

# Опанування основами Go: Практичний посібник з освоєння мови Go

---

## 1.1 Вступ до мови Go

У цьому розділі ми покриємо такі теми:

1.1.1 [Огляд мови Go](#)

1.1.2 [Особливості та переваги Go](#)

1.1.3 [Приклади використання та застосування](#)

1.1.1 Огляд мови Go



Go, відомий також як Golang, це відкрита програмна мова, розроблена Google у 2007 році, яка була відкрито випущена у 2009 році. Go був створений Робертом Грисемером, Робом Пайком та Кеном Томпсоном, які прагнули створити мову, яка б комбінувала простоту та ефективність Python та C. Go — це статично типізована, компільована мова з акцентом на простоту, ефективність та паралельне виконання. Вона розроблена так, що бути простою для вивчення, підтримки та масштабування, що робить її відмінним вибором для сучасної розробки програмного забезпечення.

### 1.1.2 Особливості та переваги Go

Деякі з найвидатніших особливостей та переваг Go:

- Простий та чистий синтаксис: Go має прямий синтаксис, який легко читати та розуміти, що робить його ідеальним для нових програмістів та досвідчених розробників.
- Сильна типізація: Go є статично типізованою мовою, що означає, що перевірка типів виконується на етапі компіляції, допомагаючи виявляти помилки перед виконанням вашого коду.

- Підтримка паралелізму: Go має вбудовану підтримку паралельного програмування з допомогою `goroutines` та каналів, що дозволяє ефективно обробляти паралельні завдання та підвищує продуктивність.
- Збірник сміття: Go автоматично керує розміщення в пам'яті та звільнення з її допомогою збірника сміття, який допомагає запобігти витокам пам'яті та зменшує навантаження на розробників.
- Швидка компіляція: Go розроблено для швидкого часу компіляції, що сприяє швидкій розробці та ітерації.
- Кросплатформенна сумісність: Go підтримує декілька платформ, включаючи Windows, macOS, Linux та різні Unix системи, що робить його надзвичайно універсальним.
- Стандартна бібліотека: Широка стандартна бібліотека Go забезпечує багатий функціонал для загальних завдань, таких як I/O файлів, мережева робота та багато іншого, без потреби в зовнішніх залежностях.
- Ростуча спільнота: Go має активно зростаючу спільноту розробників, що означає наявність численних ресурсів, бібліотек та фреймворків, доступних для допомоги вам у розробці ваших додатків.

### 1.1.3 Приклади використання та застосування

Go є чудовим вибором для різних задач розробки програмного забезпечення, включаючи:

- Веб-сервери та веб-додатки: Надійна стандартна бібліотека Go та ефективна модель паралелізму роблять його ідеальною мовою для розробки веб-серверів, RESTful API та веб-додатків.
- Мережеві та розподілені системи: Мережеві пакети Go та підтримка паралелізму роблять його підходящим для створення розподілених систем, мережевих інструментів та інших додатків, орієнтованих на комунікації.
- CLI: Швидкий час компіляції Go і кросплатформенна сумісність роблять його чудовим вибором для створення інструментів CLI та утиліт.
- Мікросервіси: Простота, ефективність та масштабованість Go робить його підходящим для розробки мікросервісів, які можна легко розгортати та масштабувати незалежно.
- Додатки на основі хмари: Go стає все більш популярним для створення хмарно-орієнтованих застосунків, з орієнтацією на простоту, ефективність та продуктивність.

### Очікувані покращення та зміни

Go дуже дінамічно розвивається. Але має "обіцянку зворотньої сумісності".

- Generics з'явилися у версії 1.18, були значно покращені у версіях 1.19.x
- Go 1.20 додала 4 зміни до сінтаксису
- Go 1.21 додала нові вбудовані функції, покращені в рантаймі, `toolchain` підтриму (це можливість вибору версії для компіляції коду незалежно від встановленої локально версії)

Найближчим часом у версії 1.22 планується:

- Оптимізація роботи зі змінними у циклі `for`
- Новий ітератор з діапазонами для цілих чисел
- Покращення для інструментарія (`go get` `go mod` `go test`)

У наступному розділі ми проведемо вас через процес встановлення Go і налаштування вашого робочого простору, щоб розпочати вашу подорож з цією потужною програмною мовою.

