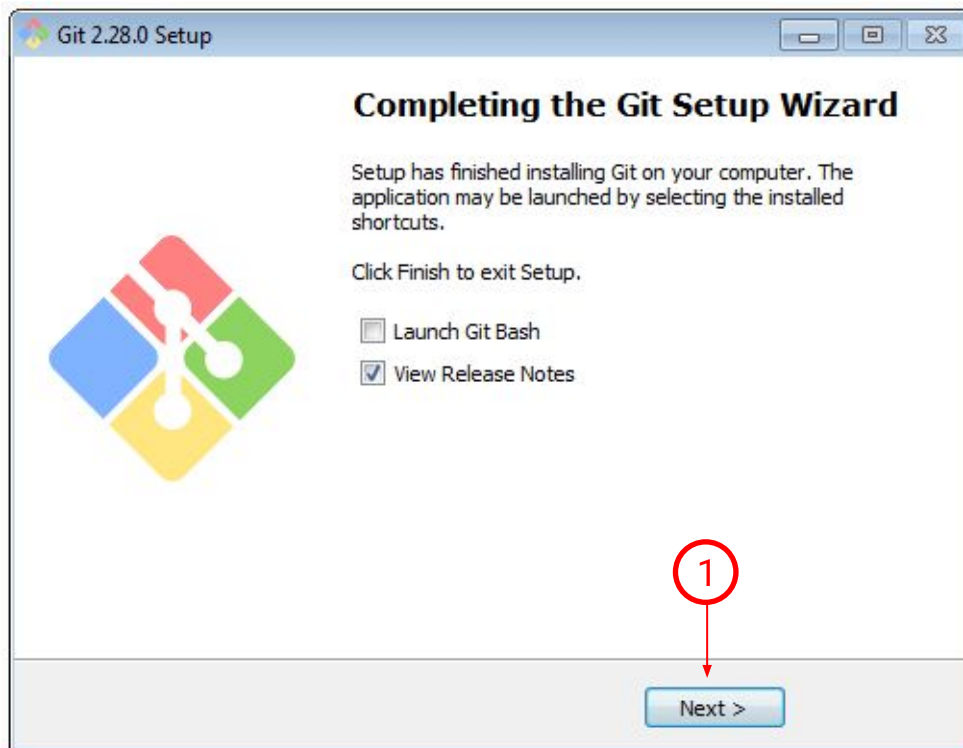
Git

Оставьте выбранные значения по умолчанию и нажмите на кнопку "Install":



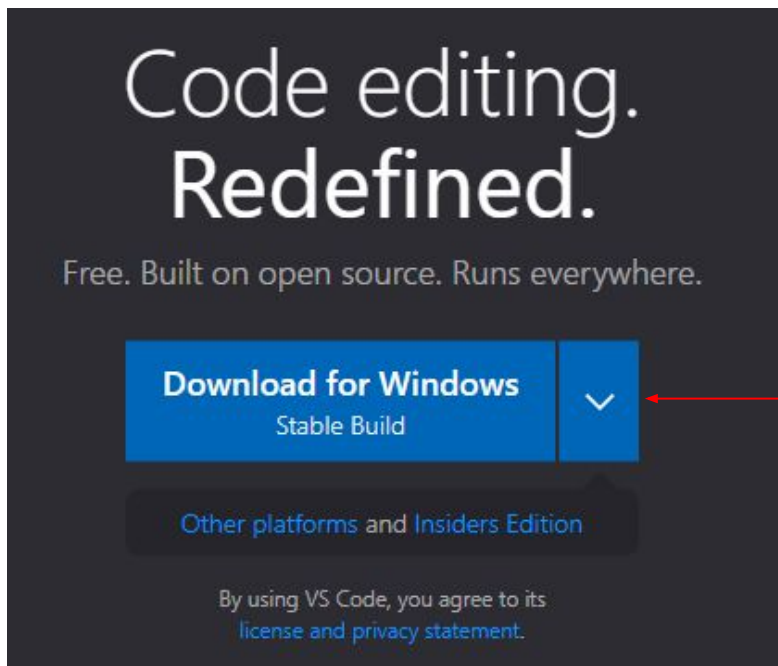
Git

Дождитесь завершения установки и нажмите кнопку "Next":



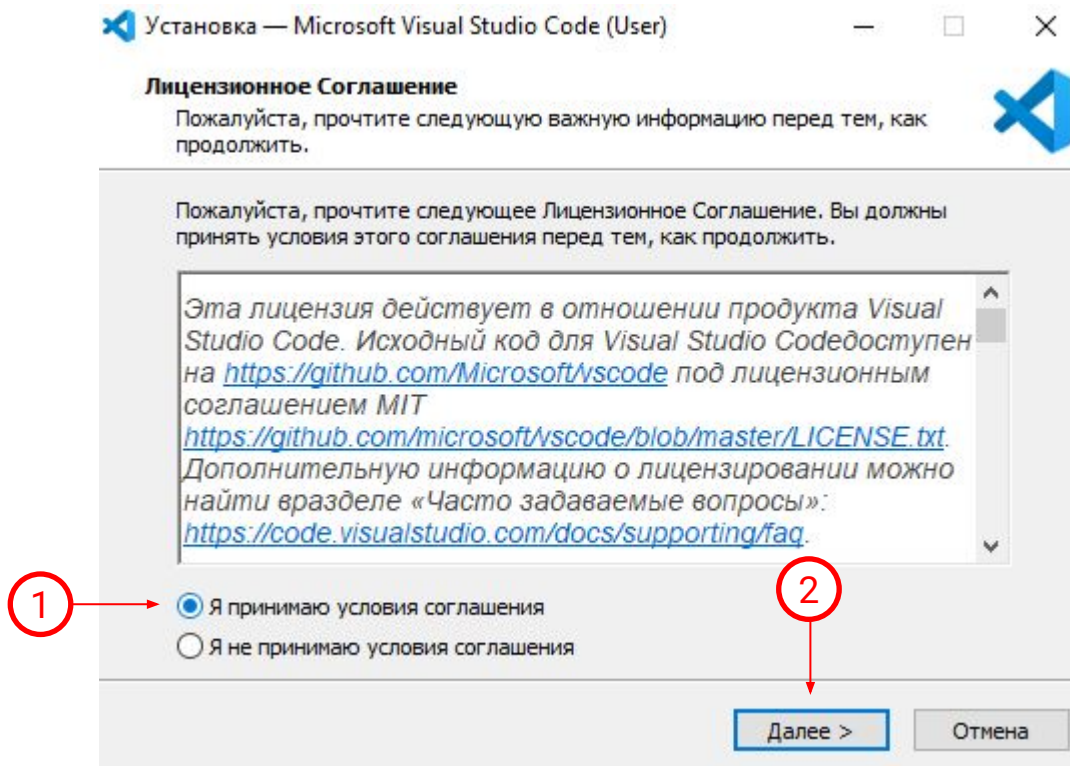
VS Code

VS Code самый популярный редактор кода (специальная программа, которая позволяет вам писать код) для JS. Перейдите по адресу code.visualstudio.com и скачайте установочный файл для вашей операционной системы. Например, для Windows, нужно выбрать Download For Windows (если у вас другая ОС – пишите в канал курса):



