


## Th.S LÊ NGỌC THẠCH

### Lời nhắn

eBook "**Chạm tới GO trong 10 ngày**" này dự kiến phát hành vào tháng 12/2021. Bạn có thể đặt hàng ngay bây giờ với ưu đãi giảm 50% chỉ **199K**, tiết kiệm 200K. Thanh toán nhanh theo 2 cách:

① MoMo	② Chuyển khoản
<p>☎ 0908550642      👤 Lê Ngọc Thạch</p> <p>📄 Nội dung tin nhắn: GO2021 email sdt Ví dụ: abc@gmail.com 0908550642 <b>Email và sdt của người nhận eBook.</b></p> <p>Trường hợp tặng bạn bè thì ghi thông tin email và sdt của bạn.</p> <p>Quét mã QR thanh toán 199K.</p> 	<p>Lê Ngọc Thạch, Ngân Hàng Tiên Phong, CN HCM Số tài khoản: 00002888001 Nội dung tin nhắn: GO2021 email sdt Vd tin nhắn: GO2021 abc@gmail.com 0908456321 Quét mã QR để thanh toán cho:</p>  <p>Quét mã vạch này để giao dịch</p>

Ngoài ra, bạn có thể đọc ngay bản nháp hiện tại với giá 0đ theo cách sau:

Cài **App MinePI** cho điện thoại tại theo link:

<https://minepi.com/thachln>

Sử dụng invitation code: **thachln**

Liên lạc với tác giả qua <https://facebook.com/ThachLN> để cung cấp account MinePI, SĐT và Email nhận nhận eBook với thông tin mã hóa đính kèm.

Lê Ngọc Thạch

# CHẠM TỚI GO TRONG 10 NGÀY

## Mục lục

Mục lục .....	2
Quy ước.....	10
Thử thách.....	13
Thử thách sau ngày 1 .....	13
Thử thách sau ngày 4 .....	14
Thử thách sau ngày 5 .....	15
Thử thách sau ngày 6 .....	16
Thử thách sau ngày 7 .....	20
Thử thách sau ngày 8 .....	20
Thử thách sau ngày 9 .....	20
Thử thách sau ngày 10 .....	21
Final Project #1 .....	21
Final Project #2.....	25
Ôn tập kiến thức cơ bản về máy tính và phần mềm.....	31
Bài 1: Quá trình tiến hóa của các mô hình phần mềm.....	31
Phần mềm trên máy cá nhân.....	32
Máy vi tính cá nhân (personal computer).....	32
Giao diện console .....	33
Giao diện đồ họa (GUI – Graphics User Interface) .....	35
Phần mềm trên mạng nội bộ .....	37
Mạng nội bộ (LAN - Local Network).....	37
Phần mềm trên nền tảng mạng Internet.....	39
Mạng Internet.....	39
Phần mềm trên mạng Internet.....	39
Bài 2 – Cấu trúc của phần mềm.....	41

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Công thức I + P + O.....	42
Thu nhận thông tin.....	42
Xử lý thông tin .....	42
Xuất kết quả. ....	42
Ví dụ 1: .....	42
Ngày 1: Làm quen với GOLANG .....	43
Bài 1 – Tại sao GO ra đời .....	44
Bài 2: Ngôn ngữ lập trình GO .....	45
Biến (Variable), Cấu trúc (Structure).....	45
Variable có nghĩa là gì? .....	48
Khai báo biến (variable declaration) .....	49
Lệnh gán (assign).....	49
Bài 3 – Chuẩn bị môi trường lập trình.....	52
GO Core.....	52
Cài thêm thư viện .....	52
Visual Code .....	52
Cài GO trên Ubuntu .....	54
Bài 4 – Viết chương trình đơn giản với GO .....	55
Viết mã.....	55
Biên dịch.....	55
Chạy trực tiếp mã nguồn.....	57
Phép gán (assign).....	59
Các toán tử cơ bản.....	60
Hàm (function).....	61
Chạy chương trình có tham số dòng lệnh trong Visual Code .....	62
Lấy tham số từ dòng lệnh .....	62
Vòng lặp (loops) .....	63
Nâng cao .....	64
Bài 5 – Biểu diễn thông tin đơn giản với GO.....	65
Kiểu chuỗi (string).....	65
Kiểu dữ liệu số (Numeric data types) .....	66
Viết chương trình Fibonacci.....	71

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Mảng (arrays).....	72
Slice (chưa biết gọi tiếng Việt là gì) .....	75
Maps.....	81
Thời gian (Times & dates).....	82
Tra cứu định dạng.....	83
Bài tập.....	85
Ngày 2: Biểu diễn thông tin phức hợp.....	86
Bài 1 – Biểu diễn thông tin phức hợp với GO.....	87
Cấu trúc (Structure).....	87
Kết hợp Slice và Structure.....	87
Con trỏ (Pointer).....	88
Tuples (Bộ dữ liệu).....	90
Đọc thêm và thực hành.....	93
Chuỗi (String) .....	93
Regular expressions and pattern matching .....	94
Bài 2 – Viết hàm cho cấu trúc .....	96
Phân tích hàm calculateBMI cho struct Employee .....	96
Tổ chức thành thư viện (module) .....	97
Bài 3 – Dữ liệu dạng JSON .....	100
Đọc dữ liệu JSON .....	100
Ngày 3: Cấu trúc điều khiển.....	100
Bài 7 – Cấu trúc rẽ nhánh.....	101
Lệnh if.....	101
Switch.....	101
Bài 8 – Vòng lặp.....	104
Vòng lặp (loops).....	104
Vòng lặp for nâng cao.....	106
Ngày 4: Làm việc với dữ liệu trên đĩa cứng.....	110
Bài 1 – Làm việc với thư mục và file.....	111
Lấy thông tin về file/thư mục.....	111
Lấy nội dung thư mục .....	112
Lấy nội dung file.....	112

