

Th.S LÊ NGỌC THẠCH

Lời nhắn

eBook "**Chạm tới GO trong 10 ngày**" này dự kiến phát hành vào ngày 31/12/2021. 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

0908550642  Lê Ngọc Thạch

 Nội dung tin nhắn: GO2021 email sdt
Ví dụ: abc@gmail.com 0908550642
Email và sdt của người nhận eBook.

Trường hợp tặng bạn bè thì ghi thông tin email và sdt của bạn.

Quét mã QR thanh toán 199K.



199.000đ

② Chuyển khoản

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:



Quét mã vạch này để giao dịch

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.....	12
Mã nguồn.....	12
Lệnh thực thi trong hệ điều hành.....	12
Đường dẫn hiện hành.....	12
Hệ điều hành Windows và Linux/Mac	12
Cấp dấu nhảy.....	13
Cách viết trình tự bấm chọn menu	13
Đường dẫn thư mục (Path)	13
Các từ viết tắt, tiếng Anh thường xuyên được sử dụng trong sách.....	13
Cách viết dấu chấm câu, ghi chú hình, bảng biểu.....	14
Ôn tập kiến thức cơ bản về máy tính và phần mềm.....	16
Ôn tập #1: Quá trình tiến hóa của các mô hình phần mềm.....	17
Phần mềm trên máy cá nhân.....	18
Máy vi tính cá nhân (personal computer).....	18
Giao diện console	19
Giao diện đồ họa (GUI – Graphics User Interface)	21
Phần mềm trên mạng nội bộ	23
Mạng nội bộ (LAN - Local Network).....	23
Phần mềm trên nền tảng mạng Internet.....	25
Mạng Internet.....	25
Phần mềm trên mạng Internet.....	25
Ôn tập #2 – Cấu trúc của phần mềm.....	27
Công thức I + P + O.....	28
Thu nhận thông tin.....	28

Xử lý thông tin	28
Xuất kết quả.....	28
Ví dụ.....	28
Ngày 1: Làm quen với GOLANG	30
Thử thách trong chương 1	30
Bài 1 – Tại sao GO ra đời	33
Bài 2: Ngôn ngữ lập trình GO	34
Biến (Variable), Cấu trúc (Structure).....	34
Variable có nghĩa là gì?	36
Khai báo biến (variable declaration).....	37
Lệnh gán (assign).....	38
Bài 3 – Chuẩn bị môi trường lập trình.....	40
GO Core	40
Cài thêm thư viện	40
Visual Code	41
Cài GO trên Ubuntu	43
Bài 4 – Viết chương trình đơn giản với GO	44
Viết mã.....	44
Biên dịch.....	44
Chạy trực tiếp mã nguồn.....	46
Phép gán (assign).....	48
Các toán tử cơ bản.....	50
Hàm (function).....	50
Chạy chương trình có tham số dòng lệnh trong Visual Code.....	52
Lấy tham số từ dòng lệnh	52
Vòng lặp (loops)	53
Nâng cao	54
Bài 5 – Biểu diễn thông tin đơn giản với GO.....	56
Kiểu chuỗi (string).....	56
Xem kiểu dữ liệu của biến.....	57
Kiểu dữ liệu số (Numeric data types)	57
Viết chương trình Fibonacci.....	63

Mảng (arrays).....	64
Slice – Mảng không giới hạn độ dài	66
Maps.....	71
Thời gian (Times & dates).....	72
Tra cứu định dạng.....	74
Bài tập cuối ngày 1	77
Ngày 2 – Cách viết một phần mềm đơn giản.....	78
Bài 1 – Phần mềm đầu tiên – Cài đặt phép toán cộng.....	79
Bước 1: Xác định và viết yêu cầu.....	80
Bước 2: Làm thiết kế.....	83
Bước 3: Lập trình và kiểm thử.....	90
Bước 3.1: Kiểm thử mã nguồn.....	112
Bước 3.2: Hoàn thiện giao diện.....	114
Bước 4: Kiểm thử hệ thống.....	115
Thử thách cho bạn: #1	120
Thử thách cho bạn: #2.....	120
Phiên bản 1: Hỗ trợ truyền văn bản trên dòng lệnh.....	120
Vd: lệnh.....	121
Tham khảo thêm source code bằng CSharp:.....	121
Tham khảo source code bằng C.....	122
Tham khảo thêm source code bằng Python.....	123
Thử thách cho ngày 2.....	124
Ngày 3: Biểu diễn thông tin phức hợp.....	127
Bài 1 – Biểu diễn thông tin phức hợp với GO.....	128
Cấu trúc (Structure).....	128
Kết hợp Slice và Structure.....	128
Con trỏ (Pointer).....	130
Tuples (Bộ dữ liệu).....	132
Đọc thêm và thực hành.....	135
Chuỗi (String)	135
Regular expressions and pattern matching	136
Bài 2 – Viết hàm cho cấu trúc	138

Phân tích hàm calculateBMI cho struct Employee	138
Tổ chức thành thư viện (module)	139
Bài 3 – Dữ liệu dạng JSON	142
Đọc dữ liệu JSON	142
Chương Ngày 4	143
Thử thách cho ngày 4	143
Chương 5: Cấu trúc điều khiển	146
Thử thách trong chương 4	147
Bài 7 – Cấu trúc rẽ nhánh	149
Lệnh if	149
Switch	149
Bài 8 – Vòng lặp	152
Vòng lặp (loops)	152
Vòng lặp for nâng cao	153
Chương 6: Làm việc với dữ liệu trên đĩa cứng	158
Bài 1 – Làm việc với thư mục và file	159
Lấy thông tin về file/thư mục	159
Lấy nội dung thư mục	161
Lấy nội dung file	161
Lưu file	161
Lưu và đọc file mã hóa	162
Bài 2 – Làm việc với file CSV	166
Bài 3 – Đọc file CSV	168
Bài 4 – Ghi file CSV	170
Bài 5 – File và cấu trúc (struct)	171
Cài đặt thư viện	171
Đọc đoạn dữ liệu binary vào mảng các struct	171
Đọc file CSV vào mảng các struct	172
Ghi mảng các struct ra file CSV	175
Bài 6 – Đọc file văn bản	176
Bài 7 – Đọc file Excel	177
Cài đặt	177

Đọc file Excel.....	177
Ghi dữ liệu ra file Excel.....	178
Chương 7: Tổ chức dự án GOLANG.....	180
Bài 1 – Tổ chức mã nguồn.....	181
Bước 1: Tạo file go.mod để mô tả tên của module.....	181
Bước 2: Tạo thư mục và file chứa hàm dùng chung.....	182
Bước 3: Viết chương trình chính.....	182
Bài 2 – Tinh chỉnh mã nguồn.....	190
Phiên bản 0.0.2.....	190
Phiên bản 0.0.3.....	191
Phiên bản 0.0.4.....	193
Bài 3 – Biên dịch dự án.....	195
Bài 3 – Tập thói quen viết phần mềm.....	196
Sử dụng logging.....	196
Bài 4 : Hàm trả về không phải là giá trị.....	200
Bài 3 – Go Packages và Functions.....	201
Anonymous function.....	201
Do it yourself:.....	202
Chương 8: Sử dụng cơ sở dữ liệu PostgreSQL.....	203
Bài 1 – Làm quen với CSDL.....	204
Bài 2 – Sử dụng PostgreSQL portable.....	205
Tải gói binary.....	205
Tạo file khởi động PostgreSQL server.....	206
Khởi động PostgreSQL server.....	207
Tương tác với PostgreSQL Server qua dòng lệnh.....	207
Tương tác với PostgreSQL Server qua web site.....	211
Tương tác với PostgreSQL Server qua web site.....	214
Bài 3 – Thực hành với PostgreSQL.....	215
Tạo một CSDL “ECP”.....	215
Nhập dữ liệu.....	219
Bài 4 – GOLANG và PGSQL.....	222
Cài thư viện.....	222

Ví dụ.....	222
Giải thích code từ ví dụ.....	223
Bài 5 – Sử dụng file cấu hình.....	226
Cài thư viện viper	226
Ngày 7: Sử dụng MySQL.....	227
Bài 1 - Tự chuẩn bị MySQL server	228
Tải MySQL.....	228
Cấu hình.....	228
Khởi tạo dữ liệu hệ thống.....	228
Thiết lập mật khẩu.....	228
Bài 2 – GOLANG và MySQL.....	231
Sử dụng thư viện go-sql-driver.....	231
Ngày 8: Sử dụng SQL Server.....	240
Bài 1 - Tự chuẩn bị MySQL server	241
Bài 2 – GOLANG và SQL Server	242
Kết nối SQL Server.....	242
Ngày 8: Interface	244
Bài 1 - Interface.....	245
Khái niệm	245
Khai báo interface.....	245
Cài đặt interface.....	246
Ngày 9 – Đảm bảo chất lượng mã nguồn	249
Đảm bảo mã nguồn trong sáng, dễ chỉnh sửa.....	250
Phát hiện những lỗi có thể nhìn thấy ngay.....	251
Đảm bảo các chức năng nhỏ nhất không có lỗi.....	252
Unit Testing.....	252
Dọn dẹp log files.....	254
Ngày 10: Lập trình đồng thời và song song với GO	256
Bài 1 – Khái niệm Concurrency và Parallelism.....	257
Tạo goroutine	257
Đợi hàm Goroutine chạy xong.....	259
Sử dụng channel cho goroutine.....	260

Khảo sát thêm ví dụ GoRoutineMessage.go sau:	262
Chỉ định rõ channel read-only write only	264
Pipeline	264
Bài 2 – Khái niệm Concurrency và Paralleilism	266
Bài 3 – Lập trình Concurrency	267
Bài 4: Lập trình Paralleilism	268
Thử thách ngày 9	269
Nâng cấp chương trình ImportCSV2DB	269
Nâng cấp chương trình DataTransform	269
Thử thách ngày 10	271
Nâng cao chất lượng mã nguồn	271
Ngày 8: GOLANG và C/C++	272
Bài 1 - Lập trình C trong GO	273
Ngày 9: Các chủ đề mở rộng/nâng cao	274
Bài 1 – Viết hàm với tham số linh động	275
Variadic functions	275
Bài 2 - Crawl dữ liệu với GOLANG	277
Request đơn giản	277
Thiết lập timeout cho request	277
Thiết lập header	278
Download URL	280
Use substring	281
Bài 3 - Lập trình CUDA với GOLANG	283
Bài 4 - Phát triển Web Application với Beego	284
Cài đặt GO	284
Cài đặt Beego	284
Tạo dự án	284
Chạy ứng dụng	286
Truy cập ứng dụng	286
Chỉnh sửa code	286
Tạo API	288
Bài tham khảo #5 - Phát triển Web Backend với Gin-Gonic	289

Cài đặt	289
Viết Backend đơn giản.....	289
Thử thách cho bạn.....	292
Triển khai lên server Ubuntu với Nginx	294
Nâng cấp ứng dụng Back-end.....	294
Cài đặt các thư viện hỗ trợ web	299
Bài 6 - Sử dụng GOLANG trong WSL2	300
Bài 7 – Sử dụng Makefile với GOLANG	301
Giới thiệu Makefile	301
Ngày 10: Tra cứu theo nhu cầu.....	303
Các API về xử lý chuỗi.....	304
Các API sử dụng GIN GO NIC	305
Ngày 11: Testing với GO	306
Biên dịch OpenCV từ mã nguồn	307
Cài đặt Anaconda:	307
Cài đặt mkl-service.....	307
Kiểm tra thông tin thiết bị GPU.....	307
Biên dịch.....	308
Ngày 12 - Blockchain	310
Bài 1: Ôn tập kiến thức cơ bản.....	311
Làm quen lại với kiểu Slice của byte	311
Thư viện bytes.....	312
Thư viện mã hóa	313
Mã hóa base58	313
Khóa công khai và khóa bí mật	315
Bài 2 – Tạo cấu trúc chuỗi khối	317
Tạo dự án	317
Viết mã nguồn main.go	317
Bài 3 – Minh họa thuật toán đồng thuận ProofOfWork.....	321
Định nghĩa bổ sung Block.....	321
Hàm tạo Block cũ	321
Hàm tạo Block cải tiến.....	321

Cài đặt ProofOfWork (PoW)	322
Bài 4 – Lưu trữ Blockchain	325
Sử dụng database dạng Key-Value	325
Đóng gói OpenSSL	327
Chuẩn bị công cụ	327
Cài đặt tool Visual Studio 2019	327
Clone mã nguồn dự án OmiseGo eWallet	328
Lập trình wxWidget	330
Phụ lục	331
Phụ lục 3	332
Lập trình giao diện với goki	333
Cài đặt GCC for Windows	333
Cài đặt thư viện goki	334
Viết ứng dụng	334
Biên dịch và chạy ứng dụng	335
GOLANG và QT	336
Cài đặt phần mềm QT	336
Cài đặt thư viện	341
Trải nghiệm lập trình	342
Phụ lục 4	344
GOLANG và Google Sheet	345
Phụ lục 5 – So sánh tốc độ truy cập dữ liệu giữa GOLANG và Python	346
Phụ lục 5 – Sample Project	350
The Links 'R'; Us Project	351
System overview – what are we going to be building?	351
Selecting an SDLC model for our project	352
Requirements analysis	354
System component modeling	361
Summary	368
Building a Persistence Layer	370
Technical requirements	370
Exploring a taxonomy of database systems	372

Understanding the need for a data layer abstraction	378
Designing the data layer for the link graph component.....	379
Data-Processing Pipelines.....	418
Synchronous versus asynchronous pipelines	425
Building a crawler pipeline for the Links 'R' Us project	440
Summary	458
Thử thách	459
Thử thách sau ngày 1	459
Thử thách sau ngày 4	460
Thử thách sau ngày 5	461
Thử thách sau ngày 6	462
Thử thách sau ngày 7	466
Thử thách sau ngày 8	466
Thử thách sau ngày 9	466
Thử thách sau ngày 10	467
Final Project #1	467
Final Project #2.....	471
Final Project #3.....	477
Final Project #4.....	482

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

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
Windows	Hệ điều hành Microsoft Windows

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

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.

<https://thachln.github.io/ebooks/cham-toi-GO-trong-10-ngay.html>

Ôn tập kiến thức cơ bản về máy tính và phần mềm

Nếu bạn viết được bài luận trả lời rành mạch các câu hỏi sau thì có thể bỏ qua phần này:

- ? Hãy nêu những khác biệt cơ bản giữa máy vi tính (computer) với các máy móc (machine) thông thường khác.
- ? Bạn hãy giải thích Phần mềm (Software) là gì và phân loại các phần mềm trên máy vi tính mà bạn đang dùng.
- ? Mạng toàn cầu (Internet) là gì? Hãy giải thích về Internet trong mối liên quan với Ứng dụng web (Web Application), và dẫn dắt từ lệnh (command) đơn giản nhất đang có trên máy tính mà bạn đang dùng.
- ? Công thức $I + P + O$ (gọi tắt IPO) trong lĩnh vực công nghệ thông tin là gì?

Ôn tập #1: Quá trình tiến hóa của các mô hình phần mềm

Bài này giúp các bạn hình dung các loại phần mềm phổ biến trên máy tính.

Phần mềm trên máy cá nhân

Máy vi tính cá nhân (personal computer)

Máy tính hay **máy điện toán** là những thiết bị hay hệ thống thực hiện tự động các phép toán số học dưới dạng số hoặc phép toán logic. Các **máy tính cỡ nhỏ** thường gọi là **máy vi tính**, trong số đó **máy dùng cho cá nhân** thường gọi là **máy tính cá nhân**.