## Встановлення Go та налаштування вашого робочого простору



Добре, давайте наженемося на налаштування вашого середовища Go! Слідуйте цим простим крокам, і ви будете готові до початку кодування за мить. Пам'ятайте, ви завжди можете звернутися до неймовірної спільноти Go, якщо вам потрібна допомога.

### 1.2 Завантаження та встановлення Go

У цьому розділі ми покриємо такі теми:

- 1.2.1 Загальний спосіб
- 1.2.1 Використовуючи Homebrew
- 1.2.2 Налаштування вашого середовища

- 1.2.3 Створення вашого робочого простору

## 1.2.1 Загальний спосіб

По-перше, давайте встановимо Go на ваш комп'ютер. Перейдіть на офіційний веб-сайт Go (<https://golang.org/dl/>) та завантажте інсталятор для вашої операційної системи (Windows, macOS або Linux).

Після того, як ви отримали інсталятор, слідуйте цим крокам, специфічним для платформи:

- Windows: Запустіть файл .msi та слідуйте інструкціям на екрані. Це просто!
- macOS: Відкрийте файл .pkg і пройдіть керований процес. Просто та легко!
- Linux: Розпакуйте архів .tar.gz та слідуйте інструкціям у доданому файлі README. Ви справитеся!

Якщо ви використали цей варіант, вам просто потрібно перейти до кроku [1.2.2 Налаштування Вашого середовища](#)

## 1.2.1 Використовуючи Homebrew

### macOS

Щоб встановити Go за допомогою **Homebrew** на системі **macOS**, дотримуйтесь цих кроків:

Відкрийте вікно Терміналу.

Встановіть Homebrew, виконавши наступну команду:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Як тільки Homebrew встановлено, виконайте наступну команду, щоб встановити Go:

```
brew install go
```

Переконайтесь, що Go встановлено правильно, виконавши наступну команду:

```
go version
```

Ви повинні побачити щось на зразок цього:

```
go version go1.21.5 darwin/amd64
```

Це повинно вивести версію Go, яку ви щойно встановили. Ось і все! Тепер у вас встановлено Go на вашій системі macOS за допомогою Homebrew.

## Linux

Щоб встановити Go за допомогою Homebrew на Ubuntu, вам потрібно виконати наведені нижче кроки:

Встановіть Homebrew на Ubuntu, виконавши наступну команду в терміналі:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Після встановлення Homebrew, оновіть свою інсталяцію Homebrew та формули, виконавши наступну команду:

```
brew update
```

Встановіть Go за допомогою Homebrew, виконавши наступну команду:

```
brew install go
```

Переконайтесь, що Go встановлено правильно, виконавши наступну команду:

```
go version
```

Це повинно вивести версію Go, яку ви щойно встановили. Ось і все! Тепер у вас встановлено Go на вашій системі **Linux** за допомогою Homebrew.

## Windows

Щоб встановити Go на Windows, ви можете слідувати цим крокам:

1. Завантажте інсталятор Go для Windows з офіційного веб-сайту Go (<https://golang.org/dl/>).
2. Запустіть виконавчий файл інсталятора (.msi файл) та слідуйте вказівкам, щоб встановити Go.
3. Після завершення встановлення відкрийте командний рядок Windows (cmd) або PowerShell.
4. Перевірте, чи Go був правильно встановлений, введіть наступну команду:

```
go version
```

Якщо Go було встановлено правильно, номер версії встановленого Go відобразиться в командному рядку.

## 5. Ви готові почати використовувати Go на Windows!

Примітка: Переконайтесь, що директорія, в якій встановлено Go, додана до змінної середовища PATH вашої системи, щоб Ви могли виконувати команди Go з будь-якого місця в командному рядку.