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Lưu file.....	113
Lưu và đọc file mã hóa .....	114
Bài 2 – Làm việc với file CSV .....	116
Bài 3 – Đọc file CSV .....	118
Bài 4 – Ghi file CSV .....	120
Bài 5 – Đọc file vào cấu trúc (struct).....	121
Cài đặt thư viện.....	121
Đọc đoạn dữ liệu binary vào mảng các struct.....	121
Đọc file CSV vào mảng các struct .....	122
Bài 6 – Đọc file văn bản.....	126
Bài 7 – Đọc file Excel .....	128
Cài đặt .....	128
Đọc file Excel.....	128
Ghi dữ liệu ra file Excel.....	129
Ngày 5: Tổ chức dự án GOLANG .....	130
Bài 1 – Tổ chức mã nguồn.....	131
Bước 1: Tạo file go.mod để mô tả tên của module .....	131
Bước 2: Tạo thư mục và file chứa hàm dùng chung.....	132
Bước 3: Viết chương trình chính .....	132
Bài 2 – Tinh chỉnh mã nguồn .....	139
Phiên bản 0.0.2.....	139
Phiên bản 0.0.3.....	140
Phiên bản 0.0.4.....	141
Bài 3 – Tập thói quen viết phần mềm.....	143
Sử dụng logging .....	143
Bài 4 : Hàm trả về không phải là giá trị .....	146
Bài 3 – Go Packages và Functions.....	147
Anonymous function.....	147
Do it yourself:.....	148
Ngày 6: Sử dụng cơ sở dữ liệu PostgreSQL .....	148
Bài 1 – Làm quen với CSDL .....	149
Bài 2 – Sử dụng PostgreSQL portable .....	150

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Tải gói binary .....	150
Tạo file khởi động PostgreSQL server .....	151
Khởi động PostgreSQL server .....	152
Tương tác với PostgreSQL Server qua dòng lệnh.....	152
Tương tác với PostgreSQL Server qua web site.....	156
Tương tác với PostgreSQL Server qua web site.....	159
Bài 3 – Thực hành với PostgreSQL .....	160
Tạo một CSDL “ECP” .....	160
Nhập dữ liệu .....	165
Bài 4 – GOLANG và PGSQL .....	167
Cài thư viện .....	167
Ví dụ.....	167
Giải thích code từ ví dụ.....	168
Bài 5 – Sử dụng file cấu hình .....	171
Cài thư viện viper .....	171
Ngày 7: Lập trình đồng thời và song song với GO .....	171
Bài 1 – Khái niệm Concurrency và Parallelism.....	172
Tạo goroutine .....	172
Đợi hàm Goroutine chạy xong.....	173
Sử dụng channel cho goroutine.....	174
Ví dụ 2.....	176
Bài 2 – Khái niệm Concurrency và Parallelism.....	177
Bài 3 – Lập trình Concurrency .....	178
Bài 4: Lập trình Parallelism.....	179
Ngày 8: GOLANG và C/C++ .....	179
Bài 1 - Lập trình C trong GO.....	180
Ngày 9: Các chủ đề mở rộng/nâng cao .....	180
Bài 1 – Viết hàm với tham số linh động .....	181
Variadic functions .....	181
Bài 2 - Crawl dữ liệu với GOLANG .....	183
Request đơn giản .....	183
Thiết lập timeout cho request.....	183

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Thiết lập header .....	184
Download URL .....	185
Use substring.....	185
Bài 3 - Lập trình CUDA với GOLANG.....	187
Bài 4 - Phát triển Web Application với Beego.....	188
Cài đặt GO .....	188
Cài đặt Beego .....	188
Tạo dự án .....	188
Chạy ứng dụng .....	189
Truy cập ứng dụng.....	190
Chỉnh sửa code.....	190
Tạo API .....	191
Bài 5 - Phát triển Web Backend với Gin-Gonic.....	192
Cài đặt .....	192
Viết Backend đơn giản.....	192
Triển khai lên server Ubuntu với Nginx .....	194
Nâng cao .....	194
Cài đặt các thư viện hỗ trợ web .....	195
Bài 6 - Sử dụng GOLANG trong WSL2 .....	197
Ngày 10: Tra cứu theo nhu cầu.....	197
Các API về xử lý chuỗi.....	198
Các API sử dụng GIN GO NIC .....	199
Ngày 11: Testing với GO .....	199
Biên dịch OpenCV từ mã nguồn.....	200
Cài đặt Anaconda: .....	200
Cài đặt mkl-service.....	200
Kiểm tra thông tin thiết bị GPU.....	200
Biên dịch.....	201
Ngày 12 - Blockchain .....	202
Bài 1: Ôn tập kiến thức cơ bản.....	203
Làm quen lại với kiểu Slice của byte .....	203
Thư viện bytes.....	204

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Thư viện mã hóa .....	204
Mã hóa base58 .....	205
Khóa công khai và khóa bí mật .....	206
Bài 2 – Tạo cấu trúc chuỗi khối .....	208
Tạo dự án .....	208
Viết mã nguồn main.go .....	208
Bài 3 – Minh họa thuật toán đồng thuận ProofOfWork.....	213
Định nghĩa bổ sung Block.....	213
Hàm tạo Block cũ .....	213
Hàm tạo Block cải tiến.....	213
Cài đặt ProofOfWork (PoW) .....	214
Bài 4 – Lưu trữ Blockchain .....	217
Sử dụng database dạng Key-Value .....	217
Đóng gói OpenSSL .....	219
Chuẩn bị công cụ .....	219
Cài đặt tool Visual Studio 2019.....	219
Clone mã nguồn dự án OmiseGo eWallet.....	220
Lập trình wxWidget .....	222
Phụ lục .....	222
Phụ lục 3 .....	222
Lập trình giao diện với goki .....	223
Cài đặt GCC for Windows .....	223
Cài đặt thư viện goki.....	224
Viết ứng dụng.....	224
Biên dịch và chạy ứng dụng.....	225
GOLANG và QT .....	226
Cài đặt phần mềm QT .....	226
Cài đặt thư viện.....	231
Trải nghiệm lập trình.....	232
Phụ lục 4 .....	234
GOLANG và Google Sheet.....	234



Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

## Quy ước

Một số nội dung trong tài liệu được trình bày với các định dạng khác nhau thì có ý nghĩa của nó, bạn đọc nên nắm thông tin này để tiện theo dõi.