Để một cái máy vi tính hoạt động được thì cần có phần mềm đặc biệt để điều khiển các thiết bị của nó gọi Hệ điều hành (Operating System). Các hệ điều hành phổ biến gồm:

- Microsoft Windows – thường được gọi tắt là Windows. Đây là hệ điều hành của hãng Microsoft. Windows như tên gọi của nó có biểu tượng là cửa sổ. Hình 1 là biểu tượng của 2 phiên bản Windows phổ biến hiện tại.

Hệ điều hành
✓ Là phần mềm đặc biệt để điều khiển máy vi tính



Hình 1: Biểu tượng Windows 7 và 10

- Macintosh - thường gọi tắt là Mac. Đây là hệ điều hành của hãng Apple.



Hình 2: Biểu tượng Quả táo của HĐH Macintosh

- Linux. Là hệ điều hành mã nguồn mở.



Hình 3: Biểu tượng Chim cánh cụt của HĐH Linux

Giao diện console

Từ những bản Hệ điều hành đầu tiên ra đời cho đến ngày nay thì việc ra lệnh cho máy vi tính thực hiện một công việc nào đó thông qua **cửa sổ gõ lệnh**¹ vẫn phổ biến.

Ví dụ 1 – Gõ lệnh:

Bạn có thể yêu cầu máy tính thực hiện một lệnh có sẵn trong OS Windows bằng cách mở cửa sổ dấu nhắc lệnh (Nhấn phím Windows + R), gõ cmd. Trong cửa sổ cmd.exe, gõ lệnh:

`dir d:`

Trong trường hợp bạn gõ một lệnh không có sẵn trong máy tính (vd: mvnc) thì máy tính sẽ báo câu lỗi như sau:

'mvnc' is not recognized as an internal or external command, operable program or batch file.

Dịch sát nghĩa: mvnc không được nhận diện như là một **lệnh bên trong** hoặc **bên ngoài**, một **chương trình có thể hoạt động** hoặc tập tin batch.

Internal command

✓ Là lệnh có sẵn trong hệ điều hành và được nạp sẵn vào bộ nhớ trong (RAM)

¹ Thuật ngữ tiếng Anh tương ứng có khác nhau trên các HĐH. Trong Windows gọi là “Prompt”. Trong Mac và Linux gọi là Terminal.

Giải thích:

- External command là lệnh bên ngoài (hệ điều hành). Tức là các phần mềm được lưu trữ trong đĩa cứng và được khai báo đường dẫn thư mục chứa nó trong biến môi trường PATH

- Tập tin batch là một tập tin văn bản có đuôi file là .bat và nội dung bên trong file gồm nhiều lệnh (batch có nghĩa là bó | khối | nhóm). Batch file này khi thực thi thì sẽ thực thi lần lượt các lệnh bên trong nó.

Lỗi trên có nghĩa là: "mvnc" không được máy tính hiểu là một lệnh hoặc một chương trình. Lý do có thể là một trong các tình huống sau:

- Nó không phải là lệnh có sẵn trong OS.
- Trong thư mục mà bạn đang gõ lệnh hoặc trong tất cả các thư mục được liệt kê trong biến môi trường PATH không có tồn tại một trong các file có thể thực thi có phần mở rộng như:
 - .com
 - .exe
 - .bat
 - .cmd
 - ...

External command

✓ Là lệnh bên ngoài hệ điều hành và không có sẵn trên bộ nhớ trong. Muốn máy tính hiểu lệnh này thì đường dẫn thư mục chứa nó phải được khai báo trong biến môi trường PATH.

Bài tập thực hành

1) Trong Windows, mở cửa sổ lệnh "cmd" gõ, quan sát, chụp hình kết quả và ghi chú hiểu biết hoặc suy đoán của các bạn vào một tài liệu để giải thích các lệnh sau:

1. path
2. path /?
3. echo %PATH%
4. set
5. set /?
6. cd

7. dir

8. hostname

9. hi

10. myname

- 2) Trong Linux/Mac, mở cửa sổ lệnh “terminal” gõ, quan sát, chụp hình kết quả và ghi chú hiểu biết hoặc suy đoán của các bạn vào một tài liệu để giải thích các lệnh sau:

11. echo \$PATH

12. set

13. set –help

14. help set

15. pwd

16. ls

17. hostname

18. hi

19. myname

Giao diện đồ họa (GUI – Graphics User Interface)

Nếu dùng máy tính mà chỉ gõ lệnh không thôi thì rất khó cho người không chuyên. Vì vậy các nhà làm phần mềm nghĩ ra cách để sáng tạo các phần mềm có giao diện đồ họa để người dùng tương tác với máy vi tính dễ dàng hơn.

Các phần mềm phổ biến dạng GUI là:

- Xử lý các công việc văn phòng: Microsoft Word, Microsoft Excel, Microsoft PowerPoint...
- Công cụ viết phần mềm cho dân lập trình: Microsoft Visual Studio, Eclipse, v.v...
- Soạn thảo văn bản đơn giản như: Notepad, Notepad Plus



Bài tập thực hành

- 1) Hãy google tìm phần mềm “Notepad++” để vào trang chủ “<https://notepad-plus-plus.org/>”. Sau đó tải và cài Notepad++ vào máy tính của bạn.

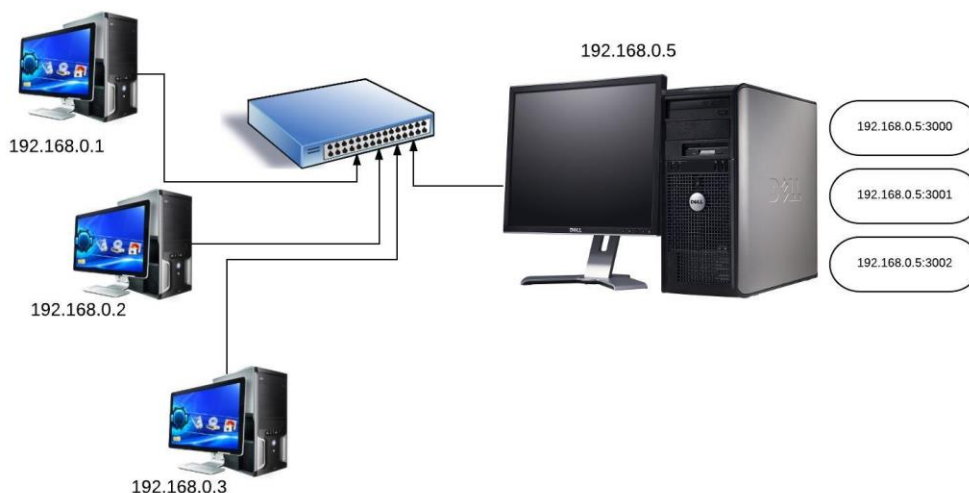
Hạn chế

Các phần mềm dạng Console hoặc GUI được cài đặt trên máy tính có ưu điểm là người dùng có thể bật máy tính và dùng ngay vì nó đã được cài sẵn trên máy. Tuy nhiên sẽ bất lợi cho các nhà sản xuất phần mềm khi cần nâng cấp phiên bản mới. Thông thường chúng ta phải tải và cài bản nâng cấp khi có phiên bản mới.

Phần mềm trên mạng nội bộ

Mạng nội bộ (LAN - Local Network)

Trong một tổ chức có nhiều máy tính được kết nối với nhau thì việc khai thác sức mạnh của các máy tính là cần thiết.



Hình 4: Một mạng máy tính đơn giản

Trong hình 4, các đường kẻ mũi tên chỉ sự kết nối giữa máy tính với một thiết bị trung tâm. Kết nối này có thể là dây cáp (cable) hoặc sóng không dây (Wireless). Phổ biến là Wifi.

Khi các bạn ra quán café kết nối vào Wifi là xem như bạn đã kết nối với mạng nội bộ của quán café.

Để xác định được máy tính của bạn trong một mạng thì người ta dùng địa chỉ IP Address (Internet Protocol Address). IP Address tương tự như địa chỉ nhà của bạn để giúp người đưa thư gửi thư đến đúng nhà bạn.

Để biết địa chỉ máy tính của bạn trong mạng thì gõ lệnh:

```
ipconfig
```

Trên Linux/Mac, gõ lệnh:

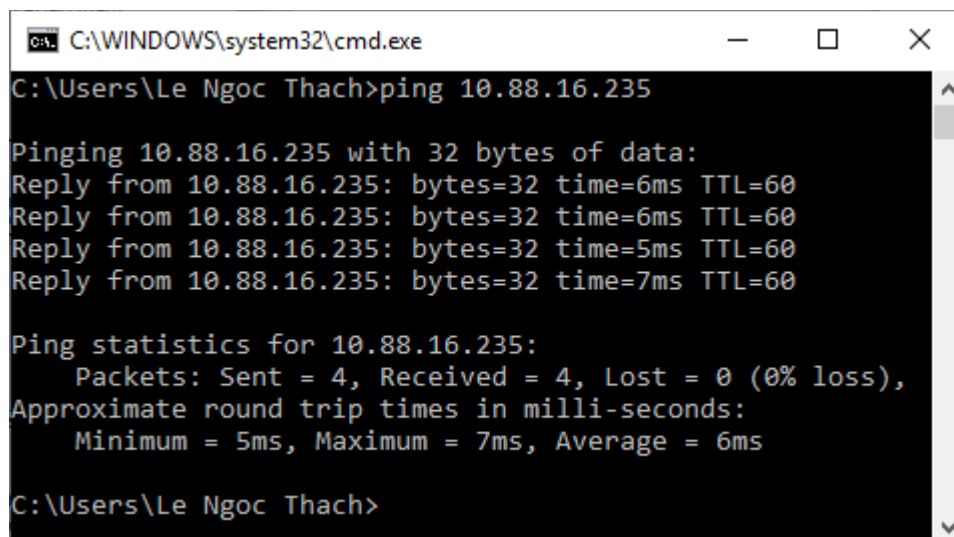
```
ifconfig
```

Để kiểm tra xem máy tính của mình có thể kết nối với một máy tính khác trong mạng nội bộ hay không thì dùng lệnh:

```
ping <ip address>
```

Ví dụ: Để kiểm tra xem máy tính của bạn có thể kết nối với máy tính có địa chỉ IP là 10.88.16.235 thì bạn gõ lệnh:

ping 10.88.16.235



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Le Ngoc Thach>ping 10.88.16.235

Pinging 10.88.16.235 with 32 bytes of data:
Reply from 10.88.16.235: bytes=32 time=6ms TTL=60
Reply from 10.88.16.235: bytes=32 time=6ms TTL=60
Reply from 10.88.16.235: bytes=32 time=5ms TTL=60
Reply from 10.88.16.235: bytes=32 time=7ms TTL=60

Ping statistics for 10.88.16.235:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 5ms, Maximum = 7ms, Average = 6ms

C:\Users\Le Ngoc Thach>
```

Bài tập thực hành

- 1) Hãy mượn máy tính của đồng nghiệp hoặc một người đang kết nối vào mạng (mạng dây hoặc mạng Wifi) gõ lệnh “ipconfig”. Ghi lại IP Address của máy tính đó.
Chú ý:
 - Nếu dùng mạng Wifi thì hãy quan sát địa chỉ của mạng Wifi (Tìm dòng nào có chữ tương tự: Wireless LAN adapter Wi-Fi)
- 2) Ngồi trên máy tính của mình gõ lệnh: ping <địa chỉ ip máy tính của đồng nghiệp>

Một bước cải tiến trong lĩnh vực phần mềm là thay vì phát triển các ứng dụng để chạy trên một máy vi tính – dùng giao diện CONSOLE hoặc GUI, thì giới lập trình có thể phát triển phần mềm đặt trên một cái máy nào đó (gọi là server) trong mạng. Người dùng có thể ngồi trên một máy tính khác nhưng vẫn có thể dùng được phần mềm trên server.

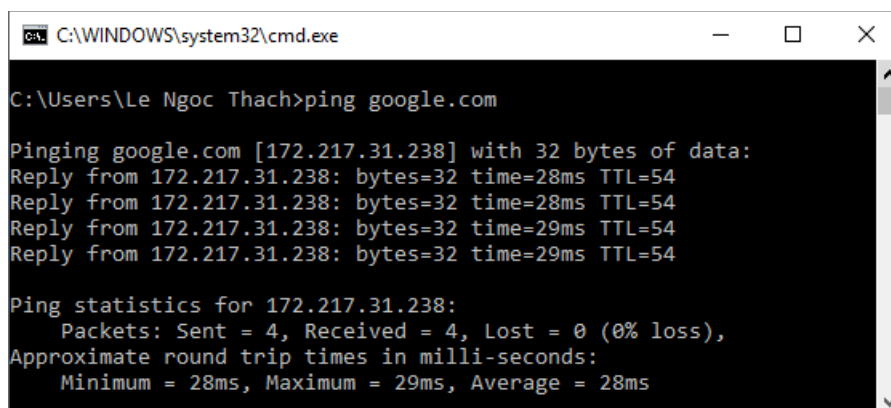
Phần mềm trên nền tảng mạng Internet

Mạng Internet

Trong phần trên các bạn đã biết khái niệm mạng cục bộ (LAN). Bây giờ tưởng tượng tất cả các mạng LAN được đấu nối với nhau (qua đường truyền điện thoại, hoặc cáp quang, hoặc vệ tinh, ...) thì khả năng là tất cả các máy tính trên thế giới có thể liên lạc được với nhau. Mạng khổng lồ này gọi là mạng Internet.

Trên mạng Internet người ta vẫn dùng địa chỉ IP để liên lạc với nhau. Tuy nhiên địa chỉ IP thì rất khó nhớ. Người ta phát minh ra việc ánh xạ địa chỉ IP thành một cái tên để dễ nhớ hơn gọi là tên miền (domain name).

Để biết địa chỉ IP của tên miền thì bạn dùng lệnh ping <tên miền. Ví dụ: ping google.com



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Le Ngoc Thach>ping google.com

Pinging google.com [172.217.31.238] with 32 bytes of data:
Reply from 172.217.31.238: bytes=32 time=28ms TTL=54
Reply from 172.217.31.238: bytes=32 time=28ms TTL=54
Reply from 172.217.31.238: bytes=32 time=29ms TTL=54
Reply from 172.217.31.238: bytes=32 time=29ms TTL=54

Ping statistics for 172.217.31.238:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 28ms, Maximum = 29ms, Average = 28ms
```

Phần mềm trên mạng Internet

Nhờ phát minh ra Internet nên giới lập trình có cơ hội viết ra các phần mềm và để nó lên một máy server. Sau đó mua dịch vụ ánh xạ máy chủ thành một cái tên cho dễ nhớ. Người dùng có thể truy cập phần mềm này thông qua tên miền.

Ví dụ : bạn có thể dùng phần mềm quản lý thư điện tử của Google qua tên miền bằng cách dùng trình duyệt gõ địa chỉ <https://mail.google.com>.

Bài tập thực hành

Hãy truy cập trang web <https://xlms.myworkspace.vn/portal> xem hướng dẫn trong mục **MESSAGE OF THE DAY** và thực hiện các việc sau:

1) Đăng ký tài khoản theo gợi ý sau:

- Bấm vào menu **New account** ở bên trái màn hình.

- Mục **User id** điền địa chỉ email của bạn

2) Tìm kiếm thử Khóa học và vào xem thử.