Оскільки у більшості випадків ви будете використовувати Linux-оточення для запуску розроблених вами додатків, я б не рекомендував використовувати Windows спеціально для розробки.

### 1.2.2 Налаштування Вашого Середовища

Далі давайте налаштуємо ваше середовище. Цей крок гарантує, що Go та його інструменти гарно працюватимуть на вашій системі.

Настройте змінну середовища **GOPATH**.

Ця змінна вказує Go, де розміщувати та зберігати ваші проекти (вони називаються "робочими просторами").

За замовчуванням, Go використовує директорію, названу "**go**", в папці вашого дому, але ви безперешкодно можете вибрати іншу, якщо бажаєте.

Ось як ви можете встановити **GOPATH** на різних платформах:

- Windows: додати нову змінну середовища користувача названу "**GOPATH**" зі значенням "**%USERPROFILE%\go**" (або вашим бажаним шляхом).
- macOS та Linux: додати наступний рядок до файлу конфігурації вашої оболонки (наприклад, `~/.bashrc` або `~/.zshrc`): `export GOPATH=$HOME/go` (або вашим бажаним шляхом).

Додайте бінарний файл Go до **PATH** вашої системи.

Цей крок гарантує, що ви зможете запустити команди Go з будь-якого вікна термінала.

Ви також захочете додати наступні шляхи:

- Windows: додайте "**%USERPROFILE%\go\bin**" та "**C:\Go\bin**" до змінної **PATH** вашої системи. Це дозволить запускати команди встановлені за допомогою `go install <path-to-repo>`

Збережіть зміни та відкрийте нове вікно термінала, щоб переконатись, що команда Go доступна.

Перевірте це, запустивши `go version`.

Якщо все налаштовано правильно, ви побачите встановлену версію Go.

### 1.2.3 Створення робочого простору

Якщо встановлення було зроблено за допомогою інсталяції (`home brew`, наприклад) додаткові дії не потрібні, це буде зроблено за вас автоматично

З встановленою Go та налаштованим середовищем настав час створити свій робочий простір. Тут будуть знаходитися всі ваші проекти Go.

Створіть директорію для свого робочого простору (якщо ви ще цього не зробили) за шляхом, указаним у вашій змінній середовища **GOPATH** (це не обов'язково для проектів, які не публікуються як пакети).

Усередині вашого робочого простору створіть три піддиректорії: "src", "pkg" та "bin". Тут будуть зберігатися ваші вихідні коди, об'єкти пакетів та скомпільовані бінарники відповідно.

І ось і все! Ви повністю готові до розробки чудових додатків Go. У наступному розділі ми напишемо просту програму "Hello, World!", щоб розпочати ваш путь з Go.

*Залишайтесь з нами та насолоджуйтесь кодуванням!*

## 1.3 Привіт, Світ! - Ваша перша програма на Go

Ура, ваше середовище Go встановлено! Тепер давайте зануримось в кодування та напишемо вашу першу програму на Go - класичний "Привіт, Світ!". У цьому розділі:

- [1.3.1 Створення вашого первого файла Go](#)
- [1.3.2 Написання програми "Привіт, Світ!"](#)
- [1.3.3 Запуск програми "Привіт, Світ!"](#)
- [1.3.4 Введення модулів Go](#)

### 1.3.1 Створення вашего первого файла Go

У вашому терміналі перейдіть до папки "src" у вашому робочому просторі (папка, яку ви створили раніше).

Створіть нову папку з назвою "hello" для вашого проекту "Привіт, Світ!": `mkdir hello`.

Перейдіть до папки "hello": `cd hello`.

Створіть новий файл під назвою "main.go", використовуючи ваш улюблений текстовий редактор або IDE.