## Mã nguồn

Mã lệnh được viết và đóng khung với font chữ **Consolas**, có thanh màu vàng bên trái; và kết quả hiển thị trên màn hình được đóng trong khung màu đỏ bên dưới như sau:

```
package main

import (
    "fmt"
)

func main() {
    name := "Thạch"
    fmt.Println("Hello ", name)
}

Hello Thạch
```

## Lệnh thực thi trong hệ điều hành

Trường hợp các lệnh thực thi trong môi trường hệ điều hành (phân biệt với các lệnh, hoặc mã nguồn của chương trình thực thi trong môi trường lập trình) thì dấu hiệu có 2 thành màu vàng như sau:

```
Hello.exe "I can do"
```

## Đường dẫn hiện hành

Đôi khi lệnh được hướng dẫn có cả tên ổ đĩa và thư mục và dấu mũi tên như bên dưới (phần chữ mờ). Phần này ý nói là chạy lệnh bên phải dấu mũi tên trong thư mục hiện hành D:\MyGo.

```
D:\MyGo> go build GoArgs.go
```

## Hệ điều hành Windows và Linux/Mac

Các bạn có thể học và làm việc với GOLANG bằng máy tính chạy hệ điều hành Windows, hoặc Linux hay Mac (gọi chung là Linux/Mac). Trong tài liệu khi mô tả các chương trình đã đóng gói vì dụ file Hello.exe thì bạn hiểu là dành cho người dùng Windows. Đối với các bạn dùng Linux/Mac thì tự hiểu là file Hello.

Đối với thư viện cũng vậy. Trên Windows thì tài liệu sẽ viết là là .dll (vd common.dll) thì các bạn dùng Linux/Mac tự hiểu là file common.o.

## Cặp dấu nháy

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Trong NNLT GO, dữ liệu **dạng kí tự** được bao đóng trong cặp **dấu nháy đơn**, dữ liệu **dạng chuỗi** được bao đóng trong **dấu nháy đôi**. Trên bàn phím máy tính thì dấu **nháy trái** và **phải** là giống nhau. Tuy nhiên trong phần mềm soạn thảo văn bản như Microsoft Word thì cặp dấu nháy đơn và đôi được thay thế bằng “, ” để tăng tính thẩm mỹ. Các dấu nháy thẩm mỹ này khác với kí tự ' và " trên bàn phím (phím bên trái phím Enter).

Đôi khi bạn copy & paste mã nguồn vào các phần mềm như Microsoft Word thì các dấu nháy có thể bị “trang trí” lại như trên. Vì vậy khi copy mã nguồn từ Microsoft vào các công cụ lập trình thì hãy thay thế lại cho đúng.

Một qui ước khác liên quan đến dấu nháy đôi là khi dùng trong văn bản để bao đóng danh từ riêng, hoặc lệnh như hướng dẫn sau: *Bạn hãy thử gõ lệnh “dir” trong cửa sổ TERMINAL để xem nội dung thư mục hiện hành.* Trong câu hướng dẫn này thì lệnh `dir` được gõ vào cửa sổ TERMINAL **KHÔNG** bao gồm cặp dấu nháy.

### Cách viết trình tự bấm chọn menu

Khi cần trình bày thứ tự các nút bấm, hoặc các mục cần bấm trong các thao tác thì sẽ dùng dấu lớn hơn >. Ví dụ khi hướng dẫn bạn sử dụng phần mềm Visual Code vào menu Run, bấm vào mục “Run Without Debugging” thì sẽ viết gọn như sau:

Vào menu Run > Run Without Debugging.

### Đường dẫn thư mục (Path)

Trong Windows thì dấu cách thư mục là dấu xuyệt trái (back slash). Ví dụ: `D:\ai2020\data`.

Tuy nhiên ngôn ngữ GO và phần mềm lập trình Visual Code được thiết kế tương thích với các hệ điều hành khác như Macintosh, Linux. Các hệ điều hành thì dùng dấu xuyệt phải (right slash) để phân cách thư mục. Ví dụ: `/mnt/d/MyGO`.

Vì vậy khi trình bày đường dẫn thư mục trong câu văn thì đôi lúc dùng \, hoặc đôi lúc dùng / do dữ liệu được minh họa trên Windows hoặc Linux/Mac.

Nhưng trong mã nguồn thì đều thống nhất là dùng dấu xuyệt phải / như:

```
read.csv("D:/MyGO/HelloGO.go")
```

### Các từ viết tắt, tiếng Anh thường xuyên được sử dụng trong sách

Viết tắt	Diễn giải
NNLT	Ngôn ngữ lập trình

### Cách viết dấu chấm câu, ghi chú hình, bảng

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Tài liệu này sẽ hạn chế tuân thủ cú pháp viết văn thông thường khi sử dụng dấu chấm cuối câu nhưng vẫn đảm bảo người đọc hiểu đúng. Ví dụ trong các đề mục được liệt kê thì đôi lúc không cần viết dấu chấm cho nhanh. Đặc biệt là trong trường hợp có tên file cuối câu.

Trong các hình, hoặc bảng biểu thì sẽ không đánh số. Thay vào đó tôi sẽ dùng những từ như: Hình dưới đây, hình bên dưới; hoặc hình phía trước để bạn đọc có thể hiểu chính xác mà không cần phải mất thêm thời gian ghi và đánh số thứ tự các hình, bảng biểu.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

## Thử thách

Dưới đây là danh sách các nhiệm vụ cơ bản của một lập trình viên GOLANG cần phải trải qua trước khi tham gia dự án.