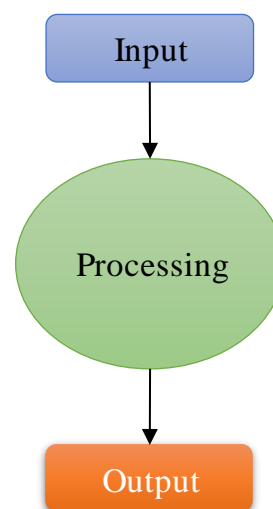
Ôn tập #2 – Cấu trúc của phần mềm

Bài này giúp bạn ghi nhớ 3 thành phần cơ bản trong hệ thống phần mềm. Đây là kim chỉ nam giúp bạn làm các bài tập cũng như xây dựng các công cụ trong tài liệu, hoặc khóa học đi kèm một cách đầy đủ, tự tin hơn.

Công thức I + P + O

Một phần mềm (PM) hoặc hệ thống thông tin (HTTT) nói chung gồm 3 phần:

- Thu nhận thông tin (**I**ntput)
- Xử lý thông tin thu nhận được (**P**rocess)
- Xuất kết quả (**O**utput)



Thu nhận thông tin

Thông thường thì người dùng sẽ **cung cấp thông tin** (nhập liệu) hoặc **ra lệnh cho phần mềm** thông qua bàn phím và chuột, cao cấp hơn qua micro (nói)

Xử lý thông tin

Thông tin được người dùng cung cấp sẽ được biến đổi, tổng hợp theo một hoặc nhiều mục đích nào đó.

Xuất kết quả.

Thông tin sau khi được xử lý có thể được trình bày cho người dùng biết, đồng thời có thể được lưu trữ lại để dùng về sau.

Cách thức trình bày phổ biến là:

- Hiển thị ra màn hình máy tính
- In ra giấy
- Lưu thành file trên máy tính để người dùng có thể tự xem bằng các phần mềm khác như: Word, Excel, PDF,...

Ví dụ

Bạn có thể dùng phần mềm xử lý bảng tính (Spreadsheet processing) như Excel, Open Office Calculator để nhập liệu và lưu thành file chứa thông tin của bạn bè mình. Sau khi bạn mở máy tính lên, khởi động phần mềm lên, quá trình IPO diễn ra như sau:

- Bạn **ra lệnh** cho phần mềm mở một tài liệu mới để bắt đầu soạn thảo bằng cách bấm phím tổ hợp phím Ctrl + N.
- Phần mềm Excel nhận lệnh Ctrl + N (**I**ntput) rồi xử lý (**P**rocessing) bằng cách mở ra một cửa sổ để bạn gõ nội dung vào (**I**ntput).

- Trong quá trình bạn gõ nội dung (Input) bạn có thể ra lệnh để Excel thực hiện (Processing) như: định dạng chữ đậm, nghiêng, ... sao chép (copy & paste), cắt dán (cut & paste), ...
- Trong quá trình bạn soạn nội dung thì phần mềm tự động lưu tài liệu vào đĩa cứng với tên file tạm (Output) với các kí tự đặc biệt để đề phòng trường hợp có sự cố.
- Sau khi soạn xong nội dung, bạn ra lệnh cho phần mềm lưu lại công sức của mình thì nhấn tổ hợp phím Ctrl+S. Phần mềm sẽ xử lý (Processing) bằng cách hiển thị hộp thoại để yêu cầu bạn cung cấp thông tin đường dẫn và tên file muốn lưu. Tiếp theo phần mềm sẽ lưu nội dung tài liệu với tên mà bạn đã cung cấp vào đường dẫn tương ứng (Output).

Như vậy bạn thấy rằng các hoạt động IPO diễn ra liên tục và đan xen với nhau tùy theo ý đồ thiết kế, sắp xếp chức năng của người lập trình.

Ngày 1: Làm quen với GOLANG

*Bạn không học được gì từ cuộc sống khi cho rằng bạn luôn đúng.
You learn nothing from life if you think you're right all the time.*

Hoàn thành các bài học trong ngày đầu tiên này, bạn có thể đạt được:

- Có kiến thức cơ bản về khái niệm **biến**, **phép gán** trong lập trình.
- Sử dụng được các lệnh cơ bản để hoàn thành một phần mềm đơn giản.
- Biết cách biên dịch, đóng gói thành phẩm.
- Làm quen và có thể sử dụng được các kiểu dữ liệu đơn giản trong GOLANG để viết ứng dụng.

Thử thách trong chương 1

Nếu bạn hoàn toàn chủ động làm được các chương trình sau và lưu kết quả lên GITLAB.COM thì có thể bỏ qua nội dung bài học trong chương 1 này. Bạn có thể chọn NNLT là Python hoặc GOLANG để thực hiện.

Chương trình 1 - BasicStatistics

Viết chương trình 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 - ReadNumber

Viết chương trình 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.
- Khuyến khích bạn chủ động nâng cấp chương trình để hỗ trợ số có tối đa 6, 9, 12 chữ số.

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**.
- **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.

Chương trình 4 – Cộng 2 số

Viết chương trình thực hiện cộng 2 số với cách sử dụng bảng cách gõ lệnh như sau:

```
sum 235 67
```

Kết quả hiển thị:

Các bước thực hiện:

Lấy '5' cộng '7' => Kết quả được 12, lưu 2, nhớ 1.

Lấy '3' cộng '6' => Kết quả được 6, cộng với nhớ 1 được 10, lưu 0, nhớ 1.

Lấy '2' cộng ' ' => Kết quả được 2, cộng với nhớ 1 được 3, lưu 3.
Kết quả cuối cùng: 312

Diễn giải yêu cầu:

- Yêu cầu viết hàm để thực hiện chức năng chính có dạng là tham số kiểu chuỗi như sau: `sum(st1 string, st2 string) string`

Khi nhận tham số từ dòng lệnh vào thì mặc định là chuỗi nên việc gọi hàm này rất thuận tiện.

- Phiên bản đầu tiên được PUSH lên GITLAB.COM chưa cần xử lý các trường hợp dữ liệu sai.
- Khuyến khích nâng cấp tiếp chương trình trên GITLAB.COM để xử lý các trường hợp dữ liệu sai và hoàn thiện phần mềm `sum`. Không cần thực hiện kiểm tra lỗi đầu vào, mà nên xử lý khi thực hiện tính toán: khi gặp ký tự không phải là ký số (digit) thì dừng và báo lỗi luôn.

Ghi chú:

- Đóng gói và phát hành (Release) chương trình `sum` lên Internet, kèm hướng dẫn để người dùng tải về, lưu vào thư mục mà họ muốn (ví dụ: D:\RunNow\). Sau đó người dùng có thể mở cửa sổ lệnh của hệ điều hành, đứng tại bất kỳ đường dẫn nào cũng có thể chạy được lệnh `sum`.
 - o Nếu gõ lệnh `sum` không có tham số thì hiển thị hướng dẫn.
 - o Nếu quá trình tính toán gặp dữ liệu không hợp lệ thì báo lỗi.

Chương trình 5

Viết chương trình `CountDate` để tính tổng số ngày giữa 2 ngày được truyền từ tham số dòng lệnh. Ví dụ gõ lệnh như sau:

```
CountDate 2019-12-31 2020-12-31
```

Có mục đích là đếm số ngày từ ngày 31 tháng 12 năm 2019 đến ngày 31 tháng 12 năm 2020.

Trường hợp không gõ tham số thứ hai như:

```
CountDate 2019-12-31
```

thì có mục đích là đếm số ngày từ ngày 31 tháng 12 năm 2019 đến ngày hiện tại của máy tính.

Tạm thời không cần xử lý các trường hợp ngoại lệ.

Bài 1 – Tại sao GO ra đời

Vào khoảng năm 2009, một nhóm chuyên gia của Google phát triển một dự án nội bộ tên là GO. GO được thiết kế để giúp các lập trình viên chuyên nghiệp tạo ra các phần mềm có tính ổn định, tin cậy và hiệu quả cao. Có thể xem GO là một hướng cải tiến của ngôn ngữ lập trình C cổ điển vốn rất mạnh nhưng kèm theo là rất phức tạp.

Bài 2: Ngôn ngữ lập trình GO

Trước khi đi vào ngôn ngữ lập trình, cụ thể là ngôn ngữ lập trình GOLANG (gọi ngắn gọn là GO) thì chúng nên biết vài khái niệm cơ bản về máy tính, về phần mềm. Đầu đó các khái niệm này có thể bạn đã học trong các lớp Tin học cơ bản, Nhập môn lập trình. Đây là cơ hội chúng ta ôn lại một chút.

Biến (Variable), Cấu trúc (Structure)

Variable là một cái tên dùng để chỉ một vùng nhớ trong máy tính. Để đơn giản, bạn hãy tưởng tượng cái máy vi tính giống như não người, trong đó có vùng nhớ (memory) để lưu thông tin tạm thời (lúc máy tính đang bật). Một variable được xem như một cái ô nhớ để đựng một giá trị nào đó.

Hình bên dưới là một thiết bị điện tử có trong máy tính của các bạn. Nó là một bản mạch gồm nhiều con chip có thể lưu trữ lại thông tin (bao gồm cả dữ liệu và lệnh) trong lúc máy tính có điện. Mọi người thường gọi ngắn gọn nó là thanh RAM.



Thanh RAM – nơi lưu "Trí nhớ" tạm thời của máy tính

Để các bạn hiểu hơn một chút về việc khai thác bộ nhớ của máy tính thì hãy tưởng tượng làm cách nào mà bạn yêu cầu cái máy tính của bạn nhớ thông tin của một người bạn thân gồm các thông tin như sau:

Tên	Lê Ngọc Thạch
Chiều cao	165 cao
Cân nặng	70.5 kg
Giới tính	Nam
Ngày sinh	29/9/1977
Các chữ số yêu thích	1, 2, 5, 10, 20, 50, 100
Các môn thể thao yêu thích	Bóng bàn, bóng đá, Quần vợt

(Bạn có thể thay bằng thông tin của chính mình cho chính xác hơn nhé!)

Mỗi thông tin ở cột bên trái được gọi là một **biến** (variable). Bạn tưởng tượng là trong thanh RAM ở phần trước có rất nhiều ô nhỏ li ti. Mỗi ô nhỏ như vậy máy tính (*cụ thể các phần mềm mà chúng ta sẽ thực hành ở phần tiếp theo*) được đặt cho một cái tên (name) – gọi là **tên biến** (variable name). Mỗi biến như vậy sẽ có một vùng nhớ khác nhau để chứa thông tin. Để đơn giản cho máy tính thì chúng ta nên sử dụng tên tiếng Anh để đặt cho tên biến.

Tên biến nên gồm các **kí tự chữ cái thường, chữ cái HOA, dấu gạch chân** (_) và có thể có kí số (ở giữa hoặc ở cuối tên biến). Để thống nhất cho các bạn khi thực hành thì tôi sử dụng quy trước theo thông lệ chung như sau:

- Tên biến bắt đầu bằng chữ thường.
- Kí tự Hoa và thường được hiểu là 2 kí tự khác nhau. Ví dụ tên biến là fullName sẽ khác với tên biến là FullName. Tức là có hai vùng nhớ khác nhau để chứa thông tin của 2 biến này.
- Tên biến phải ngắn gọn và gợi nghĩa.
- Khi tên biến gồm nhiều từ ghép lại (như Full name – 2 từ trong ví dụ trên) thì hãy viết Hoa kí tự của từ tiếp theo.

Để mô tả thông tin trong ví dụ trên thì chúng ta có thể tự quy định tên biến như bảng sau:

Thông tin	Tên biến
Tên	fullName
Chiều cao	height
Cân nặng	weight
Giới tính	sex
Ngày sinh	birthday
Các chữ số yêu thích	favorNumbers
Các môn thể thao yêu thích	favorSports

Trên đây là thông tin của một người, để mô tả thêm một người bạn nữa thì bạn phải làm sao?

Bạn có thể đặt thêm một loạt biến nữa như: fullName1, height1, ... Tức là bạn thêm số thứ tự phía sau để có bộ biến mới cho người mới. Tuy nhiên cách này không hay. Giới khoa học máy tính đưa ra khái niệm **Structure** để giúp các bạn giải quyết nhu cầu này.

Structure (cấu trúc) là một khái niệm gom nhiều loại thông tin để mô tả một vật, một người hay nói chung là một đối tượng nào đó. Nói cụ thể hơn là

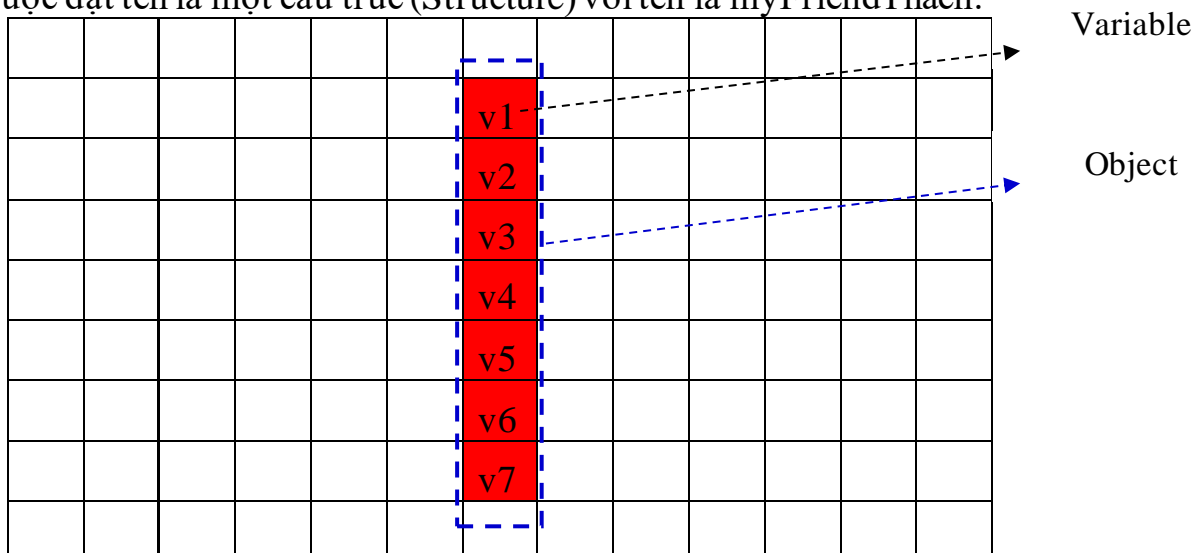
Structure sẽ chứa trong nó nhiều biến. Chúng ta mô tả lại ví dụ trình bày thông tin cho người bạn “Thạch” của chúng ta ở trên dưới dạng một Structure như sau:

Structure: myFriendThach	
fullName	Lê Ngọc Thạch
height	165 cm
weight	70.5 kg
sex	Nam
birthday	29/9/1977
favoriteNumbers	1, 2, 5, 10, 20, 50
favoriteSports	Bóng bàn, bóng đá, Quần vợt

Trong bảng trên xuất hiện từ **myFriendThach**, đây là một cái tên (name) được máy tính chỉ định (hoặc là **trở tới**) vùng nhớ của tất cả các thông tin về bạn Thạch.

Như vậy đến đây bạn biết được khái niệm biến (**variable**) là một cái tên (name) trở tới một vùng nhớ chứa thông tin cơ bản nào đó của bạn Thạch (như tên, cân nặng, v.v...). Toàn bộ các biến liên quan đến bạn Thạch được gom lại trong một vùng nhớ (đương nhiên là rộng hơn) gọi là **Structure**.

Hình minh họa bên dưới gồm 7 ô nhớ tương ứng với 7 biến để mô tả thông tin về bạn Thạch (kí hiệu v1 đến v7 tương ứng với fullName... favoriteSports). Hình chữ nhật màu xanh được bao gởi đường đứt nét được gọi là một vùng nhớ cũng được đặt tên là một cấu trúc (Structure) với tên là myFriendThach.