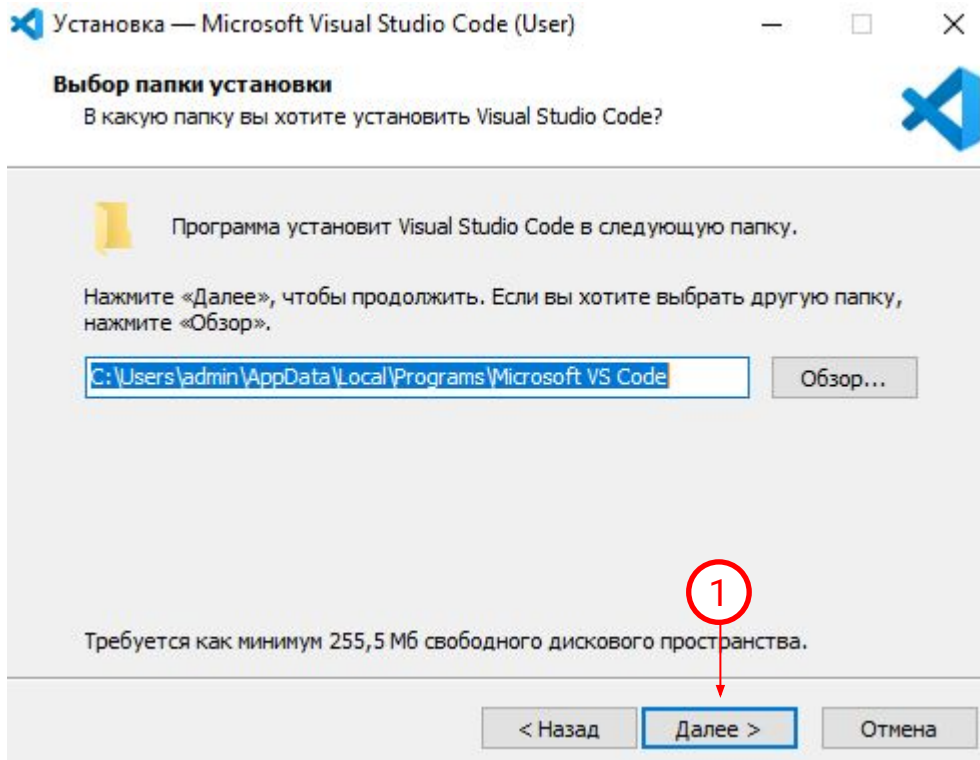
VS Code

Прочитайте и примите условия лицензионного соглашения, после чего нажмите на кнопку "Далее":



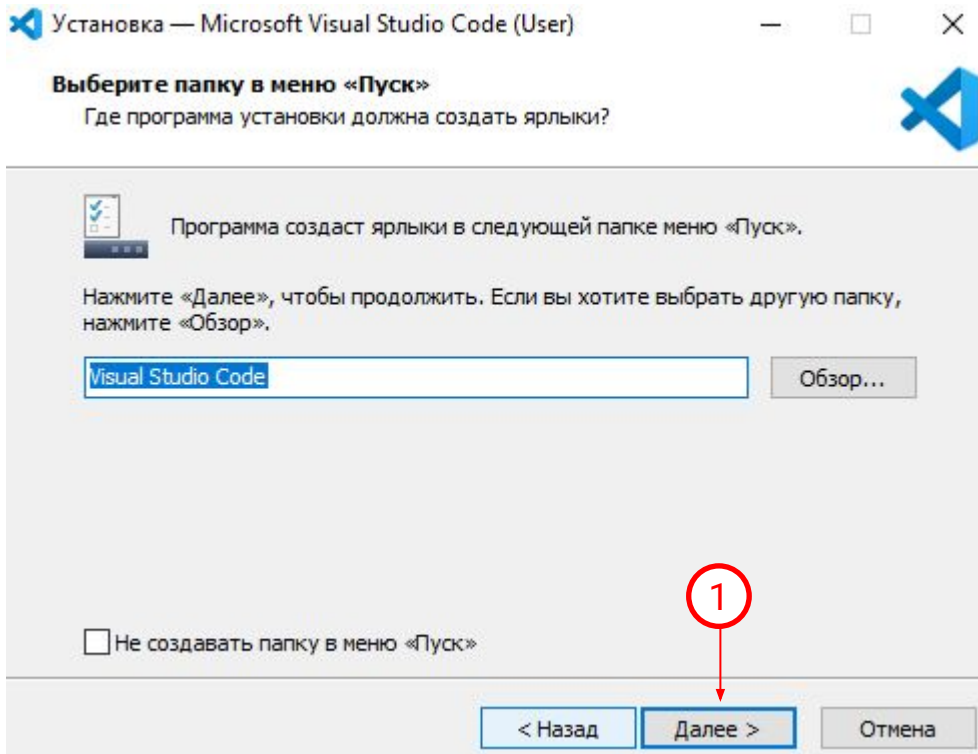
VS Code

Оставьте значения по умолчанию, после чего нажмите на кнопку "Далее":



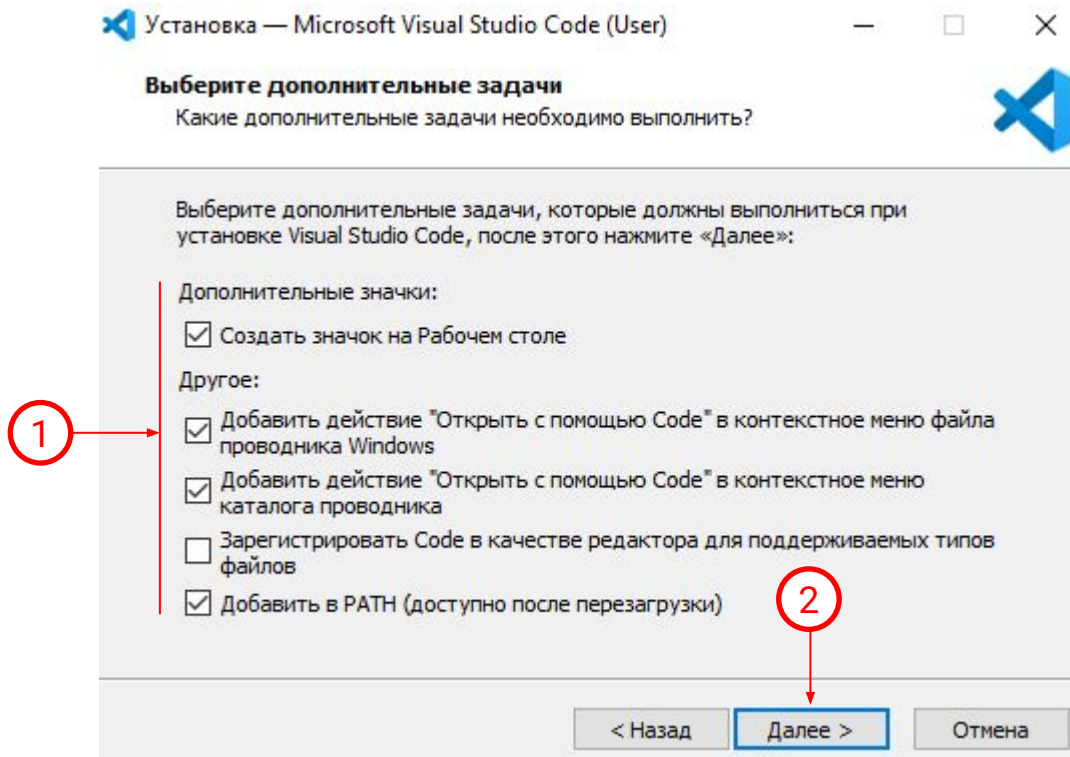
VS Code

Оставьте значения по умолчанию, после чего нажмите на кнопку Далее:



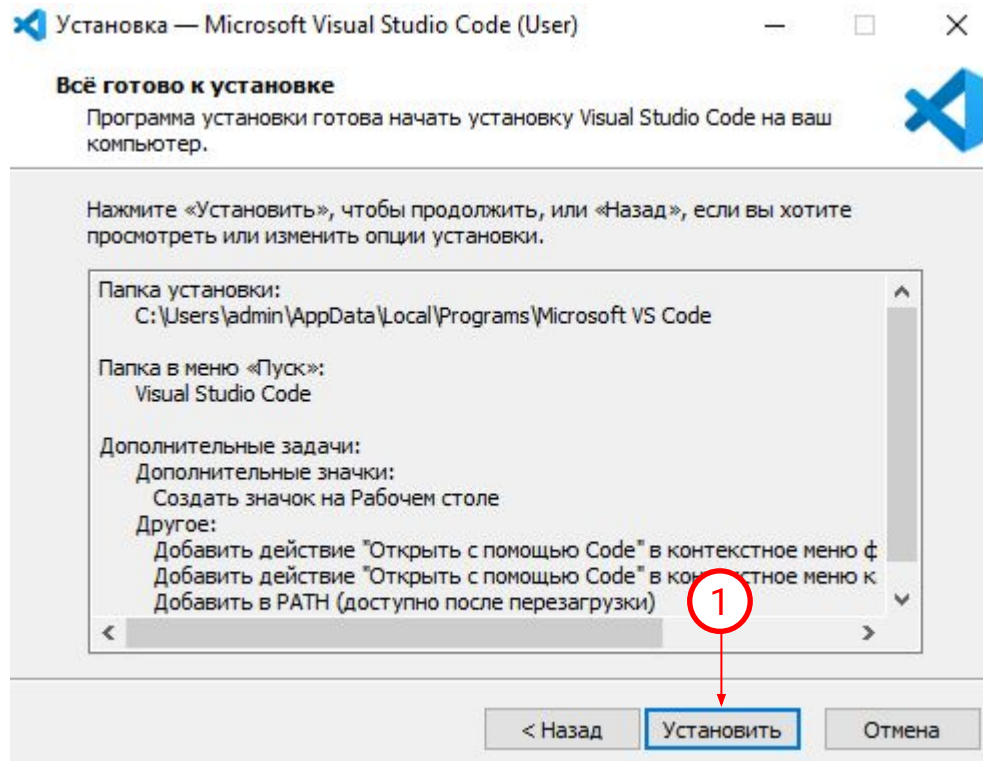
VS Code

Поставьте флажки как на скриншоте, после чего нажмите на кнопку Далее:



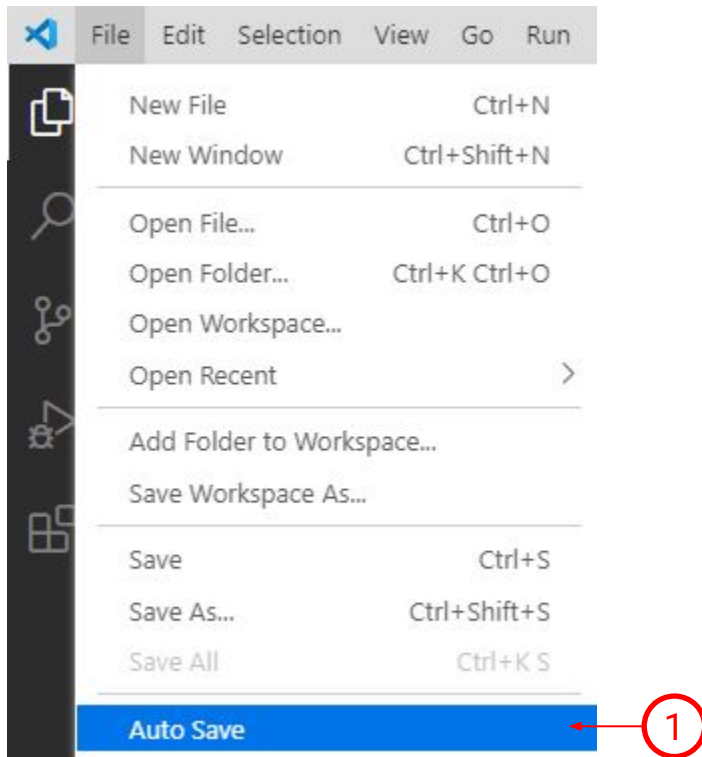
VS Code

Подтвердите установку, нажав на кнопку Установить:



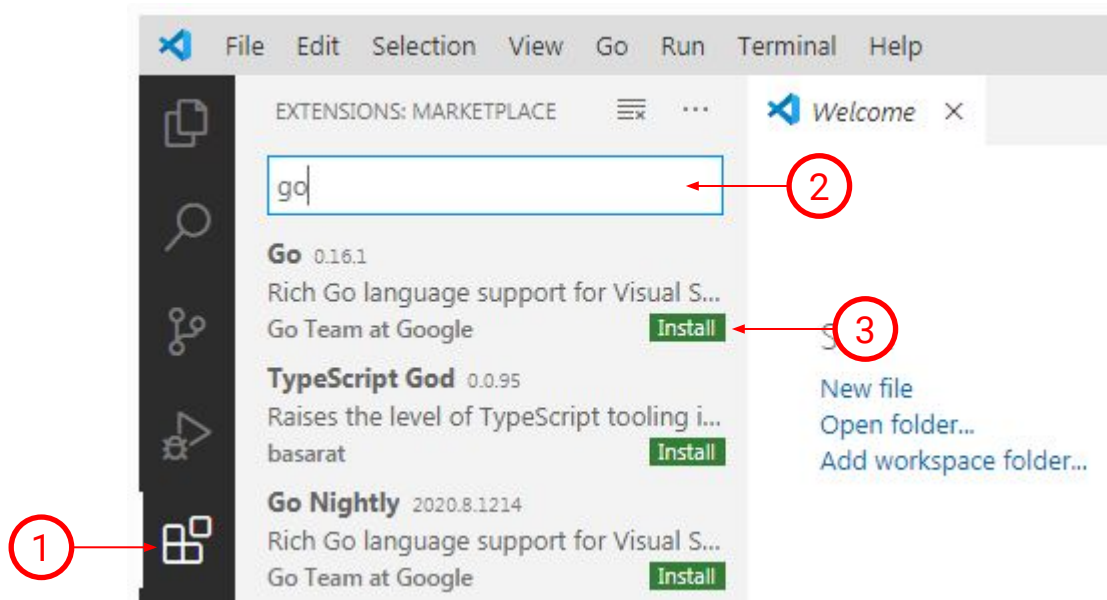
VS Code Auto Save

Чтобы изменения, которые вы вносите в файлы автоматически сохранялись, зайдите в пункт меню File и выберите Auto Save:



Go Extension

Зайдите в пункт меню расширений (1), введите в поисковую строку Go (2) и нажмите на кнопку Install для расширения Go (3):



Debugger

Нажмите **Ctrl + `** (там, где буква **ё**) и в открывшемся окне введите команду

go get github.com/go-delve/delve/cmd/dlv

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: cmd

Microsoft Windows [Version 6.1.7601]

(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\User>go get github.com/go-delve/delve/cmd/dlv

1

C:\Users\User>



Tools

Кроме того, нам понадобится ещё ряд инструментов, которые необходимо установить с помощью следующих команд (выполняйте их так же в терминале):

```
go get github.com/ramya-rao-a/go-outline
```

```
go get github.com/stamblerre/gocode
```

```
go get github.com/uudashr/gopkgs/v2/cmd/gopkgs
```

```
go get github.com/godocdr/godocdr
```

```
go get github.com/rogpeppe/godef
```

После чего закройте VS Code.