### Thử thách sau ngày 1

#### *Chương trình 1*

Viết chương trình GO nhận danh sách các số (hợp lệ) từ tham số dòng lệnh (command-line argument), thực hiện tính toán và hiển thị ra màn hình các thông tin sau:

- Số nhỏ nhất, số lớn nhất (min, max)
- Giá trị trung bình (mean)
- Giá trị trung vị (median)

Ghi chú:

- Median gọi là trung vị. Đây chính là giá trị của phần tử ở chính giữa một dãy giá trị có xếp theo thứ tự. Trong trường hợp dãy có số phần tử là chẵn thì trung vị được tính là trung bình của 2 phần tử ở giữa của dãy có thứ tự. (<https://ThachLN.github.io>)
- Giả định dữ liệu truyền trên tham số dòng lệnh là hợp lệ, không cần viết mã kiểm tra.

#### *Chương trình 2*

Viết chương trình GO nhận một số nguyên từ tham số dòng lệnh. Số này có tối đa 3 chữ số. Chương trình hiển thị dạng văn bản của số đó. Ví dụ:

123 → “Một trăm hai mươi ba.”

(Văn bản không bao gồm cặp dấu nháy đôi)

Ghi chú:

- Giả định dữ liệu truyền trên tham số dòng lệnh là hợp lệ, không cần viết mã kiểm tra.

#### *Chương trình 3*

Viết chương trình sinh ra bảng dữ liệu ngẫu nhiên từ các tham số dòng lệnh có dạng như sau:

Số\_dòng   tên\_cột\_1   kiểu\_dữ\_liệu\_1   tên\_cột\_2   kiểu\_dữ\_liệu\_2 ...

Diễn giải:

- Số\_dòng: là một số nguyên cho biết số dòng dữ liệu cần tạo ra
- Cột dữ liệu thứ **nhất** có tên là **tên\_cột\_1** và có kiểu dữ liệu như **kiểu\_dữ\_liệu\_1**.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

- Cột dữ liệu thứ **hai** có tên là **tên\_cột\_2** và có kiểu dữ liệu như **kiểu\_dữ\_liệu\_2**.

- Tương tự cột dữ liệu thứ 3, thứ 4...thứ n nếu có các cặp tham số tiếp theo,

Ghi chú:

- Giả định dữ liệu truyền trên tham số dòng lệnh là hợp lệ, không cần viết mã kiểm tra.

## Thử thách sau ngày 4

### *Chương trình 4*

Viết chương trình đọc file csv với đường dẫn được truyền trên tham số dòng lệnh. File cvs gồm có các cột thông tin:

- Mã nhân viên: kiểu chuỗi 10 kí tự có định dạng ABC#####
  - o A: kí tự từ ‘A’ tới ‘Z’ cho biết ngạch nhân viên (C: điều hành; M: Quản lý; E: chuyên gia, N: nhân viên bình thường)
  - o B: kí tự từ ‘0’ tới ‘9’ cho biết cấp bậc (level) của nhân viên trong ngạch.
  - o C: kí tự từ ‘A’ tới ‘Z’ cho biết phòng ban hoặc chuyên môn chính của nhân viên (D: Ban giám đốc; K: Kế toán; S: Kinh doanh; ....)
- Tên (First Name): kiểu chuỗi
- Chữ lót (Middle Name): kiểu chuỗi
- Họ (Last Name): kiểu chuỗi
- Ngày tháng năm sinh: định dạng yy-mm-dd
- Hệ số năng lực: số từ 1 đến 9
- Hệ số lương: số thực
- Giới tính: chữ “Nam” hoặc “Nữ” hoặc “Khác”

Chương trình thực hiện các chức năng sau:

- Hiển thị ra màn hình các thông tin:
  - o Tổng số nhân viên
  - o Tổng số nhân viên theo giới tính:
    - Nam: ?

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

- Nữ: ?
- Khác ?
- Độ tuổi trung bình của toàn bộ nhân viên trong công ty (Năm hiện tại lấy từ thời gian của máy tính đang chạy chương trình).

### **Thử thách sau ngày 5**

#### *Chương trình 5*

Mở rộng chương trình/task 4 ở trên với các gợi ý sau:

- Tổ chức mã nguồn bằng cách sử dụng các hàm riêng, các nghiệp vụ được tổ chức trong lớp thư viện riêng để sau này đóng gói và cung cấp thư viện back-end cho nhóm làm giao diện. Cần các hàm với tham số truyền vào là danh sách nhân viên với cấu trúc đã được định nghĩa thích hợp cho bài toán. Trong đó sử dụng con trỏ (pointer) hoặc không sử dụng sau cho hợp lớp (tối ưu bộ nhớ, tốc độ truyền dữ liệu) với các chức năng sau:
  - Đọc file csv vào danh sách (bộ nhớ máy tính).
  - Thêm 1 nhân viên mới vào danh sách.
  - Xóa 1 nhân viên khỏi danh sách theo Mã nhân viên.
  - Cập nhật thông tin nhân viên trong danh sách.
  - Tìm kiếm nhân viên có mã số chứa một chuỗi ký tự cho trước, không phân biệt chữ hoa và chữ thường.
  - Lưu file csv ra thư mục được chỉ định.
  - Lấy danh sách nhân viên có ngày sinh nhật trong tuần tới (Tính từ ngày Thứ Hai kế tiếp đến Chủ Nhật kế tiếp. Nếu hôm nay là thứ Hai thì tính từ Hôm nay đến Chủ Nhật).
- Chương trình hỗ trợ file CSV có lưu tên người nước ngoài (dùng mã Unicode hợp lý).

[Còn tiếp]

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

**Thử thách sau ngày 6**



Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

### 1) delete id

Gọi hàm delete trong thư viện để xóa Shop có id cho trước.

Gợi ý: Sử dụng kiến thức Reflection và ý tưởng “Call by MethodName” hoặc “Invoke Dynamically Method” trong GOLANG để gọi hàm một cách linh động.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

### Chương trình 6.1

Hãy tự chọn một hệ quản trị cơ sở dữ liệu để tạo thông tin cho 2 bảng (Table) sau:

- Address: Mô tả thông tin về địa chỉ.