Hình 1: Minh họa khái niệm biến (Variable) và cấu trúc (Structure)

Variable có nghĩa là gì?

Tra tự điển

Nếu tra từ điển Oxford thì variable có thể là danh từ, có thể là tính từ.

☞ Tính từ variable: *able to be changed or adapted* (có thể được thay đổi hoặc điều chỉnh)

☞ Danh từ variable: *an element, feature, or factor that is liable to vary or change* (một yếu tố, một nét đặc trưng, hoặc một nhân tố có khả năng **biến đổi** hoặc **thay đổi**).

Cũng trong Oxford, variable được định nghĩa trong lĩnh vực Computing (điện toán) như sau: *a data item that may take on more than one value during the runtime of a program* (một phần tử dữ liệu có thể mang một hoặc hơn một giá trị trong suốt thời gian thực thi của chương trình).

Như vậy chữ variable có hai nghĩa mà các nhà khoa học máy tính và dịch giả Việt Nam đã dùng từ “biến” đã phản ánh đầy đủ rõ khái niệm “biến” trong máy tính.

Cụ thể là từ **vary** có hàm ý là có thể **biến đổi** thành đối tượng khác. Đối tượng khác ở đây có nghĩa là bản chất thông tin thay đổi hẳn. Chữ **change** có hàm ý là thay đổi giá trị của ô nhớ. Tức là bản chất, loại thông tin không thay đổi, mà chỉ thay đổi về nội dung, về giá trị của chúng.

Ví dụ:

Biến **height** đang có giá trị là 70.5 thì có thể được thay đổi thành một giá trị khác (tùy theo ngữ cảnh, thời gian như là đo lại tại một thời điểm khác) như là 71, 70 (chúng ta hiểu đơn vị là kg). Sự thay đổi này gọi là **change**.

Tuy nhiên, vì lý do nào đó trong ứng dụng phần mềm chúng ta muốn lưu trữ thông tin không phải là chiều cao nữa mà muốn lưu giá trị là một chức vụ cao nhất mà người đó đã từng làm. Tức là height sẽ được lưu giá trị là một **tên của chức vụ** (chứ không là một con số phản ánh chiều cao nữa). Lúc này biến height được biến đổi từ mục đích lưu con số phản ánh chiều cao thành một tên phản ánh chức vụ cao nhất. Cái này gọi là **vary** theo nghĩa trong từ điển Oxford.

Sau khi bạn hiểu được khái niệm Variable rồi thì câu hỏi tiếp theo là làm sao thiết lập giá trị cho biến. Cụ thể như thiết lập giá trị cho các ô nhớ từ v1 đến v2 trong hình 4.

Để làm được việc này thì bạn cần học thêm khái niệm gán (assign) trong phần tiếp theo.

Khai báo biến (variable declaration)

Trong ngôn ngữ lập trình GO, để khai báo một variable thì dùng cú pháp var như sau:

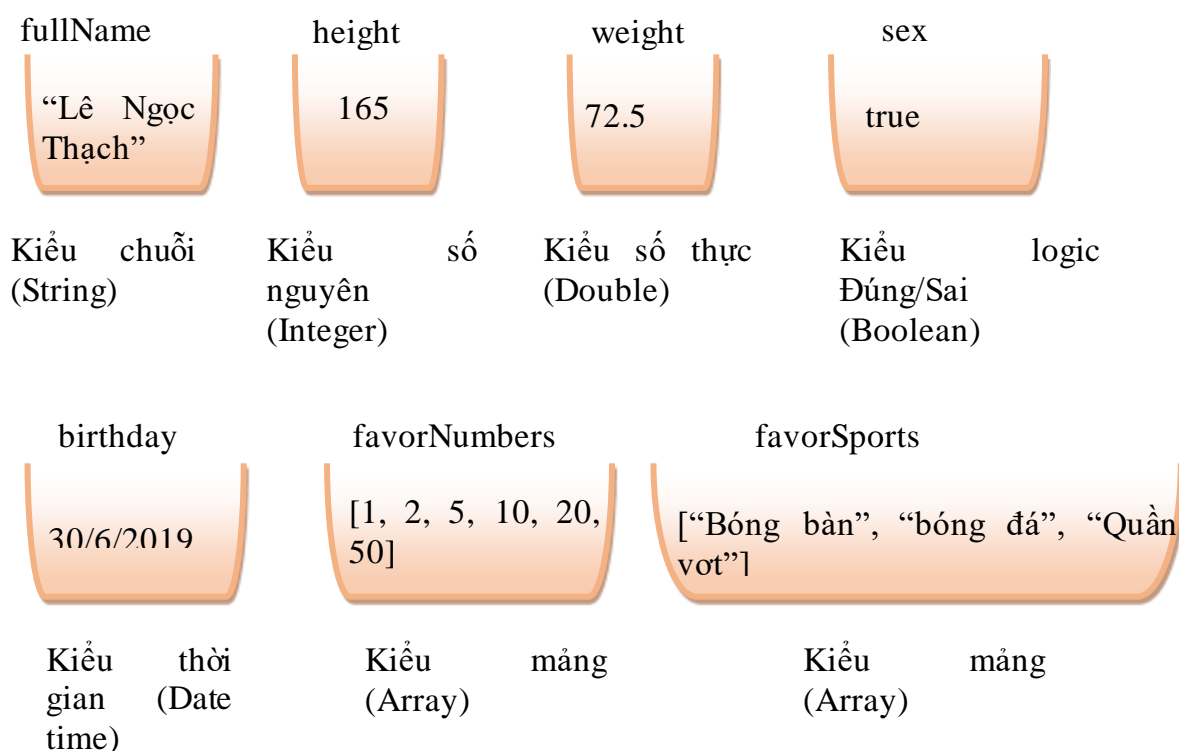
```
var <tên biến> <kiểu dữ liệu>
```

Ví dụ 2 dòng lệnh sau sẽ khai báo 2 vùng nhớ tương ứng cho `fullName`, `height` với kiểu dữ liệu tương ứng là `string` (chuỗi) và `int` (số nguyên)

```
var fullName string
var height int
```

Lệnh gán (assign)

Hình bên dưới minh họa các variable có tên `level`, `score`, `name`, `birthday` tương ứng với các ô nhớ (hãy xem như là một cái thùng) chứa bên trong nó các thông tin tương ứng.



Hình minh họa biến (variable)

Để thiết lập thông tin (hay còn gọi là dữ liệu) vào biến thì sử dụng phép gán (assign).

Cách 1 – Sử dụng cú pháp `var` và dấu `=`

Trong GO có thể vừa khai báo biến với từ khóa `var` và gán luôn giá trị cho biến với cú pháp là dấu `=` như sau:

```
var fullName string = "Lê Ngọc Thạch"
var height int = 165
```

Bạn cũng có thể không cần chỉ rõ kiểu dữ liệu, GO tự biết kiểu dữ liệu của biến với ví dụ sau:

```
var fullName = "Lê Ngọc Thạch"  
var height = 165
```

Cách 2 – Sử dụng cú pháp :=

Trong GO, có thể dùng dấu bằng := để thực hiện khai báo vùng nhớ và gán giá trị.

Ví dụ:

```
fullName := "Lê Ngọc Thạch"  
height := 165
```

Khi đã khai báo biến thì GO dùng dấu bằng "=" để thiết lập, hoặc thay đổi giá trị của biến.

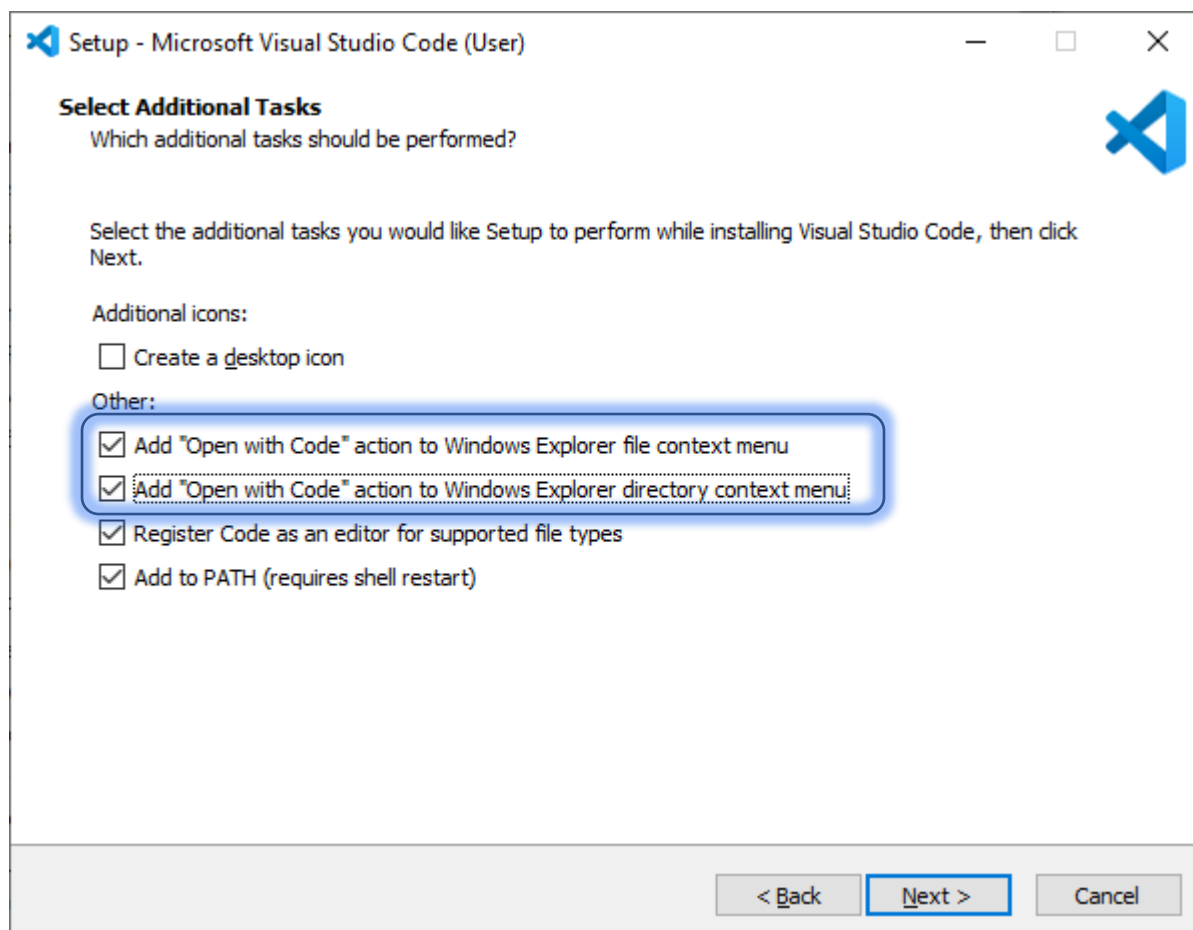
Bài 3 – Chuẩn bị môi trường lập trình

GO Core

Tải và cài gói phần mềm để biên dịch GO tại:

<https://golang.org/dl>

Chú ý lúc cài đặt thì hãy chọn 2 mục như hình bên dưới:



Cài thêm thư viện

Cộng đồng lập trình GOLANG thường đóng góp mã nguồn của mình dưới dạng các thư viện và công bố trên Internet. Ví dụ dự án này:

<https://github.com/jszwec/csvutil>

cung cấp tiện ích giúp bạn lập trình GOLANG với file CSV rất thuận tiện.

Để tải dự án này về và sử dụng chúng như là một gói phần mềm trong chương trình GOLANG của bạn thì gõ lệnh sau:

```
go get github.com/jszwec/csvutil
```

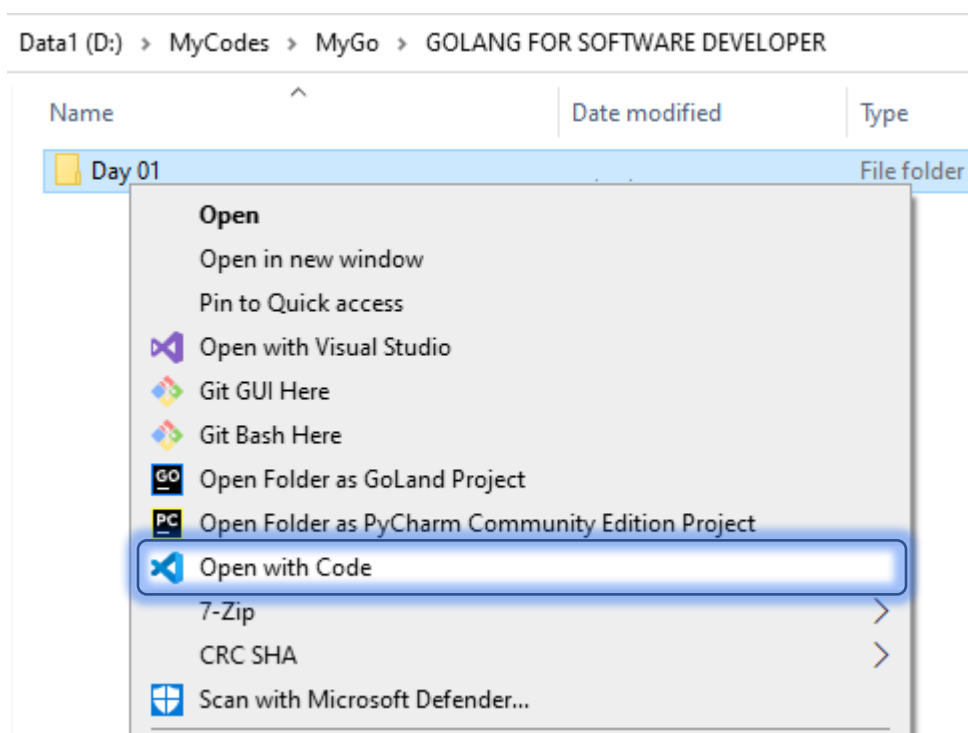

Visual Code

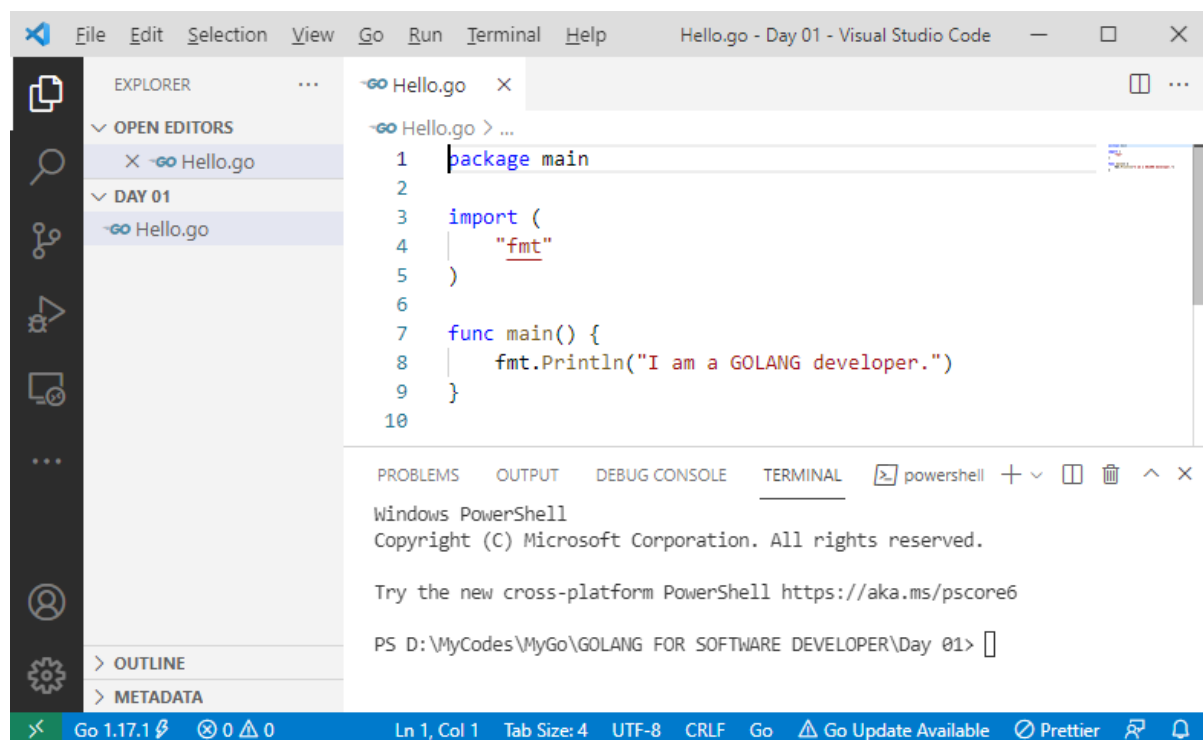
Một trong các công cụ lập trình gọn nhẹ, phổ biến hiện tại là Visual Code. Visual Code có nhiều phần mở rộng giúp cho việc phát triển dự án bằng ngôn ngữ GO dễ dàng.