### 1.3.2 Написання програми "Привіт, Світ!"

Тепер прийшов час написати трохи коду! У вашому файлі "main.go" введіть наступне:

```
package main

import "fmt"

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

Давайте розглянемо, що відбувається тут:

- package main: Цей рядок говорить Go, що цей файл є частиною пакета "main". Кожен виконуваний файл Go потребує пакет "main" з функцією "main".
- import "fmt": Тут ми імпортуємо пакет "fmt", який надає функції для форматованого I/O (наприклад, друк в консолі).
- func main() { ... }: Це функція "main", яка служить входом для нашої програми. Коли програма виконується, вона почне виконувати код всередині цієї функції.
- fmt.Println("Hello, World!"): Цей рядок друкує рядок "Hello, World!" в консолі.

### 1.3.3 Створення та запуск вашої першої програми Go

З написаним кодом "Hello, World!", давайте зіберемо та запустимо програму:

- У терміналі переконайтесь, що ви все ще в папці "hello" (де розташований файл "main.go").
- Зберіть програму, виконавши команду: go build main.go. Це скомпілює ваш код Go в виконуваний двійковий файл під назвою "hello" (або "hello.exe" на Windows).
- Запустіть скомпільований двійковий файл, введіть ./hello (або .\hello.exe на Windows) та натисніть Enter.

Проте, якщо ви запустите команду go build, це призведе до помилки

```
go: go.mod file not found in current directory or any parent directory; see  
'go help modules'
```

Це тому, що Go за замовчуванням вимагає оголошення модуля. Щоб вирішити цю проблему, нам потрібно додати файл go.mod до нашого проекту.

### 1.3.4 Знайомство з модулями Go

Модулі Go є офіційним рішенням для керування залежностями в Go, впровадженими в Go 1.11. Вони допомагають вам керувати залежностями вашого проекту, відстежуючи необхідні пакети та їхні конкретні версії в простому, зручному для читання файлі. Це робить ваші проекти більш відтворюваними та легкими для обміну з іншими.

#### 1.3.4.1 Ініціалізація модуля Go

Щоб створити модуль Go для вашого проекту "Hello, World!", слідуйте цим крокам: У вашому терміналі переконайтесь, що ви знаходитесь в папці "hello" (де розташований ваш файл "main.go").

Виконайте наступну команду для ініціалізації нового модуля Go:

```
go mod init hello
```

Якщо ви плануєте створити публічний або приватний репозиторій з пакетом на GitHub, вам потрібно використовувати інший формат для назви модуля.

Виконайте наступну команду для ініціалізації нового модуля Go:

```
go mod init github.com/<username>/hello
```

Замініть на ваше ім'я користувача GitHub або будь-який інший унікальний ідентифікатор.

Це створює новий файл під назвою "go.mod" в папці вашого проекту. Файл "go.mod" буде містити назву модуля та версію Go.

#### 1.3.4.2 Додавання залежностей до вашого модуля Go

Коли ви імпортуєте пакети, які не є частиною стандартної бібліотеки, модулі Go автоматично додають їх до вашого файла "go.mod", разом з їхніми відповідними версіями. Припустимо, ви хочете імпортувати популярний сторонній пакет "gorilla/mux" для роутингу в веб-застосуванні. Ось як ви це зробите: Додайте оператор імпорту для "gorilla/mux" до вашого файла "main.go":

```
import (
    "fmt"
    _ "github.com/gorilla/mux"
)
```

У вашому терміналі перейдіть до папки "hello" та виконайте наступну команду:

```
go mod tidy
```

Ця команда оновить ваш файл "go.mod", додаючи залежність "gorilla/mux" та її версію. Крім того, буде згенеровано файл "go.sum", який допомагає забезпечувати цілісність вашої залежності, зберігаючи їх контрольні суми.

Ось і все! Тепер ви знаєте основи модулів Go та як керувати залежностями у ваших проектах Go. З ростом ваших проектів ви усвідомите, що модулі Go є незамінним інструментом для відстеження ваших залежностей та роботи вашого коду більш модульним та підтримуваним.

Маючи ці нові знання, ви готові перейти до більш складних тем Go. У наступній главі ми дослідимо типи даних, змінні та константи в Go.

Продовжуйте свій відмінний прогрес, і продовжимо навчання разом!