Cột dữ liệu	Yêu cầu	Ý nghĩa
id	Required	Số thứ tự duy nhất để định danh cho dòng dữ liệu
full_addr	Required, Unique	Địa chỉ đầy đủ
lng	Optional	Longitude (Kinh độ) của full_addr
lat	Optional	Latitude (Vĩ độ) của full_addr
created_at	Required	Cho biết dòng dữ liệu được tạo vào lúc nào
updated_at	Optional	Cho biết dòng dữ liệu được thay đổi gần đây nhất vào lúc nào
deleted_at	Optional	Cho biết dòng dữ liệu được xóa vào lúc nào

- Shop: mô tả thông tin về các cửa hàng. Trong đó có trường dữ liệu `addr_id` là lưu trữ số nguyên để tham chiếu tới bảng Address. Cụ thể làm tham chiếu tới trường `id` của bảng Address.

Cột dữ liệu	Yêu cầu	Ý nghĩa
id	Required	Số thứ tự duy nhất để định danh cho dòng dữ liệu
addr_id	Required	Tham chiếu tới Address.id
name	Required	Tên cửa hàng
short_name	Optional	Tên viết tắt của cửa hàng
created_at	Required	Cho biết dòng dữ liệu được tạo vào lúc nào
updated_at	Optional	Cho biết dòng dữ liệu được thay đổi gần đây nhất vào lúc nào
deleted_at	Optional	Cho biết dòng dữ liệu được xóa vào lúc nào

Viết một GOLANG package (module) cung cấp các tiện ích (các hàm có thể gọi bởi chương trình GOLANG khác) để thao tác với đối tượng Shop gồm các thông tin (Name, Short Name, Full Name) :

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Chức năng	Ý nghĩa
<b>Add</b>	Thêm mới
<b>Update</b>	Cập nhật
<b>Get()</b>	Lấy danh sách cửa hàng với trường deleted_at khác rỗng.
<b>Delete</b>	Xóa bằng cách cập nhật trường dữ liệu deleted_at là thời gian tại lúc chức năng này được thực thi..
<b>GetBy(field_name, operator, value)</b>	Lấy danh sách cửa hàng bởi câu truy vấn SELECT theo điều kiện WHERE field_name operator value Ví dụ: field_name là “Name”, operator là “like”, và value là ““%Abc%”” (nội dung bao gồm dấu nháy) thì câu query có dạng như sau:  SELECT Name, Address WHERE Name like “%Abc”

Ghi chú:

- Tên hàm và các tham số do bạn tự thiết kế (sao cho phù hợp với Business Logic hoặc phù hợp với tên hàm trong các thư viện bạn sử dụng. Ví dụ các thư viện ORM như GORM).
- Thông tin kết nối Database được lưu trong file cấu hình để dễ dàng thay đổi.

Hãy tự viết code để thử hoặc test (dùng Unit Testing nếu biết) các hàm ở trên.

Đóng gói thư viện ở trên thành mã nhị phân (để sẵn sàng cung cấp cho đồng đội sau này). Gọi tắt là thư viện.

### **Chương trình 6.2**

Viết chương trình GOLANG tên là “PackageRelection” sử dụng tính năng Reflection, Interface của GOLANG để phân tích các thông tin của gói nhị phân trong chương trình 6.1 (gọi tắt là thư viện). Thông tin tối thiểu cần phân tích: tên hàm, tham số và kiểu dữ liệu của hàm. Có thể viết thử code để gọi hàm trong thư viện.

Bạn tự quy định thư mục chứa thư viện.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

### Chương trình 6.3

Viết chương trình GOLANG dạng console có tên là “ShopCRUD” nhận tham số dòng lệnh như sau:

```
func_name param1, param2,...
```

Tùy theo giá trị của func\_name thì các tham số tiếp theo sẽ có ý nghĩa và số lượng khác nhau.

Sử dụng thư viện trong mục 6.1 để thực hiện CRUD (Thêm, Tìm kiếm, Sửa, Xóa) dữ liệu về SHOP.

Bạn tự tạo file cấu hình và đặt vào thư mục phù hợp với yêu cầu của thư viện.

Tham số func\_name trùng với tên hàm trong thư viện 6.1.

Cách sử dụng chương trình ShopCRUD với các tham số theo các dạng như sau:

#### 2) add name short\_name full\_add

Ví dụ:

```
ShopCRUD add "Cửa hàng máy tính Cầu Giấy" "MTCG" "Số 1, Kim  
Mã, Cầu Giấy, HN"
```

**Ý nghĩa: Gọi hàm add trong thư viện để thêm cửa hàng “Cửa hàng máy tính Cầu Giấy”, tên viết tắt “MTCG”, và địa chỉ “Số 1, Kim Mã, Cầu Giấy, HN” vào Database.**

#### 3) get

Gọi hàm get trong thư viện để lấy danh sách cửa hàng (không lấy và không hiển thị các thông tin hệ thống như id, created\_at, updated\_at, deleted\_at

#### 4) update id name short\_name full\_add

Gọi hàm update trong thư viện để cập nhật thông tin của cửa hàng có id cho trước. Các thông tin tiếp theo tương ứng trên tham số dòng lệnh. Giá trị nào rỗng (tham số là “”) thì không cập nhật.

### Thử thách sau ngày 7

Chủ động học và áp dụng kỹ thuật lập trình đồng thời và lập trình song song.

### Thử thách sau ngày 8

Chủ động học và áp dụng Unit Testing cho các chương trình đã làm.

### Thử thách sau ngày 9

Chủ động học và áp dụng Gin-Gonic để viết Restful API.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

## Thử thách sau ngày 10

Chủ động học và áp dụng GOLANG Swagger để viết Restful API.

Ví dụ có 2 trang phổ biến:

- <https://github.com/go-swagger/go-swagger>
- <https://github.com/swaggo/swag>

## Final Project #1

### *Project #1 – Module 1: datacmd*

Nhu cầu chung (User story)

Người dùng cần một chương trình dạng dòng lệnh để nạp dữ liệu từ file CSV vào CSDL.