Khởi động Visual Code

Mở Visual Code để bắt đầu công việc lập trình của bạn như sau:

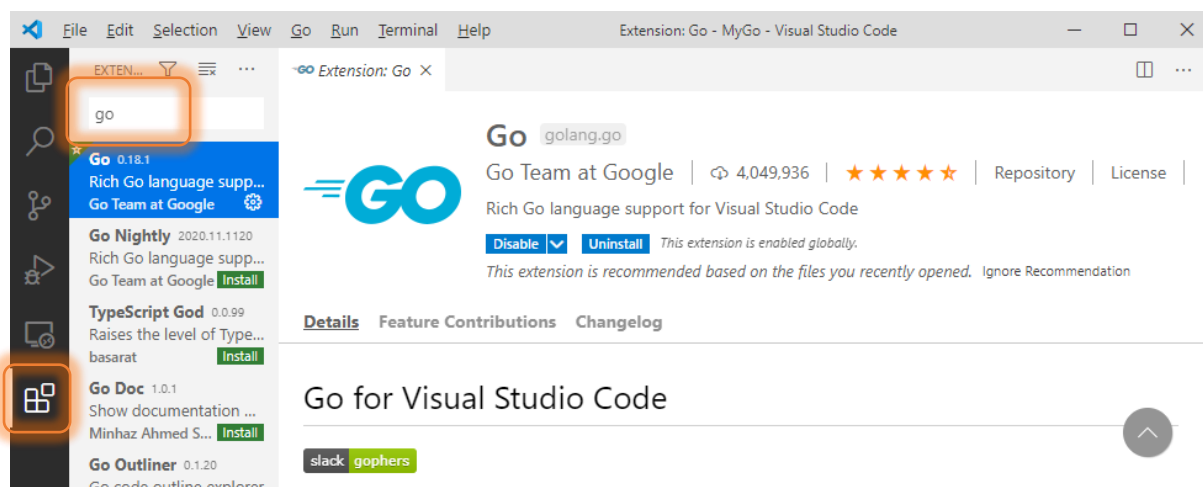
- Chuẩn bị sẵn thư mục chứa mã nguồn.
- Trong cửa sổ Windows Explorer, **click phải chuột vào thư mục**, chọn menu như sau:





Cài extensions cơ bản

Go for Visual Studio Code



Phím tắt trong Visual Code

Hãy học thêm các phím tắt để sử dụng trong Visual Code tại link sau:

<https://code.visualstudio.com/docs/languages/go>

Cài extentions nâng cao

Gọi là nâng cao thôi chứ thật ra cũng không phải cao gì đâu. Chỉ là các công cụ này có nhiều chức năng hay mà nếu bạn khai thác tốt thì giúp cải thiện đáng kể năng suất lập trình.

#	Từ khóa	Ghi chú	Link
1	docs-markdown		Trang chủ

Cài GO trên Ubuntu

Trong Ubuntu gõ lệnh sau để kiểm tra version của GO:

```
go version
```

Nếu chưa có GO thì sẽ ra hướng dẫn cài đặt. Ví dụ chạy lệnh sau:

```
sudo apt install gccgo-go
```

Bài 4 – Viết chương trình đơn giản với GO

Tạo thư mục D:\MyGo để chứa mã nguồn của các bài tập.

Khởi động Visual Code, nhấn tổ hợp phím Ctrl + K + O rồi chọn thư mục D:\MyGo.

Viết mã

Tạo file D:\MyGo\HelloGo.go với nội dung sau:

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello GO! Xin Chào GO nhé!")
}
```

Đoạn chương trình khai báo package là main ý muốn nói đoạn code phía sau được gọi để thực thi chương trình.

Đoạn chương trình trên sử dụng gói thư viện `fmt` bằng lệnh `import`.

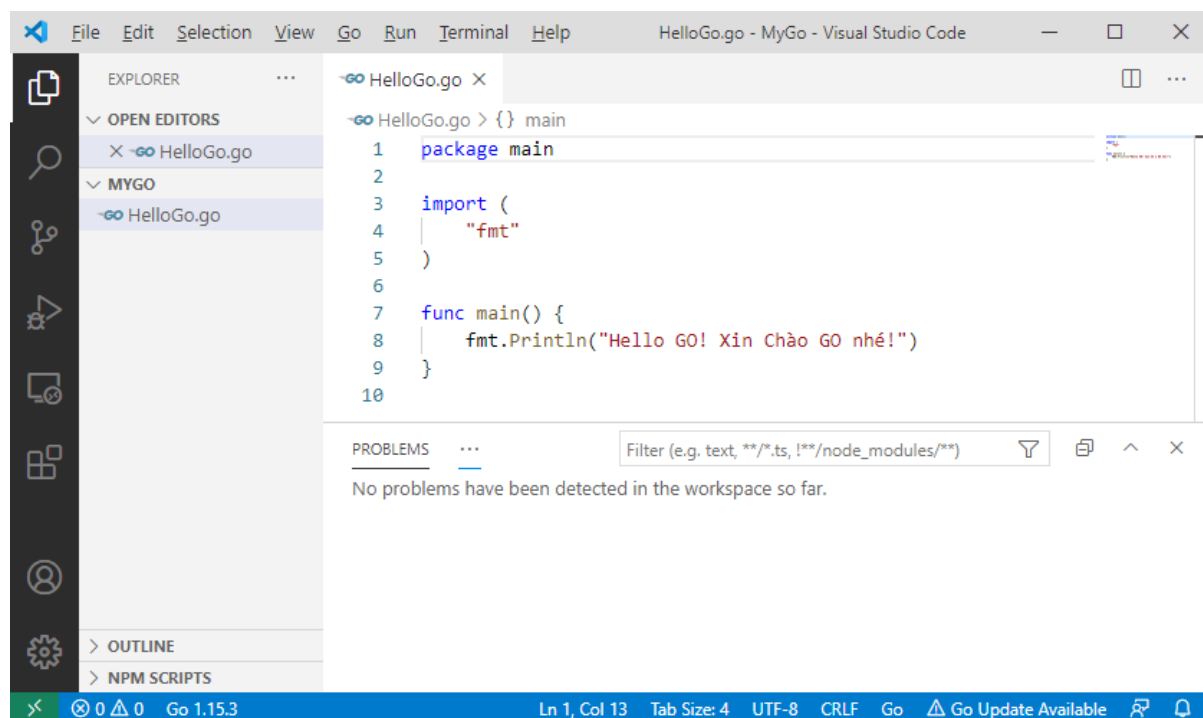
Khai báo hàm có tên là `main` là nơi bắt đầu của chương trình. Trong hàm `main` viết một lệnh đơn giản bằng cách gọi hàm `Println` trong gói thư viện `fmt` để hiển thị ra màn hình một câu chào (chuỗi đơn giản bao đóng bởi cặp dấu nháy đôi).

Biên dịch

Trong Visual Code nhấn phím Ctrl + Shift + ` (Thông thường phím ` là phím bên trái phím số 1, phía trên phím Tab) để mở dấu nhắc lệnh.

Trường hợp thư mục hiện hành không phải là D:\MyGo thì bạn thực hiện hai lệnh sau:

```
D:
cd D:\MyGo
```



Lệnh "go build HelloGo.go" sẽ biên dịch file mã nguồn HelloGo.go thành file mã máy HelloGo.exe. Cách gõ nhanh như sau:

Bước 1: Gõ

```
go build H
```

Bước 2: Nhấn phím tab

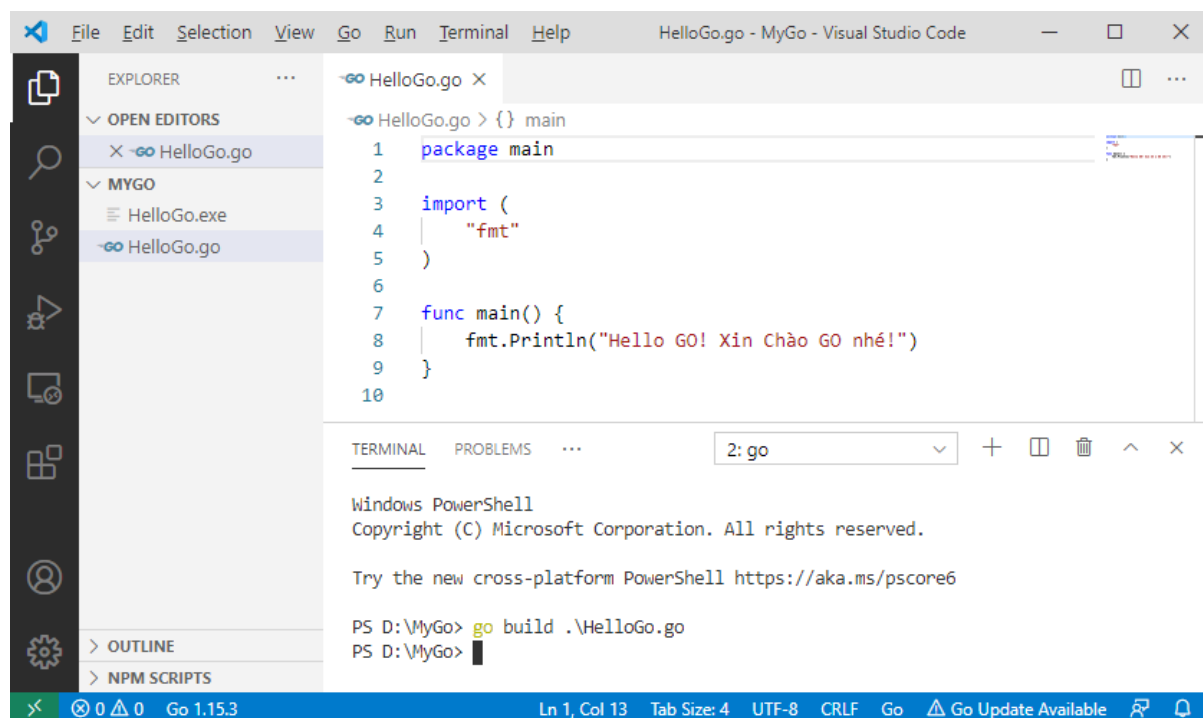
Cửa sổ lệnh sẽ tự động điền tiền file đầu đủ bắt đầu có chữ H. Trong trường hợp này là HelloGo.go. Kết quả lệnh đầy đủ là:

```
go build .\HelloGo.go
```

Kí hiệu dấu chấm có nghĩa là thư mục hiện hành. ".\HelloGo.go" có nghĩa là file HelloGo.go trong thư mục hiện hành.

Tiếp theo bạn gõ lệnh HelloGo.exe để chạy thử. Cách gõ nhanh tương tự như sau:

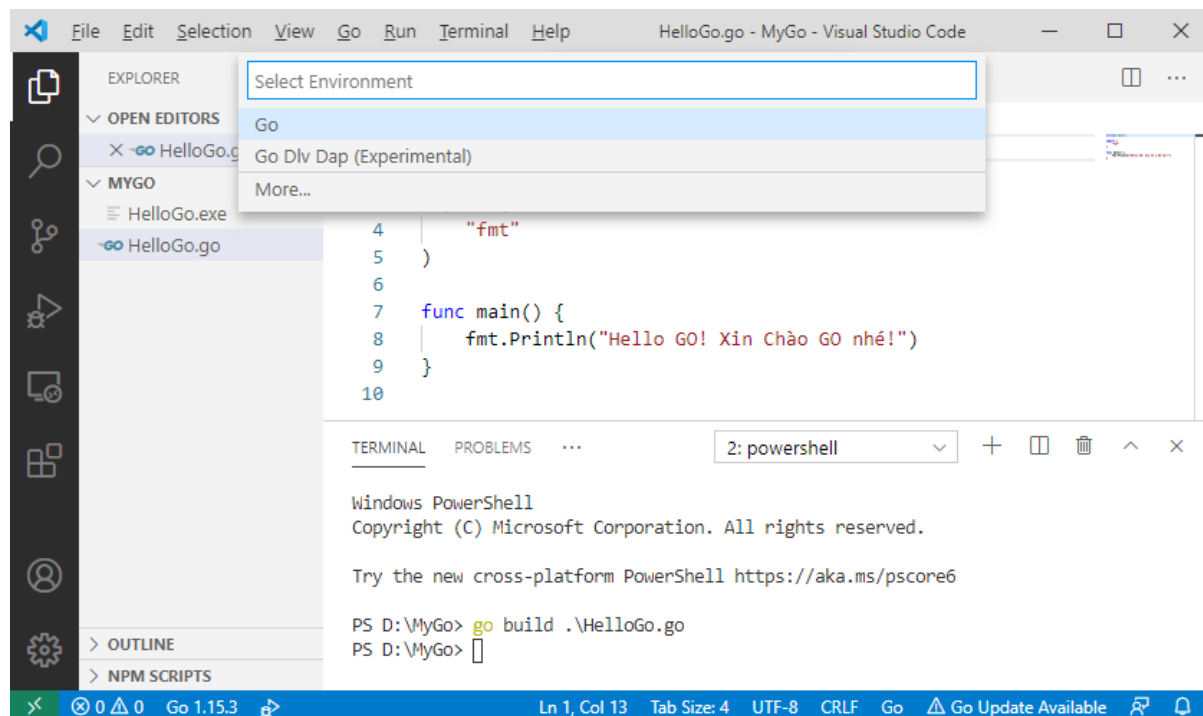
Bạn gõ chữ H rồi nhấn phím Tab, cửa sổ lệnh sẽ thông minh hiển thị sẵn cho mình lệnh .\HelloGo.exe. Xong nhấn Enter như hình bên dưới.

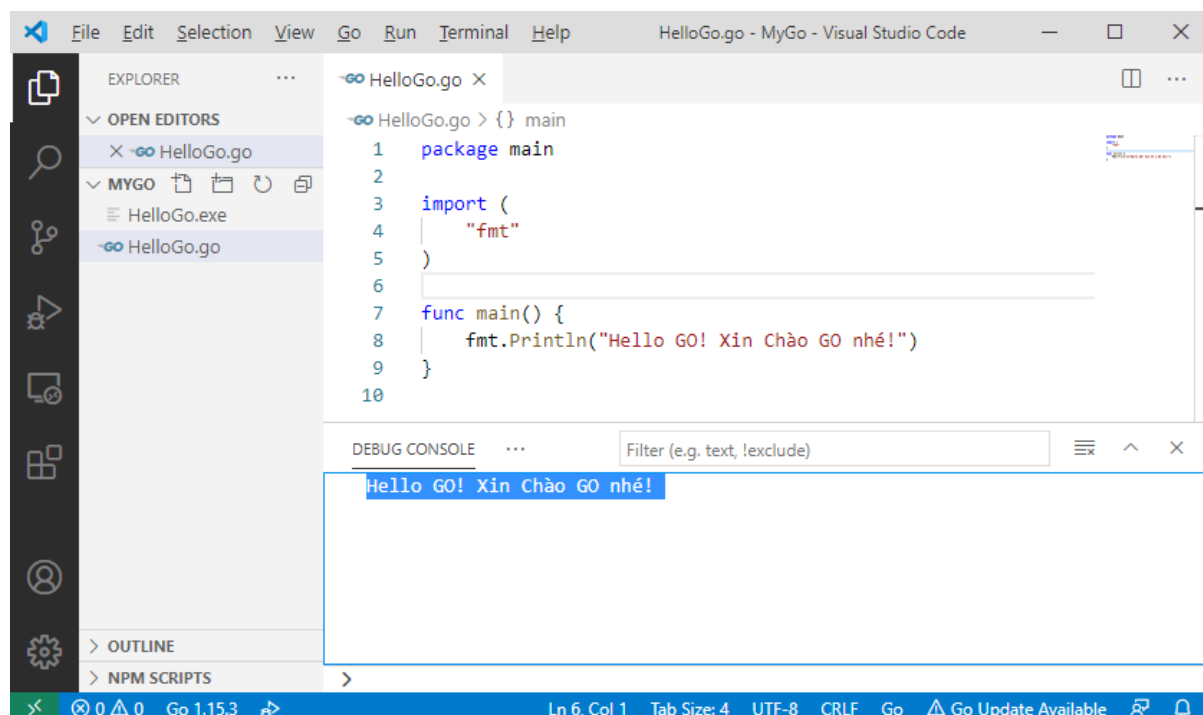


Chạy trực tiếp mã nguồn

Chạy trong Visual Code

Để chạy chương trình mã không cần phải gõ lệnh biên dịch như ở trên thì bạn nhấn tổ hợp phím `Ctrl + F5`. Một hộp thoại nhỏ yêu cầu chọn môi trường (Select Environment), chọn mục Go. Sau đó xem kết quả trong cửa sổ TERMINAL như bên dưới:





Trong trường hợp mã nguồn của bạn có sử dụng thư viện bên ngoài thì bạn phải thiết lập module và download thư viện trước khi chạy bằng 2 lệnh sau:

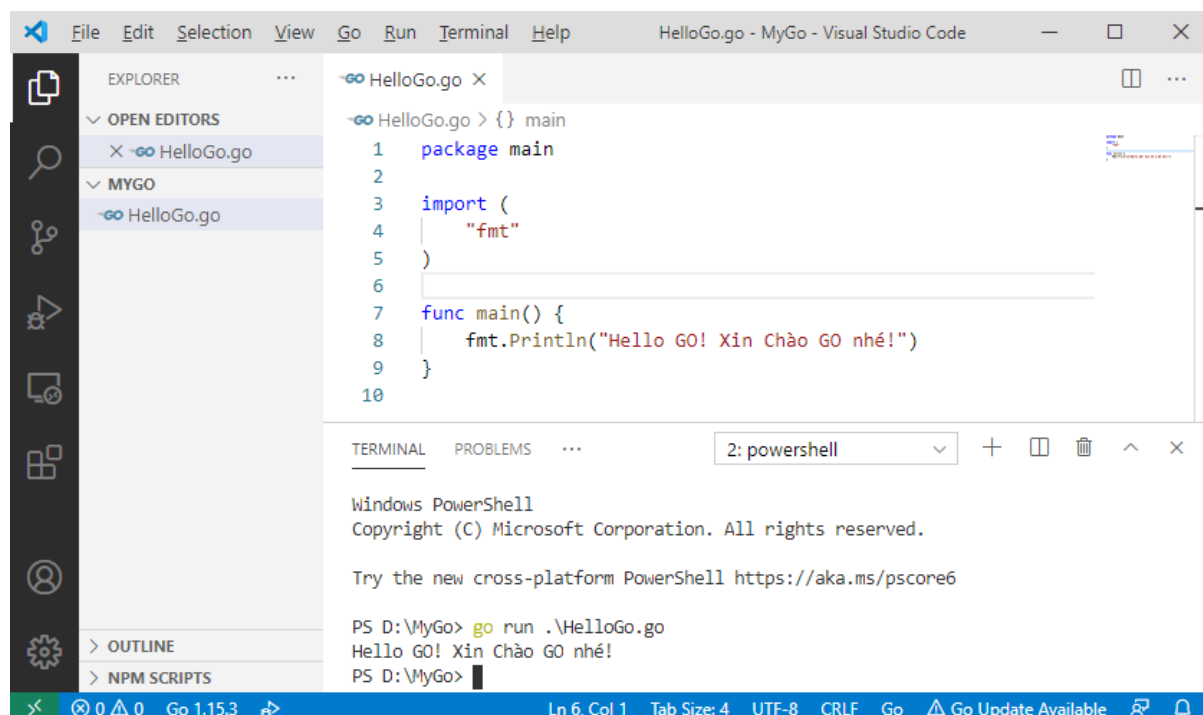
```
go mod init myapp
go mod tidy
```

Chạy bằng gõ lệnh

Bạn có thể gõ lệnh ngoài chương trình Command của Windows, gọi tắt là console; hoặc bạn có thể mở Terminal trong Visual Code bằng phím tổ hợp phím **Ctrl + Shift + `**.

Lệnh để chạy file mã nguồn HelloGo.go:

```
go run HelloGo.go
```



```
File Edit Selection View Go Run Terminal Help
HelloGo.go - MyGo - Visual Studio Code

EXPLORER
OPEN EDITORS
HelloGo.go
MYGO
HelloGo.go

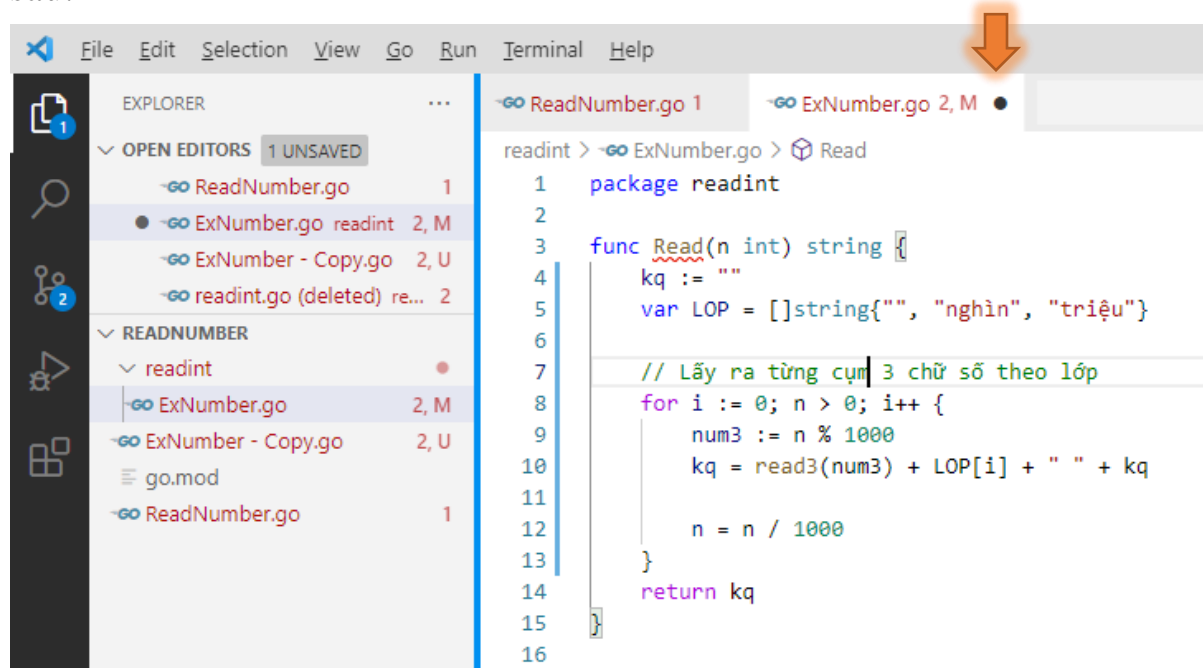
HelloGo.go
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     fmt.Println("Hello GO! Xin Chào GO nhé!")
9 }
10

TERMINAL
2: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\MyGo> go run .\HelloGo.go
Hello GO! Xin Chào GO nhé!
PS D:\MyGo>
```

Chú ý là bạn phải lưu file trước khi gõ lệnh. Dấu hiệu để bạn biết là file nguồn được sửa và chưa lưu là có dấu chấm tròn đen bên phải tên file như hình sau:



```
File Edit Selection View Go Run Terminal Help
ReadNumber.go 1 ExNumber.go 2, M
readint > ExNumber.go > Read
1 package readint
2
3 func Read(n int) string {
4     kq := ""
5     var LOP = []string{"", "nghìn", "triệu"}
6
7     // Lấy ra từng cụm 3 chữ số theo lớp
8     for i := 0; n > 0; i++ {
9         num3 := n % 1000
10        kq = read3(num3) + LOP[i] + " " + kq
11
12        n = n / 1000
13    }
14    return kq
15 }
16
```

Phép gán (assign)

Cú pháp :=

Phần trước bạn đã biết cách viết một đoạn chương trình nhỏ để hiển thị ra màn hình một câu đơn giản. Thử cải tiến một chút để làm quen với khai báo biến name và phép gán với kí hiệu dấu bằng:

```
package main
```



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

Đoạn chương trình trên sử dụng cú pháp := để vừa khai báo biến vừa thiết lập giá trị cho nó. Tôi tạm gọi cú pháp := là **gán khai báo**.

Kết quả chương trình sẽ hiển thị ra chuỗi:

```
Hello  Thạch
```

Cú pháp =

Chạy thử đoạn chương trình sau:

```
package main  
  
import (  
    "fmt"  
)  
  
func main() {  
    name := "Thạch"  
    fmt.Println("Hello ", name)  
  
    name = "Lê Ngọc " + name  
    fmt.Println("Họ và tên: ", name)  
}
```

Bạn học thêm từ đoạn chương trình trên:

Sử dụng phép gán với cú pháp = để thay đổi giá trị của biến name bằng cách ghép nó với một chuỗi vào phía bên trái. Trong tài liệu này khi nói phép gán tức là dùng dấu =. Khi nói **phép gán khai báo** thì dùng hai chấm bằng := nhé!

Các toán tử cơ bản

Các phép toán số học (Arithmetic Operator)

Phép toán	Ý nghĩa	Ví dụ
+		
-		
*		
/	Chia lấy phần nguyên	
%	Chi lấy phần dư	
++		
--		

Các toán tử so sánh (Relational Operator)

Phép toán	Ý nghĩa	Ví dụ
==		
!=		
>		
>=		
<		
<=		

Các toán tử logic (Logical Operators)

Phép toán	Ý nghĩa	Ví dụ
&&		
!		

Hàm (function)

Khái niệm hàm trong lập trình là cách để người lập trình chia nhỏ một chương trình lớn thành các đoạn chương trình nhỏ hơn. Các chương trình nhỏ này có thể được tái sử dụng nhiều lần trong các tình huống khác nhau bằng cách thay đổi các thông số đầu vào.

Để viết hàm thì dùng cú pháp như sau:

```
func tên_hàm(tham_số) kết_quả_trả_về {  
}
```

Khảo sát chương trình có hàm tính toán tuổi như sau:

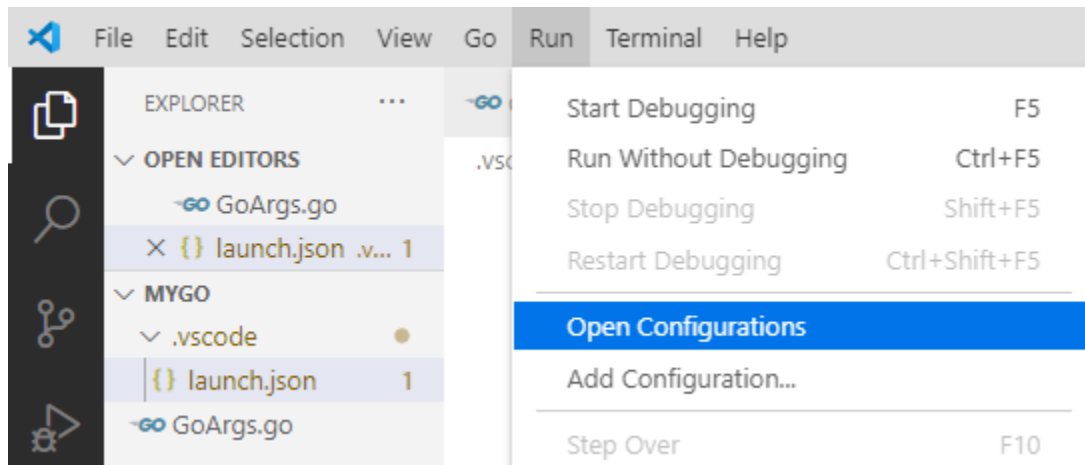
```
package main  
  
import (  
    "fmt"  
)  
  
func calAge(birthYear int) int {  
    return 2020 - birthYear  
}  
  
func main() {  
  
    myAge := calAge(1977)  
  
    fmt.Println(myAge)  
}
```

Vài điểm chú ý về khai báo hàm:

- Từ khóa: **func**
- Sau từ khóa **func** là **tên hàm**. Ví dụ: **calAge**
- Tiếp theo tên hàm là cặp dấu ngoặc, trong cặp dấu ngoặc là tham số. Kiểu dữ liệu của tham số được viết bên phải của tên tham số. Ví dụ: **birthYear int**
- Sau dấu ngoặc **)** kết thúc phần tham số là kiểu dữ liệu trả về của hàm. Trong ví dụ này là kiểu **int**.
- Nội dung của hàm được viết trong cặp dấu ngoặc nhọn **{ }**
- Tên hàm nên dùng tiếng Anh; có thể viết tắt; và nên bắt đầu là động từ. Tại thời điểm này thì bạn dùng chữ cái đầu tiên trong tên hàm là chữ Hoa hay chữ thường đều được.

Chạy chương trình có tham số dòng lệnh trong Visual Code

Để chạy code GO trong Visual Code và truyền các tham số từ dòng lệnh thì bạn cần cấu hình một chút. Cụ thể là bạn vào menu Run > Open Configurations



Lần đầu tiên bạn vào menu này thì Visual Code sẽ cài đặt thêm một vài thứ. Bạn chỉ cần ngồi theo dõi là được.

Sau đó Visual Code sẽ tự tạo file launch.json trong thư mục làm việc của bạn. Bạn tìm đến dòng bên dưới để thêm các tham số:

```
"args": []
```

Bạn điền tham số vào giữa cặp dấu ngoặc, các tham số bao đóng bởi cặp dấu nháy đôi và cách nhau bởi dấu phẩy. Ví dụ

```
"args": ["Hải", "2000"]
```

Lựa file và quay lại file mã nguồn để chạy lại.

Lấy tham số từ dòng lệnh

Khảo sát mã nguồn của file D:\MyGo\GoArgs.go như sau:

```
package main

import (
    "fmt"
    "os"
)

func main() {
    fmt.Println("Number of arguments:", len(os.Args))
}
```

```
    fmt.Println("The first argument:", os.Args[0])
}
```

Biên dịch chương trình bằng lệnh

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

Chú ý phần bên trái "D:\MyGo>" ý nói đường dẫn thư mục hiện hành chứ không phải là nội dung của lệnh

Thực thi chương trình bằng cách gõ lệnh `GoArgs.exe`:

```
D:\MyGo> GoArgs.exe
```

```
Number of arguments: 1
The first argument: D:\MyGo\GoArgs.exe
PS D:\MyGo>
```

Kiến thức học được:

- Sử dụng thuộc tính `Args` trong thư viện `os` để lấy ra danh sách các tham số trên dòng lệnh.
- Phần tử đầu tiên của `os.Args` là đường dẫn của chương trình đang chạy.

Hãy thử chạy luôn mã nguồn mà không cần biên dịch bằng lệnh:

```
go run .\GoArgs.go
```

Vòng lặp (loops)

Thử chạy đoạn code sau để khám phá cú pháp vòng lặp `for`. Ngoài ra học thêm cách sử dụng dấu phẩy để ngăn cách tham số trong lệnh `fmt.Println()`:

Chương trình `EvenNum.go`:

```
package main

import (
    "fmt"
)

func main() {
    for i := 0; i < 10; i++ {
        fmt.Println("i x 2 = ", i*2)
    }
}
```

Nâng cao

Sử dụng Logging

Trong các bài trước để ghi các lỗi, hoặc các thông tin để theo dõi chương trình chạy ra màn hình thì các bạn dùng hàm `Println` trong thư viện `fmt`. Việc hiển thị các thông tin theo dõi, hoặc các lỗi như thế này gọi chung là quá trình truy vết (tracing) hoặc logging. Có vài vấn đề liên quan đến việc sử dụng hàm `Print`, `Println`, hoặc `Printf` trong tình huống này, nói chung là không hợp lý cho các dự án thực tế. Cụ thể:

- Các lỗi này phải được kết xuất ra một nơi nào đó (như file, gọi chung là logging file; hoặc database) để có thể phân tích sau này.
- Các lệnh truy vết như thế này phải đảm bảo không làm ảnh hưởng lớn đến tốc độ thực thi của chương trình. Ví dụ một lệnh `fmt.Println` đơn giản cũng phải tốn thời nhất định (dù rất ít).

Để thực hiện truy vết, theo dõi chương trình chạy thì trong thực tế người ta sử dụng các thư viện logging. Một trong các thư viện logging cho ngôn ngữ GO là thư viện `logrus` này:

```
github.com/sirupsen/logrus
```

Để cài đặt thư viện `logrus` thì thực hiện lệnh sau trong dấu nhắc lệnh của hệ điều hành.

```
go get github.com/sirupsen/logrus
```

Sử dụng logging cho chương trình `EvenNum.go`:

```
package main

import (
    "fmt"

    log "github.com/sirupsen/logrus"
)

func main() {
    log.SetLevel(log.DebugLevel)
    log.Debug("Hiển thị các số chẵn từ 0 đến 10...")
    for i := 0; i < 10; i++ {
        fmt.Println(i, " x 2 = ", i*2)
    }
}
```

Diễn giải:

- Ba dòng bôi vàng có ý nghĩa theo trình tự như sau:
 - Khai báo sử dụng thư viện `logrus` với alias là `log`

- Lệnh `log.SetLevel (...)` để thiết lập mức độ log là Debug. Ngoài ra còn nhiều cấp độ khác nữa với các mục đích khác nhau. Coi như đây là bài tập cho bạn.
- Lệnh `log.Debug (...)` để kết xuất thông tin ra hệ thống lưu trữ log. Hệ thống lưu trữ mặc định ở đây là màn hình console của ứng dụng.

Để chạy được chương trình trên thì bạn cần thực hiện các lệnh sau để cài đặt thư viện cho ứng dụng:

```
go mod init myapp
go mod tidy
```

Sau đó rồi mới chạy chương trình bằng lệnh sau:

```
go run .\EvenNum.go
```

Kết quả như sau:

```
time="2021-10-18T10:27:20+07:00" level=debug msg="Hiển thị các số chẵn
từ 0 đến 10..."
0 x 2 = 0
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
```

Diễn giải:

- Dòng đầu tiên là kết xuất của lệnh `log.Debug (...)`. Lệnh này dùng để lưu vết chương trình nhằm giúp cho việc điều tra chương trình sau này. Thông thường thì các lệnh log này sẽ không liên quan đến việc hiển thị thông tin về chức năng hiển thị của chương trình (tức nó không liên quan đến giao diện của người dùng). Sau này sẽ có cách để cấu hình các kết quả log này vào file riêng, hoặc cơ sở dữ liệu, không ảnh hưởng đến màn hình kết quả của ứng dụng.
- Các lệnh `fmt.Print...` là các lệnh liên quan đến chức năng của chương trình: chương trình yêu cầu hiển thị thông tin ra màn hình.

Bài 5 – Biểu diễn thông tin đơn giản với GO

Câu hỏi chung cho những ai mới học lập trình là làm sao biểu diễn được các thông tin để máy tính hiểu được. Cụ thể trong ngữ cảnh eBook này là làm sao biểu diễn các thông tin cơ bản với ngôn ngữ GO.

Trong bài này chúng ta sẽ học các loại thông tin cơ bản, thường dùng sau:

- String
- Numeric
- Go arrays
- Go slices
- Go maps
- Go pointer
- Times & dates

Kiểu chuỗi (string)

Trong bài 4, bạn đã làm quen với một chương trình đơn giản là hiển thị một câu ra màn hình. Câu này được bao đóng trong cặp dấu nháy đôi như:

"Đây là một chuỗi các kí tự"

Lấy ra một kí tự của chuỗi

Khảo sát đoạn code sau:

```
package main

import (
    "fmt"
)

func main() {
    st := "I can do it"

    fmt.Println(st[0])
    fmt.Printf("%c", st[0])
}
```


73

I

Như vậy cú pháp `st[0]` sẽ lấy ra kí tự đầu tiên của chuỗi nhưng giá trị trả lại là một số nguyên. Giá trị này chính là giá trị mã kí tự.

Để hiển thị ra màn hình dạng kí tự của mã thì dùng hàm `fmt.Printf` với định dạng là `%C`.

Xem kiểu dữ liệu của biến

Khảo sát đoạn code sau để biết cách xem kiểu dữ liệu của biến bằng lệnh `fmt.Printf` với tham số `%T`

```
package main

import (
    "fmt"
)

func main() {
    st := "I can do it"

    fmt.Println(st[0])
    c := st[0]
    fmt.Printf("%T", c)
}
```

uint8

Kiểu dữ liệu số (Numeric data types)

Số nguyên (Integer)

GO hỗ trợ 4 kiểu dữ liệu số nguyên không dấu tương ứng với số byte đi kèm như: `int8`, `int16`, `int32`, `int64`

Và 4 kiểu dữ liệu số nguyên có dấu: `uint8`, `uint16`, `uint32`, `uint64`. (`uint` là viết tắt của `unsigned integer`, số nguyên không dấu)

Thêm vào đó có 2 kiểu dữ liệu không ghi rõ số byte được sử dụng: `int` và `uint`. Kích thước (số byte) cho kiểu `int` và `uint` này tùy thuộc vào kiến trúc phần cứng của máy tính và hệ điều hành và phần mềm dùng để lập trình của bạn.

Dưới đây là bảng các giá trị nhỏ nhất và lớn nhất

uint8	0 ~ 255
--------------	----------------

uint16	0 ~ 65535
uint32	0 ~ 4294967295
uint64	0 ~ 18446744073709551615
int8	-128 ~ 127
int16	-32768 ~ 32767
int32	-2147483648 ~ 2147483647
int64	-9223372036854775808 ~ 9223372036854775807

Vài ví dụ để bạn tự khám phá:

Ví dụ 1:

```
package main

import "fmt"

func main() {
    var n uint8

    n = 255

    fmt.Println(n)

    n = n + 1

    fmt.Println(n)
}
```

Ví dụ 2:

```
package main

import "fmt"

func main() {
    var n uint8

    n = 255

    fmt.Println(n)

    n = n + 1

    fmt.Println(n)

    // New

    n = n - 1

    fmt.Println(n)
}
```

```
}
```

Ví dụ 3:

```
package main

import "fmt"

func main() {

    var n int8

    n = 127

    fmt.Println(n)
    n = n + 1

    fmt.Println(n)

    n = n - 1

    fmt.Println(n)
}
```

Trong GOLANG, một kí tự được xem như là một số nguyên. Khác với các ngôn ngữ lập trình khác có kiểu char. Hãy khảo sát đoạn chương trình sau:

```
package main

import (
    "fmt"
    "reflect"
)

func main() {

    ch := 'a'
    fmt.Println(ch)
    fmt.Printf("%c", ch)
    fmt.Println()
    fmt.Println(reflect.TypeOf(ch))

    ch = 'A'
    fmt.Println(ch)

    ch = '0'
    fmt.Println(ch)

    ch = '1'
    fmt.Println(ch)
}
```

Kết quả:

```
97
a
int32
65
48
49
```

Như vậy kí tự ‘a’, ‘A’, ‘0’, ‘1’ có giá trị nguyên lần lượt là: 97, 65, 48, 49.

Để hiển thị kí tự ra màn hình tương ứng với giá trị nguyên của nó thì cùng kí hiệu %c trong hàm fmt.Printf. Hãy thử lệnh sau:

```
fmt.Printf("%c", 98)
```

Để hiển thị các biến số nguyên ra màn hình trong lệnh fmt.Print(..) thì dùng kí hiệu %d trong mẫu lệnh sau:

```
fmt.Printf("...%d...%d", i1, i2)
```

Số thực (floating-point numbers)

GO hỗ trợ 2 kiểu dữ liệu để biểu diễn số thực: float32 và float64.

Vài ví dụ để bạn trải nghiệm với số thực bằng cách dự đoán kết quả của các lệnh fmt.Println và chạy lại chương trình để đúc kết kinh nghiệm.

Ví dụ 1:

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var f32 float32 = 1.2
    var f64 float64 = 1.3
    fmt.Println(f32)
    fmt.Println(f64)
    fmt.Println(math.MaxFloat32)
    fmt.Println(math.MaxFloat64)
}
```

Ví dụ 2:

```
package main

import "fmt"

func main() {
    n := 2
```

```
m := 3
result := n / m
fmt.Println(result)
}
```

Ví dụ 3:

```
package main

import "fmt"

func main() {
    n := 2.0
    m := 3.0
    result := n / m
    fmt.Println(result)
}
```

Kết quả:

0.6666666666666666

Để hiển thị giá trị của các biến số thực ra màn hình trong các lệnh `fmt.Printf()` thì dùng 2 dạng sau:

- ① `fmt.Printf("...%.mf", f1)`
- ② `fmt.Printf("...%.nf...%n.mf", f1)`

m trong 2 lệnh trên là con số cụ thể cho biết số lượng số lẻ trong phần thập phân cần hiển thị ra màn hình.

n.m trong lệnh có ý nghĩa như sau:

- ✓ Hiển thị m số lẻ trong phần thập phân
- ✓ Hiển thị thêm khoảng trắng phía trước sao cho tổng cộng (số khoảng trắng + số ký tự phần nguyên + dấu chấm + số ký tự phần thập phân) là n ký tự

Hãy tự trải nghiệm các đoạn chương trình sau.

Ví dụ 1: Sử dụng hàm `fmt.Printf(...)` để hiển thị số nguyên và số thực.

```
package main

import "fmt"

func main() {
    fmt.Printf("Năm %d là năm chẵn\n", 2020)
    fmt.Printf("Kết quả của 2 chia cho 3: %f", 2.0/3.0)
}
```

Ví dụ 2: Sử dụng hàm `fmt.Println(...)` hoặc `fmt.Print()` để hiển thị số thực.

```
package main

import "fmt"
```

```
func main() {  
    n := 2.0  
    m := 3.0  
    result := n / m  
    fmt.Println(result)  
}
```

Ví dụ 3:

```
package main  
  
import "fmt"  
  
func main() {  
    fNumber := 34567.123456789  
    fmt.Printf("Hiển thị số thập phân mặc định: %f", fNumber)  
    fmt.Println()  
    fmt.Printf("Hiển thị số thập phân với 2 số lẻ:\n%.2f", fNumber)  
    fmt.Println()  
    fmt.Printf("Hiển thị số thập phân với 2 số lẻ và thêm khoảng trắng vào  
phía trước cho đủ 10 kí tự:\n%10.2f", fNumber)  
}
```

Ví dụ 4: Tính giá trị của e^x với x chạy từ 0 đến 7. Sau đó in ra kết quả với lề được anh phải.

```
package main  
  
import (  
    "fmt"  
    "math"  
)  
  
func main() {  
    for x := 0; x < 8; x++ {  
        fmt.Printf("x = %d, e^%d = %8.3f\n", x, x, math.Exp(float64(x)))  
    }  
}
```

Kết quả:

```
x = 0, e^0 = 1.000  
x = 1, e^1 = 2.718  
x = 2, e^2 = 7.389  
x = 3, e^3 = 20.086  
x = 4, e^4 = 54.598  
x = 5, e^5 = 148.413  
x = 6, e^6 = 403.429  
x = 7, e^7 = 1096.633
```

Viết chương trình Fibonacci

Đến đây bạn đã biết các kí hiệu để biểu diễn các thông tin cơ bản dạng số, và vòng lặp. Bây giờ hãy thực hành một chút bằng cách viết một chương trình hiển thị ra dãy Fibonacci.

Quy tắc của dãy số Fibonacci đơn giản được áp dụng trong bài này như sau:

1 2 3 5 8 13 21 34 55 89 144...

Cho 2 số đầu tiên là 1 và 2. Số tiếp theo được tính bằng cách cộng 2 số liền kề phía trước.

Áp dụng các kiến thức đã học để viết chương trình Fibonacci bên dưới:

- Sử dụng cú pháp **gán khai báo** := để khai báo biến và gán dữ liệu luôn mà không cần nói rõ kiểu dữ liệu. Cụ thể là khai báo hai biến cho 2 số bên liền kề là n và m với n là 0; m là 1.
- Sử dụng vòng lặp for 10 bước với biến đếm là nCount
- Sử dụng lệnh Print trong thư viện fmt với 2 tham số: số fibonacci tiếp theo, và dấu cách.

```
package main

import (
    "fmt"
)

func main() {
    n := 0
    m := 1
    p := n + m

    for nCount := 0; nCount <= 10; nCount++ {

        fmt.Print(p, " ")
        n = m
        m = p
        p = n + m
    }
}
```

Kết quả:

1 2 3 5 8 13 21 34 55 89 144

Mảng (arrays)

Bạn đã làm quen với kiểu dữ liệu số, vòng lặp và viết được chương trình Fibonacci. Bây giờ mở rộng kiến thức với kiểu mảng nhé!

Mảng 1 chiều

Khám phá đoạn chương trình sau để biết cách khai báo mảng 1 chiều, ghi rõ số phần tử là 4 và liệt kê giá trị 4 phần tử là 1, 2, 3, 4.

```
package main

import "fmt"

func main() {
    arrayA := [4]int{1, 2, 3, 4}

    fmt.Println(arrayA, " len =", len(arrayA))
}
```

Kết quả:

```
[1 2 3 4] len = 4
```

Bạn tự đoán ý nghĩa của hàm `len(array)` nhé, `len` là viết tắt của `length` (độ dài).