Git

В терминале VS Code выполните следующие команды (их нужно выполнить только один раз, а не в каждом проекте):

```
git config --global user.name "Student"
```

```
git config --global user.email "student@go.alif.academy"
```

Естественно, вместо `Student` и `student@go.alif.academy` вы можете использовать собственные значения.



HELLO WORLD



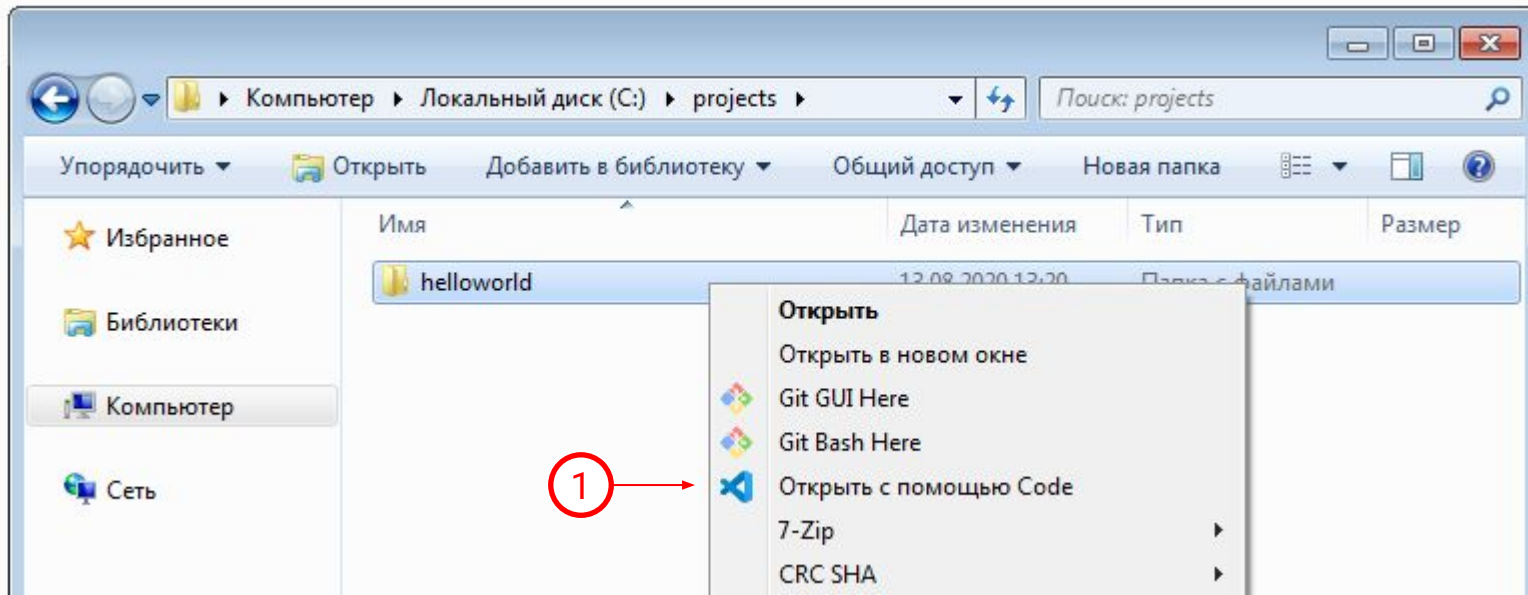
Hello World

Хватит теории, давайте создадим наш первый проект. По традиции это будет небольшая программа, которая печатает Hello World на экран.



Hello World

Создайте на диске **C:** каталог **projects**, внутри - **helloworld**. Кликните на нём правой кнопкой мыши и выберите Открыть с помощью Code:



Hello World

Важно: делайте все ваши проекты именно в `C:\projects`, а не где-то на рабочем столе! Некоторые инструменты, которые мы будем рассматривать не будут работать, если вы их поместите на рабочий стол или "запрячете" куда-то. Мы будем исходить из того, что все ваши проекты находятся в `C:\projects`.



Общая схема работы

Пока для нас схема работы будет выглядеть следующим образом:

1. В терминале (**Ctrl + `** в VS Code) инициализируем модуль командой
`go mod init <имя модуля>`
2. Создаём файл `main.go`
3. Пишем в нём (в файле `main.go`) код
4. Запускаем на исполнение
5. Повторяем п.2 - п.5



Модули

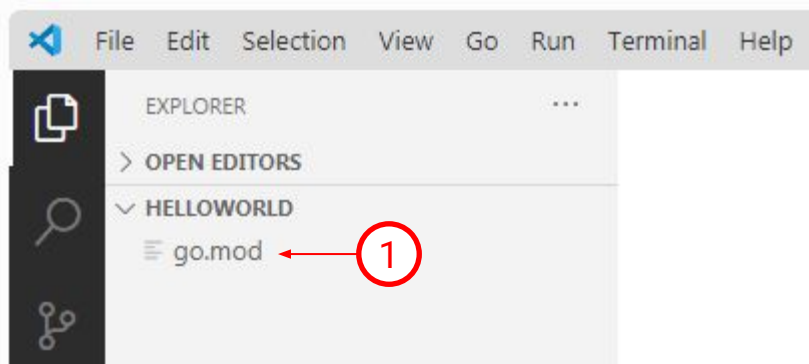
Проекты в Go существуют в виде модулей (что это такое мы будем разбирать чуть позже). Модуль инициализируется командой: `go mod init <имя модуля>`:

```
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
```

```
C:\projects\helloworld>go mod init helloworld
go: creating new go.mod: module helloworld
```

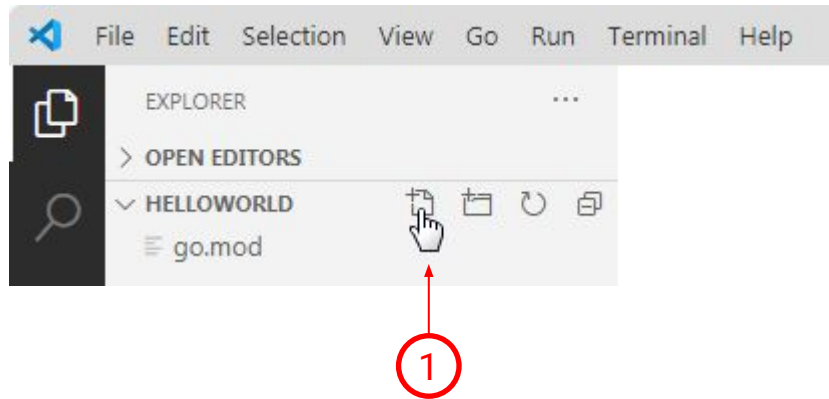
```
C:\projects\helloworld>
```

В результате выполнения этой команды в боковой панелике Explorer VS Code появится файл `go.mod`:

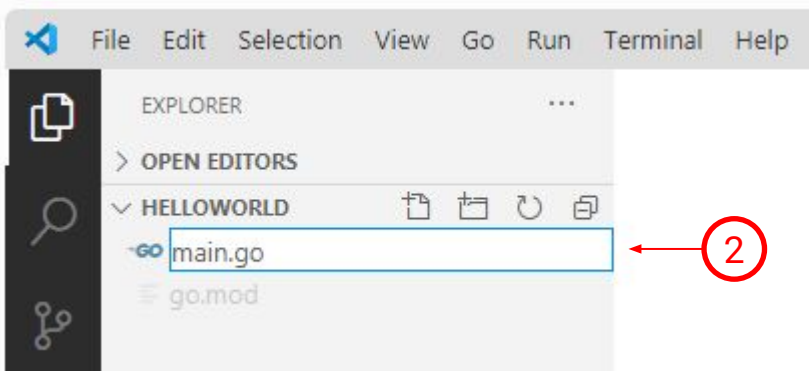


main.go

Там же, в боковой панельке нажмите на + для создания нового файла:

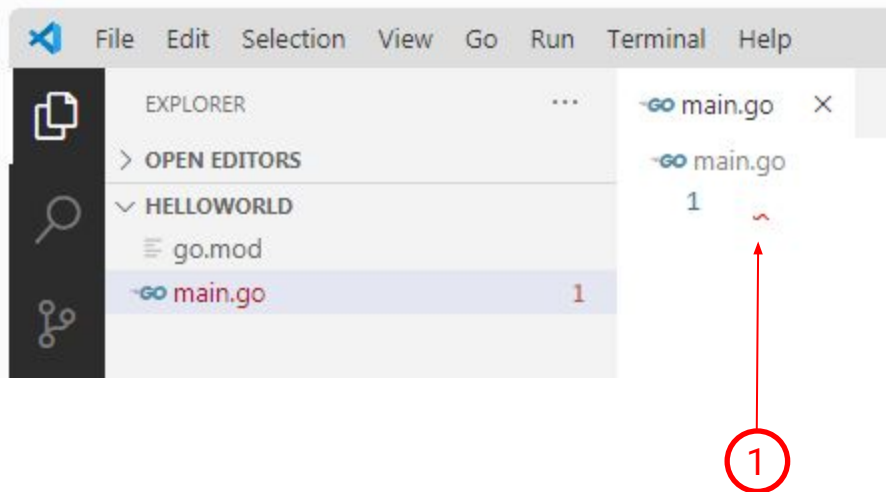


Введите `main.go` и нажмите на Enter:



main.go

После этого файл откроется для редактирования в основной части редактора:



Там мы и будем писать код.



main.go

Общая структура этого файла будет следующая:

1. Описание пакета (`package main`)
2. Функция `main`

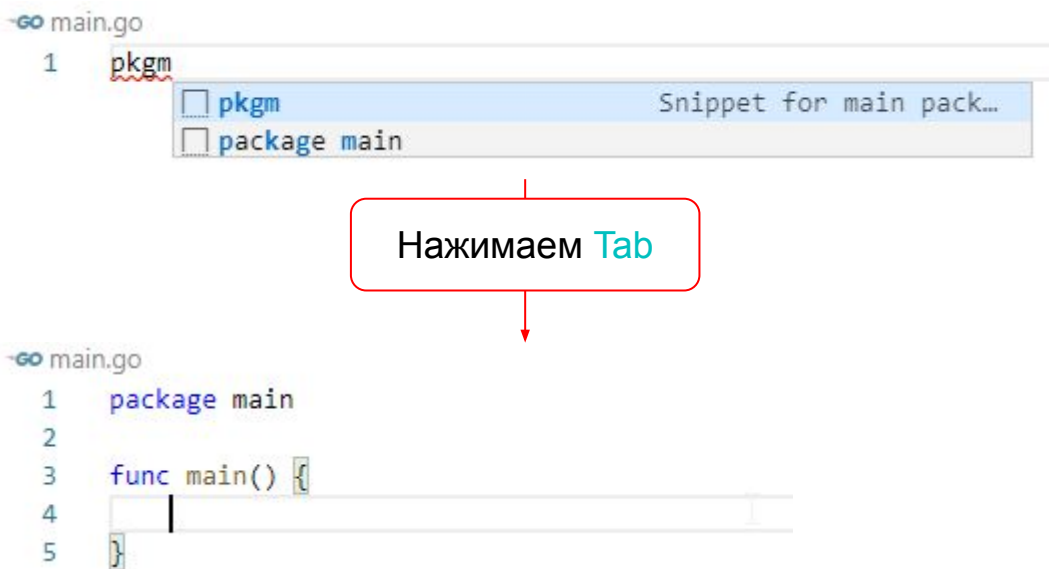
Важно: воспринимайте пока модули, пакеты и функцию `main` просто как шаблон.

На следующих лекциях мы с вами детально разберём, что это значит.



VS Code

VS Code (и установленные расширения) предоставляют вам значительное количество клавиатурных сокращений, которые позволят вам писать меньше кода. Например, сокращение **pkgm + Tab** приведёт к следующему:



Если вдруг выпадающая подсказка пропала, то вернуть её можно с помощью **Ctrl + пробел**. Важно: обязательно пользуйтесь подобными сокращениями!



VS Code

Мы воспользуемся специальной функцией `println`, которая позволяет выводить на экран данные (в нашем случае - строку):

```
go main.go > {} main > main
1  package main
2
3  func main() {
4      print
5  }
6
7
```



Автоподсказки также работают, вы можете выбрать нужный вариант с помощью стрелок и дописать уже руками ("Hello World"):

```
go main.go > ...
1  package main
2
3  func main() {
4      println("Hello world")
5  }
```



Запуск

Чтобы запустить нашу программу, нужно нажать **Ctrl + F5** и выбрать Go:

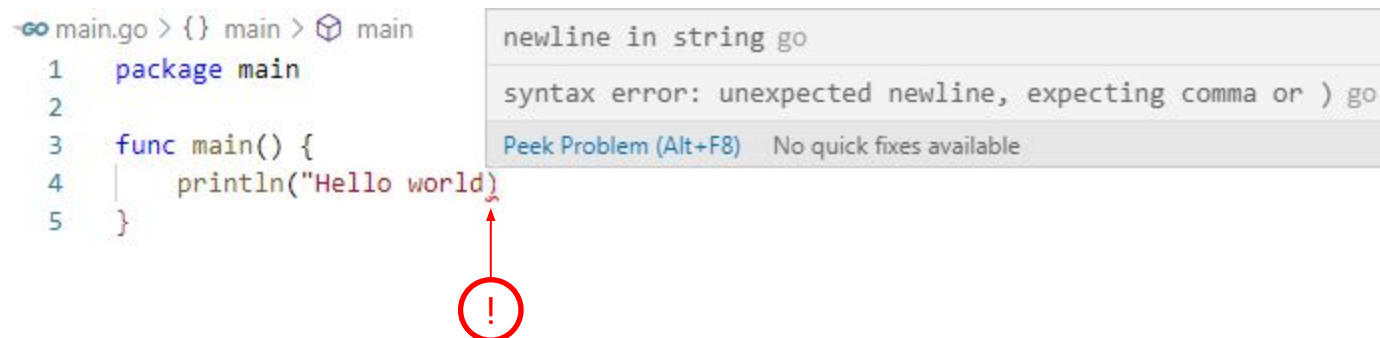


После этого во вкладке Debug Console вы увидите результат выполнения программы:



Errors

Обратите внимание, если вы напишете что-то неправильно (забудете поставить закрывающие кавычки и т.д.), то VS Code подсветит вам ошибку:



В этом случае запускать код до исправления ошибки смысла нет.