Yêu cầu chức năng

Module **datacmd**:

- Chương trình dạng dòng lệnh có cú pháp như sau:

```
datacmd <file_path> <separator>
```

<file\_path>: là đường dẫn của file CSV. Tên file CSV có định dạng như sau:

```
<folder_path>\<table_name>_<action>.csv
```

Ví dụ với <table\_name> là product thì

<action> có giá trị (phần bôi đậm) và ý nghĩa trong tên file như sau:

- product\_**new**.csv: chứa đầy đủ các cột để thêm mới dữ liệu.
- product\_**edit**.csv: chứa ít nhất cột id và 1 cột dữ liệu khác cần cập nhật product. Cột nào có xuất hiện trong file thì cập nhật thông tin tương ứng cho cột đó. Cột nào không xuất hiện trong file thì không đụng đến.
- product\_**del**.csv: chứa sẵn cột **id** để chỉ định các dòng dữ liệu sẽ xóa (không quan tâm đến các cột dữ liệu khác)
- product\_**exp**.csv: chứa sẵn các tên cột cần để lấy dữ liệu. File kết quả có dạng product\_exp\_yyymmdd-hhmmss-ms.csv. Lưu ở đâu là do bạn thiết kế.

- Phần thời gian “yyymmdd-hhmmss-ms” lưu đến mức milliseconds)

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

<separator>: là kí tự phân cách các cột dữ liệu trong file CSV

Dữ liệu mẫu của file `product_new.csv` có dạng như sau:

id	name	description	created_at	price
1	Bàn gỗ nhỏ A1	Kích thước 20 x 30	2020-07-10 03:22:20	123.5
2	Tủ lớn T12	Gỗ tự nhiên	2020-07-10 04:42:26	
...				

- Chương trình `datacmd` đọc một file cấu hình `datacmd.yml` bên trong thư mục ứng dụng. File `datacmd.yml` có nội dung và ý nghĩa như ghi chú:

```
## Cấu hình gọi hàm động từ một thư viện khác để xử lý các hành động
# Gọi hàm động cho action new
new:
  func_name:

# Gọi hàm động cho action new
edit:
  func_name:

# Gọi hàm động cho action del
del:
  func_name:

# Gọi hàm động cho action exp
exp:
  func_name:
```

- Tùy theo giá trị “`func_name`” được cấu hình cho mỗi hành động ở trên thì bạn sẽ gọi hàm thực sự được người khác (cũng có thể chính bạn) lập trình và cung cấp (dưới dạng mã nguồn hoặc đóng gói dạng binary) trong [Module #2](#).

Yêu cầu khác

- Tên dự án GOLANG là **datacmd**
- Có script `build.cmd` hoặc `build.sh` (áp dụng trên Linux/Mac) để biên dịch dự án ra file `datacmd.exe` (trên Windows) hoặc `datacmd` (trên Linux/Mac)
- Dùng logging để lưu lại các thông tin sau mới level là Debug:
  - o Câu query chi tiết hoặc định dạng và giá trị các biến đi kèm (trường hợp có dùng sql template thì log ra query và các biến riêng chứ không cần log ra câu query hoàn chỉnh)

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

- Tự tổ chức thư mục trên gitlab.com, làm và commit mỗi ngày. Khi đã viết được một đoạn code thực hiện được một việc nhỏ nào đó thì commit và push để người khác thấy tiến độ của mình mỗi ngày.
- Áp dụng GOLANG coding style từ các nguồn ở đây:
  - o <https://github.com/smallnest/go-best-practices>
  - o <https://github.com/bahlo/go-styleguide>
- Bạn có quyền đề xuất thay đổi các cấu hình, cách thức xử lý nghiệp vụ, tài liệu GOLANG coding style sẽ áp dụng và phải có xác nhận trước khi nộp sản phẩm.
- Bạn có thể chủ động làm file Q&A để ghi các câu hỏi và câu trả lời giả định, thông báo cho người quản lý để xem và xác nhận. Trong thời gian đợi xác nhận thì bạn có thể làm theo giả định đã ghi – bạn cho là phù hợp nhất.

### *Project #1 – Module 2: datacore*

Nhu cầu chung (User story)

Người dùng cần một thư viện để xử lý dữ liệu đơn giản với CSDL MS SQL hoặc MySQL, hoặc PostgreSQL. Các nghiệp vụ bao gồm Thêm, Sửa, Xóa, Export dữ liệu.

Yêu cầu chức năng

Module **datacore**:

- Chương trình được đóng gói và cung cấp dạng binary (giống như DLL trên windows, file .o trên Linux/Mac) hoặc mã nguồn (nếu người gọi chưa quen). Người gọi ở đây là người lập trình [Module #1](#) ở trên.
- Module cung cấp các hàm với cú pháp như sau:
  - process\_new(table\_name, \*File) (int, error)
  - process\_edit(table\_name, \*File) (int, error)
  - process\_del(table\_name, \*File) (int, error)
  - process\_export(table\_name, \*File) (outfile \*File, error)

Xem tài liệu kiểu file tại:

https://pkg.go.dev/os#File

- table\_name: Là tên Table trong CSDL.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

- Khi các hàm trên có lỗi xảy ra với bất kỳ lý do gì thì CSDL không bị thay đổi (dữ liệu được xử lý trong 1 transaction, khi có lỗi thao tác dữ liệu thì thực hiện **rollback**)
- Thư viện `datacore` đọc một file cấu hình `datacore.yml` có nội dung và ý nghĩa như ghi chú:

```
## Cấu hình kết nối cơ sở dữ liệu
database:
  type: "postgres"
  host: "localhost"
  port: 5439
  name: "ecp"
  user: "ecp_user"
  password: "Ecp!123"
  sslmode: "disable"
```

(type có các giá trị: `postgres` | `mysql` | `mssql`. Các thuộc tính khác sẽ thay đổi giá trị tương ứng với CSDL mà bạn sử dụng)

### Yêu cầu khác

- Tên dự án GOLANG là `datacore`
- Có script `build.cmd` hoặc `build.sh` (áp dụng trên Linux/Mac) để biên dịch dự án.
- Dùng logging để lưu lại các thông tin sau mới level là `Debug`:
  - Câu query chi tiết hoặc định dạng và giá trị các biến đi kèm (trường hợp có dùng sql template thì log ra query và các biến riêng chứ không cần log ra câu query hoàn chỉnh)
- Có sử dụng Unit Testing để đảm bảo các hàm đã được testing các luồng chính.
- Tự tổ chức thư mục trên `gitlab.com`, làm và commit mỗi ngày. Khi đã viết được một đoạn code thực hiện được một việc nhỏ nào đó thì commit và push để người khác thấy tiến độ của mình mỗi ngày.
- Áp dụng GOLANG coding style từ các nguồn ở đây:
  - <https://github.com/smallnest/go-best-practices>
  - <https://github.com/bahlo/go-styleguide>
- Bạn có quyền đề xuất thay đổi các cấu hình, cách thức xử lý nghiệp vụ, tài liệu GOLANG coding style sẽ áp dụng và phải có xác nhận trước khi nộp sản phẩm.



Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

- Bạn có thể chủ động làm file Q&A để ghi các câu hỏi và câu trả lời giả định, thông báo cho người quản lý để xem và xác nhận. Trong thời gian đợi xác nhận thì bạn có thể làm theo giả định đã ghi – bạn cho là phù hợp nhất.

Gợi ý:

Thiết kế một CSDL, bảng “product” có cấu trúc cơ bản như sau:

id	name	description	created_at	price
integer	varchar(256)	text	datetime	decimal

## Final Project #2

### *Project #2 – dwhcmd*

Nhu cầu chung (User story)

Người dùng cần một chương trình dạng dòng lệnh để tổng hợp dữ liệu từ một CSDL nguồn vào CSDL đích.

Yêu cầu chức năng

Chương trình dạng dòng lệnh **dwhcmd** được mô tả như bên dưới:

- Chương trình dạng dòng lệnh có cú pháp như sau:

```
dwhcmd <sub_cmd> <config_folder> <sub_args>
```

<sub\_cmd>: sub command, là lệnh tương ứng (để gọi chương trình con)

<sub\_args>: tham số bổ sung cho sub command

<config\_foler>: là thư mục chứa các file cấu hình. Cụ thể như sau:

Tên file	Ý nghĩa
<b>source.yml</b>	## Ví dụ cấu hình kết nối cơ sở dữ liệu <b>database:</b> <b>type:</b> "mysql" <b>host:</b> "localhost" <b>port:</b> 3306 <b>name:</b> "ecp" <b>user:</b> "ecp_user" <b>password:</b> "Ecp!123" <b>sslmode:</b> "disable"
<b>dest.yml</b>	## Ví dụ cấu hình kết nối cơ sở dữ liệu

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

	<b>database:</b> <b>type:</b> "postgres" <b>host:</b> "localhost" <b>port:</b> 5439 <b>name:</b> "ecp" <b>user:</b> "ecp_user" <b>password:</b> "Ecp!123" <b>sslmode:</b> "disable"
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Cấu trúc dữ liệu trong CSLD nguồn gồm có 1 bảng “product” và bảng “order” với cấu trúc như sau:

Table **product**:

Column name	Data type	Description
id	integer	Khóa chính, tự tăng
name	varchar(256)	Trỏ tới cột product.id
description	text	
created_at	datetime	Ngày nhập sản phẩm
price	decimal	Giá gốc

Table **order**:

Column name	Data type	Description
id	integer	Khóa chính, tự tăng
product_id	decimal	Trỏ tới cột product.id
quantity	integer	Số lượng
unit_price	decimal	Đơn giá
created_at	datetime	Ngày phát sinh đơn hàng
modified_at	datetime	Ngày chỉnh sửa đơn hàng. Hành động sửa xem trong cột modified_type
modified_type	integer	1: Chỉnh sửa giá trị hoặc số lượng mặt hàng. 2: Xóa đơn hàng (đơn hàng bị hủy)

Cấu trúc dữ liệu trong CSLD đích gồm có 1 bảng “dim\_date” và bảng “fact\_order” với cấu trúc như sau:

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Table **dim\_date**:

date_id	date_actual
integer	date

Script để tạo table có dạng như sau:

```
create table dim_date
(
    date_id INT not null auto,
    date actual DATE not null
)
```

Table **fact\_order**:

Column name	Data type	Description
day_dim_id	integer	Trỏ tới dim_date.date_id
sum	decimal	Tổng giá trị các đơn hàng trong ngày (tra cột dim_date.date_actual)
mean	decimal	Giá trị trung bình các đơn hàng trong ngày
median	decimal	Giá trị trung vị các đơn hàng trong ngày
min	decimal	Giá trị nhỏ nhất các đơn hàng trong ngày
max	decimal	Giá trị lớn nhất các đơn hàng trong ngày

Hãy viết sub command **gendate** để thực hiện các việc sau:

- Viết chương trình **gendate** để biên dịch thành lệnh **gendate** có chức năng phát sinh dữ liệu cho bảng **dim\_date** với cú pháp sử dụng như sau:

```
gendate <start_date> <n>
```

<start\_date> có dạng: yyyy-mm-dd

<n>: là số năm tính thêm từ năm hiện tại của máy tính. Mặc định là 0. Giá trị <n> này dùng để xác định end\_date như ví dụ sau:

Ví dụ lệnh “gendate 2021-7-31” sẽ phát sinh các record cho cột dim\_date.actual\_date từ ngày 31/7/2021 đến ngày cuối năm hiện tại 31/12/2021.