Обратите внимание: все мелочи имеют значение - вид кавычек, регистр символов, пробелы, переносы строк и т.д.

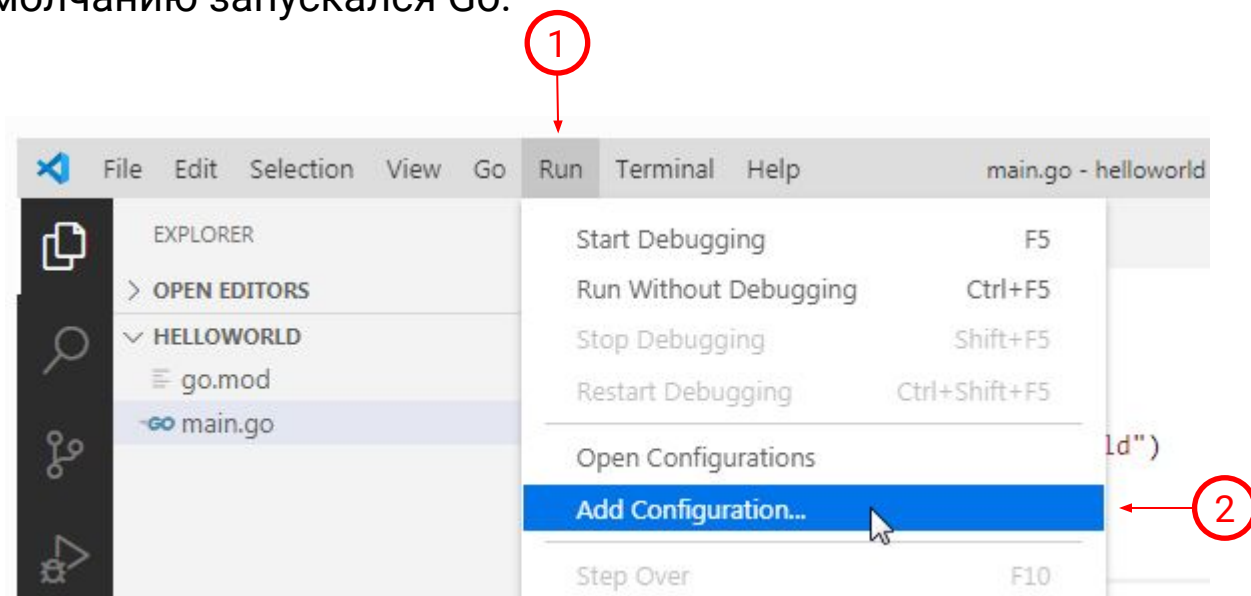


КОНФИГУРАЦИЯ ЗАПУСКА



Конфигурация запуска

Каждый раз выбирать Go из выпадающего списка не особо удобно, поэтому мы можем настроить конфигурацию запуска: сообщить VS Code, что мы хотим, чтобы по умолчанию запускался Go:



Конфигурация запуска



Это приведёт к тому, что у вас автоматически создастся каталог `.vscode` с файлом `launch.json` в нём. А дальнейшие нажатия `Ctrl + F5` будут приводить к автоматическому запуску Go.



Конфигурация запуска

Вот так это будет выглядеть (вам файл создавать не нужно - он создастся автоматически):

.vscode > {} launch.json > ...

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Launch",
9              "type": "go",
10             "request": "launch",
11             "mode": "auto",
12             "program": "${fileDirname}",
13             "env": {},
14             "args": []
15         }
16     ]
17 }
```

Переключитесь снова на вкладку [main.go](#) (нужно кликнуть на ней) и продолжим.



ПЕРВОЕ ПРИЛОЖЕНИЕ






Конвертер валют

Hello world, это, конечно, хорошо, но слишком примитивно. Давайте попробуем написать что-то приближенное к реальности.

На сайте <https://alif.tj> есть конвертер валют:

Курс валют 13 августа 2020 в 08:03

Валюта	Курс НБТ	Покупка	Продажа
 USD Доллар США	10.3128	10.3150	10.3200
 EUR Евро	12.1382	12.1000	12.6500
 RUB Российский рубль	0.1403	0.1420	0.1470

Конвертер валют

100 USD ▾

= 1031.50 TJS ▾



Конвертер валют

Соответственно пойдём по нашей схеме:

1. Создадим новый проект converter
2. Откроем его в VS Code
3. Инициализируем модуль с помощью команды `go mod init converter`
4. Создадим файл `main.go`
5. Введём базовую заготовку `pkgm + Tab`:

```
go main.go > {} main > main
1  package main
2
3  func main() {
4
5  }
```



Вычисления

Мы уже знаем, что функция `println` выводит всё то, что записано в круглых скобках.

Почему бы туда сразу не записать выражение для конвертации?

```
go main.go > ...  
1 package main  
2  
3 func main() {  
4     println(100.0 * 10.3150)  
5 }
```

числа записываются без кавычек
разделитель - . (точка)
операция умножения - * (звёздочка)

При запуске мы получим следующий результат:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
+1.031500e+003
```

Что это значит, ведь должно было получиться 1031.50?



Вычисления

Такая форма записи (**+1.031500e+003**) называется Scientific Notation. Буквально она значит следующее: **1.031500 * 10⁺⁰⁰³**, что как раз и будет **1031.50**.

Теперь давайте подумаем, что не так с нашей программой?

go main.go > ...

```
1 package main
2
3 func main() {
4     println(100.0 * 10.3150)
5 }
```



Вычисления

Сможем ли мы сказать через неделю, месяц, что значило каждое число?

go main.go > ...

```
1 package main
2
3 func main() {
4     println(100.0 * 10.3150)
5 }
```

Вполне возможно, что нет. Это примерно то же самое, что на сайте alif.tj виджет бы выглядел вот так:

Курс валют 13 августа 2020 в 08:03

10.3150

12.1000

0.1420

100




= 1031.50

Согласитесь, что понятного тут мало.



Вычисления

Курс валют 13 августа 2020 в 08:03

Валюта	Курс НБТ	Покупка	Продажа
 USD Доллар США	10.3128	10.3150	10.3200
 EUR Евро	12.1382	12.1000	12.6500
 RUB Российский рубль	0.1403	0.1420	0.1470

Конвертер валют

100 USD ▾

= 1031.50 TJS ▾

VS

Курс валют 13 августа 2020 в 08:03

10.3150

12.1000

0.1420

100

= 1031.50



Вычисления

Такие значения, жёстко записанные в коде (hardcoded), предназначение которых не понятно, называют магическими значениями (magic values):

```
main.go > ...  
1  package main  
2  
3  func main() {  
4      println(100.0 * 10.3150)  
5  }
```

Это считается плохим стилем программирования и подобные работы будут отправляться на доработку.



Пользователи

Пользователем веб-сайта банка являются клиенты, которые затем пользуются услугами банка и на этом банк зарабатывает, получая возможность платить зарплату своим сотрудникам и развиваться.

Пользователем же написанного вами кода будете являться вы и другие программисты, которые будут работать с вами в команде. От того, насколько понятным будет код, будет зависеть как быстро вы сможете понять программу и доработать её (а также исправить в ней ошибки).