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

Ví dụ khác: lệnh “gendate 2000-1-1 1” sẽ phát sinh các record cho cột `dim_date.actual_date` từ ngày 1/1/2000 đến ngày cuối năm hiện tại cộng thêm 1. Năm hiện tại là 2021 thì năm hiện tại cộng thêm 1 là năm 2022. Tức là sẽ phát sinh dữ liệu từ ngày 1/2/2000 đến 31/12/2022.

Tương tự lệnh “gendate 2000-1-1 3” với ngày hệ thống là năm 2021 thì sẽ phát sinh dữ liệu từ ngày 1/2/2000 đến 31/12/2024.

Trường hợp record trong CSDL đã có sẵn thì bỏ qua (không làm thay đổi dữ liệu có sẵn).

- Có 2 cách gọi sub command `gendate` như sau:

- Gọi thông qua lệnh `dwhcmd`

```
dwhcmd gendate <config_folder> <start_date> <n>
```

Lúc này chương trình `gendate` đọc thông tin cấu hình trong thư mục `<config_folder>`.

- Gọi trực tiếp

```
gendate <start_date> <n>
```

Lúc này chương trình `gendate` đọc thông tin cấu hình trong thư mục mặc định. Ví dụ thư mục “**config**” ngang cấp với lệnh `gendate.exe`.

- Viết sub command `loadorder` để biên dịch thành lệnh `loadorder` có chức năng tổng hợp dữ liệu từ database nguồn sang database đích như sau:
  - Nghiệp vụ, cách tổng hợp số liệu `sum`, `mean`, `median`, `min`, `max` như đã mô tả ở trên.
  - Giá trị đơn hàng được tính bằng: `quantity * unit_price`
  - Lưu lại giá trị **`order.created_at`** mới nhất đã được xử lý, lưu vào file hoặc CSDL do bạn tự thiết kế (gọi chung là biến **`last_order_created_at`**). Hoặc mỗi lần chạy bạn tự xác định biến **`last_order_created_at`** từ 2 bảng `dim_date`, `fact_order`.
  - Nghiệp vụ xử lý không quan tâm đến 2 cột `order.modified_at` và `order.modified_type`. Phần này được đề cập trong Chức năng nâng cao bên dưới.
- Có 2 cách gọi sub command `loadorder` như sau:
  - Gọi thông qua lệnh `dwhcmd`

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

```
dwhcmd loadorder <config_folder>  
<last_order_created_at>
```

Lúc này chương trình loadorder đọc thông tin cấu hình trong thư mục <config\_folder>.

○ Gọi trực tiếp

```
loadorder
```

Lúc này chương trình gendate đọc thông tin cấu hình trong thư mục mặc định. Ví dụ thư mục “**config**” ngang cấp với lệnh loadorder.exe.

Chức năng nâng cao (điểm từ 8 đến 10)

Nâng cấp chương trình **dwhcmd** ở trên để bổ sung thêm tính năng sau:

- Trường hợp CSDL nguồn có thay đổi (thêm, sửa, xóa) dữ liệu trước hoặc trong ngày **last\_order\_created\_at** thì được phản ánh như sau:
  - Muốn biết có phát sinh đơn hàng mới sau thời điểm **last\_order\_created\_at** thì tra trong cột order.created\_at với điều kiện sau:
    - order.created\_at > last\_order\_created\_at
  - Muốn biết đơn hàng có **chỉnh sửa số liệu** sau thời điểm **last\_order\_created\_at** thì tra 2 cột order.modified\_at và order.modified\_type với 2 điều kiện đồng thời như sau:
    - order.modified\_type = 1
    - order.modified\_at > last\_order\_created\_at
  - Muốn biết đơn hàng có **xóa** sau thời điểm **last\_order\_created\_at** thì tra 2 cột order.modified\_at và order.modified\_type với 2 điều kiện đồng thời như sau:
    - order.modified\_type = 2
    - order.modified\_at > last\_order\_created\_at

Gợi ý cách thực hiện:

- Nâng cấp chương trình thêm 1 tham số như sau:

Đây là eBook của riêng bạn – đề nghị không chia sẻ cho ai khác nhé!

```
loadorder <type>
```

Nếu <type> không được truyền thì xử lý như trong mục “Yêu cầu chức năng”.

Nếu <type> = update thì sẽ rà soát bên dữ liệu nguồn để xác định các đơn hàng nào được Thêm | Sửa | Xóa sau thời điểm **last\_order\_created\_at** để tính toán lại số liệu trong bảng fact\_order cho đúng.

#### Yêu cầu khác

- Tên dự án GOLANG là dwhcore
- Có script `build.cmd` hoặc `build.sh` (áp dụng trên Linux/Mac) để biên dịch dự án.
- Dùng logging để lưu lại các thông tin sau mới level là Debug:
  - o Câu query chi tiết hoặc định dạng và giá trị các biến đi kèm (trường hợp có dùng sql template thì log ra query và các biến riêng chứ không cần log ra câu query hoàn chỉnh)
- Có sử dụng Unit Testing để đảm bảo các hàm đã được testing các luồng chính.
- Tự tổ chức thư mục trên gitlab.com, làm và push mỗi ngày. Khi đã viết được một đoạn code thực hiện được một việc nhỏ nào đó thì commit và push để người khác thấy tiến độ của mình mỗi ngày.
- Áp dụng GOLANG coding style từ các nguồn ở đây:
  - o <https://github.com/smallnest/go-best-practices>
  - o <https://github.com/bahlo/go-styleguide>
- Bạn có quyền đề xuất thay đổi các cấu hình, cách thức xử lý nghiệp vụ, tài liệu GOLANG coding style sẽ áp dụng và phải có xác nhận trước khi nộp sản phẩm.
- Bạn có thể chủ động làm file Q&A để ghi các câu hỏi và câu trả lời giả định, thông báo cho người quản lý để xem và xác nhận. Trong thời gian đợi xác nhận thì bạn có thể làm theo giả định đã ghi – bạn cho là phù hợp nhất.

[Kết thúc thử thách]

Liên lạc giao lưu: <https://facebook.com/ThachLN>