Переменные

Чтобы нам было понятно, что это за числа, мы можем дать им имена.

Переменные - это имена, с привязанными к ним значениями. Переменные нужны только нам (Go они совсем не нужны).

Переменные в Go записываются вот в таком виде: `<имя> := <значение>`

значение переменной



Например: `amount := 100.0`

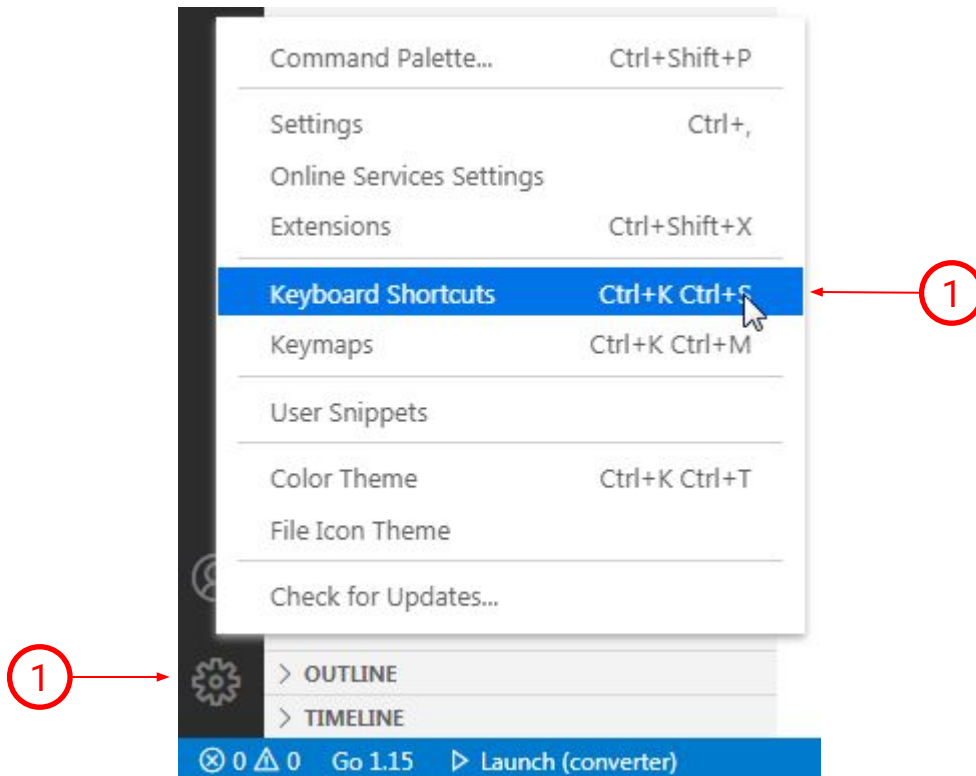


имя переменной



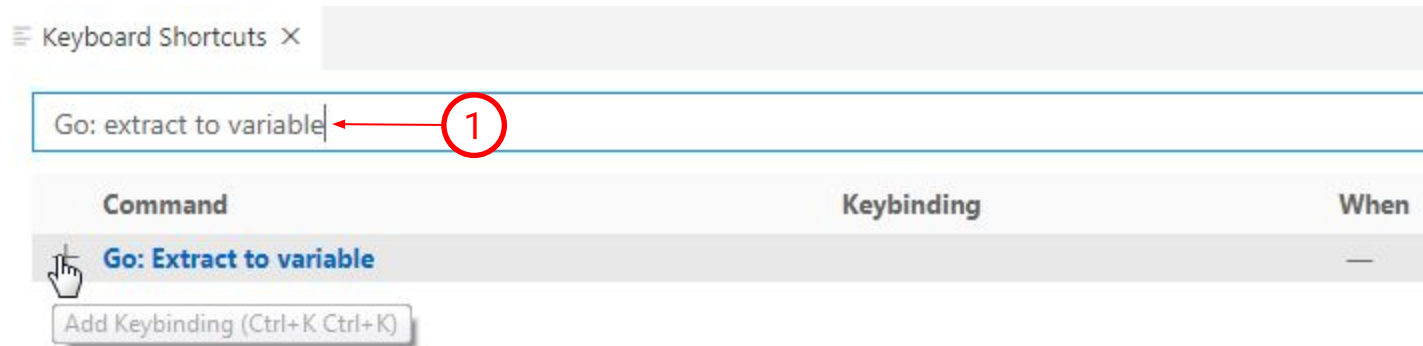
Рефакторинг

Давайте исправим наше приложение. Для удобства мы создадим собственные клавиатурные сокращения:



Рефакторинг

На открывшейся странице необходимо ввести Go: extract to variable и нажать на +:



В открывшемся окне нажмите клавиши **Ctrl + Alt + v**, после чего нажмите **Enter**:



Рефакторинг

После этого выделяем конкретное число и нажимаем **Ctrl + Alt + v**:

```
main.go x amount
main.go > {} main > m
1 package main
2
3 func main() {
4 | println(100.0 * 10.3150)
5 | }
Press 'Enter' to confirm your input or 'Escape' to cancel
```

Получим следующий код:

```
main.go > ...
1 package main
2
3 func main() {
4 | amount := 100.0
5 | println(amount * 10.3150)
6 | }
```



Рефакторинг

Прделаем то же самое со вторым числом:

```
main.go > ...  
1 package main  
2  
3 func main() {  
4     amount := 100.0  
5     buyRate := 10.3150  
6     println(amount * buyRate)  
7 }
```

А потом и с выражением `amount * buyRate`:

```
main.go > ...  
1 package main  
2  
3 func main() {  
4     amount := 100.0  
5     buyRate := 10.3150  
6     result := amount * buyRate  
7     println(result)  
8 }
```

После чего нужно запустить приложение и убедиться, что оно работает.



Рефакторинг

Как это работает? Go сам заменяет везде имена переменных на значения, которые в них содержатся (т.е. вместо `amount` везде будет подставлено `100.0` и т.д.):

```
go main.go > ...  
1  package main  
2  
3  func main() {  
4      amount := 100.0  
5      buyRate := 10.3150  
6      result := amount * buyRate  
7      println(result)  
8  }
```



DEBUGGER



Debugger

Для того, чтобы увидеть, как Go исполняет наш файл (и исполняет ли вообще), есть специальный инструмент, который называется Debugger (отладчик).

Отладчик – это специальный инструмент, который позволяет перевести Go в режим пошагового выполнения. При этом мы можем смотреть, что и как выполняется.



Debugger

Открываем файл `main.go` и на в области нумерации строк кликаем левой кнопкой мыши (либо клавиша **F9**) - поставится точка остановки (breakpoint):

```
go main.go > ...  
1  package main  
2  
3  func main() {  
4      amount := 100.0  
5      buyRate := 10.3150  
6      result := amount * buyRate  
7      println(result)  
8  }
```


Точка остановки – это строка, на которой остановится выполнение. Для того, чтобы её активировать, нужно запустить приложение в режим отладки (**F5**).



Debugger

Мы увидим строку, подсвеченную жёлтым - это значит, что эту строку Go ещё не исполнил:

```
~GO main.go > {} main > main
1  package main
2
3  func main() {
4  amount := 100.0
5  buyRate := 10.3150
6  result := amount * buyRate
7  println(result)
8 }
```

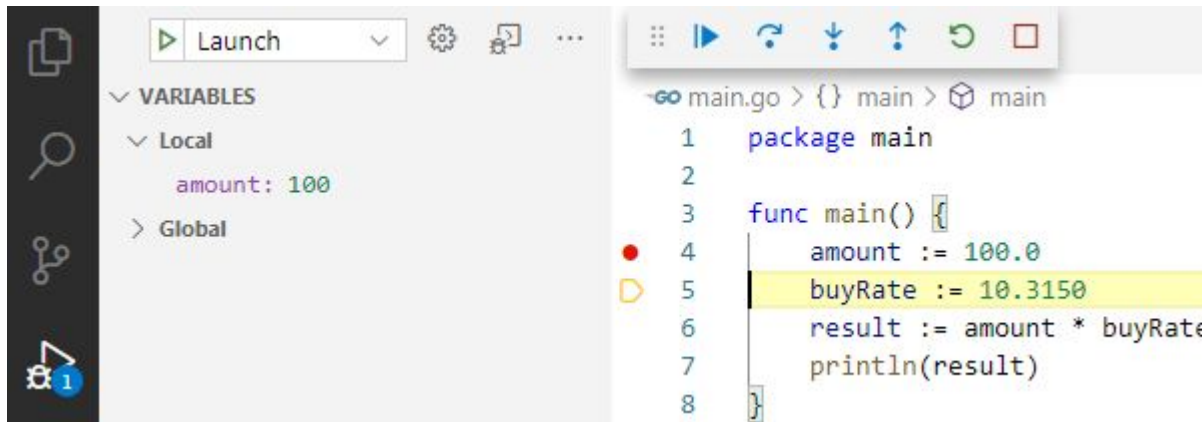
Чтобы перейти на следующую строку, нужно нажать на кнопку  или F10:

```
⋮ ▶ ↺ ⚡ ⬆ ⬆ ⬆ ⬆
~GO main.go > {} main > main
1  package main
2
3  func main() {
4  amount := 100.0
5  buyRate := 10.3150
6  result := amount * buyRate
7  println(result)
8 }
```



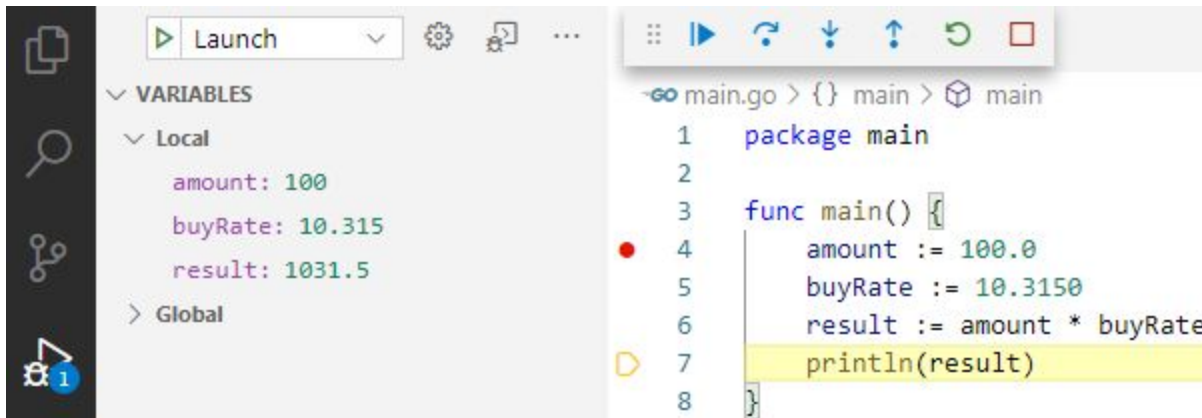
Debugger

После перехода на вторую строку, в панельке Variables появится информация о том, что создана переменная с конкретным именем и значением (и после этого она становится доступной для использования):




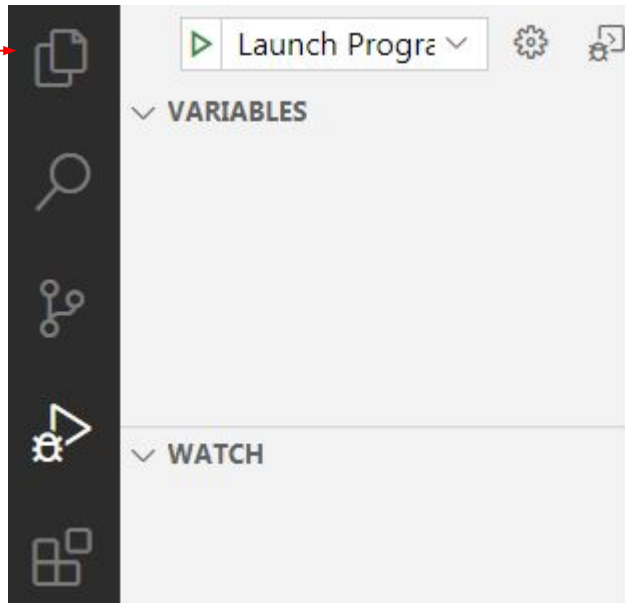
Debugger

Прошагав так всю программу мы увидим, как последовательно (сверху вниз) она исполняется и вычисляются значения переменных:



Debugger

После того, как вы дошли до последней строки, нужно нажать на кнопку  или **F5**, чтобы не проваливаться в код Go.



Чтобы обратно переключиться в режим отображения файлов, просто выберите в боковой панели файловый менеджер

Таким образом, Debugger позволяет нам прошагать всю программу, и понять, как она выполняется на самом деле.



ИТОГИ



Итоги

В этой лекции мы обсудили достаточно много важных моментов:

1. Установку инструментов
2. Работу в VS Code

В следующих лекциях мы будем опираться на то, что вы уже изучили в этой лекции (и не будем детально описывать процессы создания файлов, запуска приложений, переход в режим отладчика и т.д.).



ДОМАШНЕЕ ЗАДАНИЕ



ДЗ №1: Hello Go

Создайте проект аналогично тому, как мы это делали на лекции (включая инициализацию модуля, создание конфигурации запуска и т.д.). В [main.go](#) в функции `main` разместите код: `println("Hello, Go!")`.

Проект должен располагаться в каталоге [docs](#) (именно его и нужно заархивировать).



Как сдавать ДЗ

Вам нужно запаковать в zip-архив ваш каталог с проектом (не содержимое каталога, а сам каталог) - выделяете его и выбираете Отправить → Сжатая ZIP-папка:



Полученный архив загружаете в личном кабинете пользователя.




Важно: учитывается только последняя отправленная попытка.



ДЗ №2: Converter

На базе примера с лекции создайте приложение, которое будет конвертировать TJS в RUB по формуле со скриншота:

Курс валют 13 августа 2020 в 08:03

Валюта	Курс НБТ	Покупка	Продажа
 USD Доллар США	10.3128	10.3150	10.3200
 EUR Евро	12.1382	12.1000	12.6500
 RUB Российский рубль	0.1403	0.1420	0.1470

Конвертер валют

100 TJS ▾

= 680.27 RUB ▾

Проект должен располагаться в каталоге [converter](#) (именно его и нужно заархивировать).



ДЗ №2: Converter

Для этого:

1. В переменную `amount` сохраните значение `100.0`
2. В переменную `sellRate` сохраните значение коэффициента продажи
3. Результат вычислений сохраните в переменную `result`
4. С помощью инструкции `println` выведите значение переменной `result` на экран



Спасибо за внимание

alif academy

2020