Duyệt mảng 1 chiều bằng cú pháp `range`

```
package main

import "fmt"

func main() {
    arrayX := [5]int{1, 2, 3, 4, 5}
    for _, number := range arrayX {
        fmt.Print(number, " ")
    }
}
```

Lấy một phần của mảng tại từ một vị trí

Khảo sát đoạn code sau để biết cách dùng cú pháp `[i:]` và `[i:j]` của mảng

```
package main

import "fmt"

func main() {
    arrayX := [5]int{1, 2, 3, 4, 5}

    fmt.Println(arrayX[2:])
    fmt.Println(arrayX[2:4])
}
```

Kết quả:

```
[3 4 5]
```


[3 4]

Kinh nghiệm học được:

- `a[i:]` sẽ trả lại mảng con tính từ phần tử tại vị trí `i`. Trong ví dụ trên phần tử có vị trí 2 (chú ý vị trí bắt đầu từ 0) là 3.
- `a[i:j]`: sẽ trả lại mảng con tính từ phần tử tại vị trí `j` cho đến phần tử ở vị trí **trước** `j`. Chú ý trước `j` tức là không bao gồm phần tử tại vị trí `j`.

Mảng 2 chiều (2 dimensions-array)

Khám phá đoạn code sau để hình dung cách khai báo và thiết lập mảng 2 chiều, hay còn gọi là ma trận (matrix). Code minh họa là ma trận gồm 3 dòng và 2 cột.

```
package main

import "fmt"

func main() {
    twoD := [3][2]int{
        {1, 2},
        {3, 4},
        {5, 6}
    }

    fmt.Println(twoD, " len =", len(twoD))
}
```

Slice – Mảng không giới hạn độ dài

Ý tưởng chính của slides là:

- Được sử dụng như là array nhưng không cố định độ dài
- Kích thước của slice được mở rộng tự động
- Khi slice được sử dụng như là tham số của một hàm thì nó được truyền kiểu tham chiếu (passed by reference). Tức là các thay đổi slide bên trong hàm thì sau khi kết thúc hàm thì giá trị slice được thay đổi theo. Nếu bạn chưa quen khái niệm hàm, tham số dạng tham chiếu thì không sao, tạm thời chưa quan tâm đến nó nhé!

Quan sát đoạn code sau để thấy sự khác biệt khi khai báo và thiết lập giá trị giữa slice và array.

```
package main

import "fmt"

func main() {

    arrayA := [4]int{1, 2, 3, 4}
    fmt.Println(arrayA, " len =", len(arrayA))

    sliceA := []int{5, 6, 7, 8}
    fmt.Println(sliceA, " len =", len(sliceA))

}
```

Tạo slice với hàm make

Đoạn code sau sẽ tạo slice gồm 5 phần tử, mỗi phần tử mặc định có giá trị là 0.

```
package main

import "fmt"

func main() {

    intSlice := make([]int, 5)

    fmt.Println("Số phần tử của slice ", len(intSlice))

}
```

Duyệt các phần tử của slice hoặc array

```
package main

import "fmt"

func main() {

    intSlice := make([]int, 5)
```

```
fmt.Println("Số phần tử của slice ", len(intSlice))

for i := 0; i < len(intSlice); i++ {
    fmt.Println(intSlice[i])
}
}
```

Thêm phần tử vào slice

```
package main

import "fmt"

func main() {

    intSlice := []int{1, 2, 3, 4, 5}

    intSlice = append(intSlice, 6)

    fmt.Println("intSlice = ", intSlice)
}
```

Truy xuất các phần tử của slice

Để truy cập 1 phần tử của slice hoặc array thì sử dụng cú pháp `[i]` với `i` là số thứ tự của phần tử, bắt đầu từ 0.

Khảo sát đoạn code sau để khám phá cú pháp `[i: j]`:

```
package main

import "fmt"

func main() {

    intSlice := []int{5, 6, 7, 8, 9, 10}

    fmt.Println("Lấy các phần tử từ vị trí 1 đến 3 = ", intSlice[1:3])
    fmt.Println("Lấy các phần tử từ vị trí 0 đến 3 = ", intSlice[0:3])
}
```

Kết quả:

```
Lấy các phần tử từ vị trí 1 đến 3 = [6 7]
Lấy các phần tử từ vị trí 0 đến 3 = [5 6 7]
```

Phân tích:

- Cú pháp `s[i: j]` bao gồm phần tử tại `i` nhưng **không** bao gồm phần tử tại `j`.

Trích slice con

Khi cần trích xuất từ đầu slice đến một vị trí nào đó trong slice hoặc trích xuất từ vị trí nào đó trong slice đến phần tử cuối cùng thì dùng lại cú pháp `[i: j]` nhưng không chỉ định `i` hoặc `j` như sau:

- `s[: j]` để trích xuất các phần tử trong slice `s` từ đầu đến vị trí `j`
- `s[i:]` để trích xuất các phần tử trong slice `s` từ vị trí `i` đến cuối slice

Đoạn code sau sẽ trích các phần tử từ vị trí thứ 2 (bao gồm nó) đến cuối slice `intSlice`:

```
subSlice := intSlice[2:]  
  
fmt.Println("subSlice=", subSlice)
```

```
subSlice= [7 8 9 10]
```

Xóa một phần tử của slice

Để xóa một phần tử của slice tại vị trí `i` thì dịch chuyển các phần tử từ vị trí `i + 1` lên phía trước. Hình minh họa bên dưới khi bạn cần xóa phần tử thứ 2:

0	1	2	3	4	5	6	7
0	1	3	4	5	6	7	

Cách làm:

- Trích xuất ra slice con từ đầu đến vị trí cần xóa. Dùng cú pháp `s[: i]`
- Trích xuất ra slice con từ vị trí `i + 1` đến cuối slice. Dùng cú pháp `s[i+1:]`
- Ghép 2 slice đã trích xuất ở trên. Dùng hàm `append(s1, s2)`

Code minh họa

```
package main  
  
import "fmt"  
  
func main() {  
  
    intSlice := []int{0, 1, 2, 3, 4, 5, 6, 7}  
  
    i := 2  
    newSlice := append(intSlice[:2], intSlice[i+1:]...)  
  
    fmt.Println("newSlice=", newSlice)  
}
```

Tham khảo thêm hàm `append` tại:

```
https://golang.org/pkg/builtin/#append
```

Dung lượng và Kích thước của slice

Slice có 2 thuộc tính quan trọng là capacity và length.

Hàm `len(slice)` cho biết số phần tử thật đang có của slice.

Hàm `cap(slice)` sẽ cho biết khả năng lưu trữ của slice. Bạn hình dung là GO chuẩn bị sẵn bộ nhớ để có thể lưu trữ các phần tử mới.

Hãy chạy và quan sát kết quả đoạn code sau để khám phá kết quả của hàm `cap(slice)` sau khi được thêm 1 phần tử nhé!

```
package main

import "fmt"

func main() {
    aSlice := []int{1, 2, 3}
    fmt.Println("aSlice: ", aSlice, "; len= ", len(aSlice), " ;cap=", cap(aSlice))

    // Thêm 1 phần tử
    aSlice = append(aSlice, 4)
    fmt.Println("aSlice: ", aSlice, "; len= ", len(aSlice), " ;cap=", cap(aSlice))
}
```

Kết quả:

```
aSlice: [1 2 3] ; len= 3 ;cap= 3
aSlice: [1 2 3 4] ; len= 4 ;cap= 6
```

Bạn có rút ra được điều gì không?

Slice 2 chiều

Tương tự như mảng 2 chiều thì slice 2 chiều được minh họa trong ví dụ sau:

```
package main

import "fmt"

func main() {
    aSlice := [][]int{
        {1, 2, 3},
        {4, 5},
    }
    fmt.Println("aSlice: ", aSlice, "; len= ", len(aSlice))
}
```

```
aSlice: [[1 2 3] [4 5]] ; len= 2
```

Chú ý kích thước của dòng 1 và dòng 2 là khác nhau. Bạn tự rút ra nhận xét nhé.

Thử so sánh kết quả với đoạn code sau minh họa mảng 2 chiều gồm 2 dòng và 3 cột như sau:

```
package main

import "fmt"

func main() {
    aSlice := [2][3]int{
        {1, 2, 3},
        {4, 5},
    }
    fmt.Println("aSlice: ", aSlice, "; len= ", len(aSlice))
}
```

```
aSlice: [[1 2 3] [4 5 0]] ; len= 2
```

Chuyển slice thành array

Khảo sát đoạn chương trình sau:

```
package main

import "fmt"

func main() {

    intSlice := []int{0, 1, 2, 3, 4, 5, 6, 7}

    var arr [8]int

    copy(arr[:], intSlice[:])
    fmt.Println(arr)

    copy(arr[:], intSlice)
    fmt.Println(arr)
}
```

Hai dòng lệnh `copy(dst, src)` ở trên có kết quả như nhau:

```
[0 1 2 3 4 5 6 7]
[0 1 2 3 4 5 6 7]
```

Tạm dừng việc làm quen kiểu dữ liệu `slice` ở đây. Còn nhiều điều thú vị về `slice` sẽ được giải thích trong tài liệu nâng cao nhé!

Maps

Kiểu Maps dùng để biểu diễn một bảng dữ liệu gồm có 2 cột Key và Value.

Ví dụ một bảng Map đơn giản ánh xạ 1 số nguyên thành 1 chữ (Text):

Key	Value
1	"Một"
2	"Hai"
...	

Cú pháp khai báo

```
map[key_type]value_type
```

key_type: là kiểu dữ liệu của khóa

value_type: là kiểu dữ liệu của giá trị

Ví dụ

Khảo sát đoạn code sau để khám phá kiểu Maps

```
package main

import "fmt"

func main() {

    numberMap := map[int]string{
        1: "Một",
        2: "Hai",
    }

    for key, value := range numberMap {
        fmt.Println(key, value)
    }

    fmt.Println(numberMap[1])
}
```

```
2 Hai
```

1 Một
Một

Khởi tạo Map rỗng

Để tạo một bảng Map rỗng thì dùng lệnh make theo cú pháp chung như sau:

```
make (map [key_type] value_type)
```

Ví dụ tạo bảng Map có khóa là chuỗi, giá trị là số nguyên:

```
package main

import (
    "fmt"
    "reflect"
)

func main() {
    iMap := make(map[string]int)

    fmt.Printf("Number of items: %d\n", len(iMap))
    fmt.Printf("Data type of iMap: %s", reflect.TypeOf(iMap))
}
```

Kết quả:

```
Number of items: 0
Data type of iMap: map[string]int
```

Thời gian (Times & dates)

Khám phá đoạn code sau để làm quen với thư viện time:

```
package main

import (
    "fmt"
    "time"
)

func main() {
    fmt.Println("Epoch time:", time.Now().Unix())
    t := time.Now()
    fmt.Println(t, t.Format(time.RFC3339))
    fmt.Println(t.Weekday(), t.Day(), t.Month(), t.Year())
}
```



```
time.Sleep(time.Second)

t1 := time.Now()

fmt.Println("Time difference:", t1.Sub(t))

}
```

```
Epoch time: 1605330249
2020-11-14 12:04:09.5610308 +0700 +07 m=+0.001000301 2020-11-14T12:04:0
9+07:00 Saturday 14 November 2020
Time difference: 1.0013609s
```

Chuyển một chuỗi ngày tháng năm thành biến thời gian

Khi có nhu cầu chuyển một chuỗi “31/12/1980” với ý nghĩa là ngày 31 tháng 12 năm 1980 thành đối tượng kiểu thời gian thì cần sử dụng hàm Parse của thư viện time. Hãy thử đoạn chương trình sau:

```
package main

import (
    "fmt"
    "reflect"
    "time"
)

func main() {
    strDate := "31/12/1980"

    // "2006-01-02" since that is the yyyy-mm-dd formatting of the magical
    reference date
    myDate, err := time.Parse("02/01/2006", strDate)
    if err != nil {
        fmt.Println(err)
    } else {
        fmt.Println(myDate)
    }

    fmt.Println("Type of myDate: ", reflect.TypeOf(myDate))
}
```

Kết quả hiện thị như sau:

```
1980-12-31 00:00:00 +0000 UTC
Type of myDate: time.Time
```

Vài nhận xét

- GOLANG sử dụng định dạng 02/01/2006 cho tham số thứ nhất trong hàm time.Parse như là định dạng kiểu ngày như dd/mm/yyyy. Ý nói 02 có

nghĩa là ngày, 01 có nghĩa là tháng và 2006 có nghĩa là năm.
Để trải nghiệm thêm thì hãy thử sửa lại 2 lệnh sau:

```
strDate := "12/31/1980"  
myDate, err := time.Parse("01/02/2006", strDate)
```

Kết quả sẽ hiển thị ra đúng Năm là 1980, tháng là 12, ngày là 31 như sau:

1980-12-31 00:00:00+0000 UTC

00:00:00 là không giờ, không phút, không giây thì các bạn hiểu.

+0000 UTC là gì chưa hiểu thì thôi tạm bỏ qua nhé!

Tra cứu định dạng

Tra cứu các định dạng về ngày trong các ngôn ngữ GO, Java và C

Cú pháp GO	Cú pháp Java	Cú pháp C	Ghi chú
2006-01-02	yyyy-MM-dd	%F	ISO 8601
20060102	yyyyMMdd	%Y%m%d	ISO 8601
January 02, 2006	MMMM dd, yyyy	%B %d, %Y	
02 January 2006	dd MMMM yyyy	%d %B %Y	
02-Jan-2006	dd-MMM-yyyy	%d-%b-%Y	
01/02/06	MM/dd/yy	%D	US
01/02/2006	MM/dd/yyyy	%m/%d/%Y	US
010206	MMddyy	%m%d%y	US
Jan-02-06	MMM-dd-yy	%b-%d-%y	US
Jan-02-2006	MMM-dd-yyyy	%b-%d-%Y	US
06	yy	%y	
Mon	EEE	%a	
Monday	EEEE	%A	
Jan-06	MMM-yy	%b-%y	

Tra cứu các định dạng về giờ trong các ngôn ngữ GO, Java và C




Cú pháp GO	Cú pháp Java	Cú pháp C	Ghi chú
15:04	HH:mm	%R	
15:04:05	HH:mm:ss	%T	ISO 8601
3:04 PM	K:mm a	%l:%M %p	US
03:04:05 PM	KK:mm:ss a	%r	US

Tra cứu các định dạng về ngày giờ trong các ngôn ngữ GO, Java và C

Go layout	Java notation	C notation	Notes
2006-01-02T15:04:05	yyyy-MM-dd'T'HH:mm:ss	%FT%T	ISO 8601
2006-01-02T15:04:05-0700	yyyy-MM-dd'T'HH:mm:ssZ	%FT%T%z	ISO 8601
2 Jan 2006 15:04:05	d MMM yyyy HH:mm:ss	%e %b %Y %T	
2 Jan 2006 15:04	d MMM yyyy HH:mm	%e %b %Y %R	
Mon, 2 Jan 2006 15:04:05 MST	EEE, d MMM yyyy HH:mm:ss z	%a, %e %b %Y %T %Z	RFC 1123 RFC 822

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 Email và sdt của người nhận eBook.</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

Bài tập cuối ngày 1

Sử dụng GITLAB.COM để tạo dự án và viết các chương trình sau:

Chương trình ①: Viết chương trình tên là GoNameYear nhận 2 tham số từ dòng lệnh. Tham số thứ nhất là Tên, tham số thứ hai là Năm sinh. Ví dụ:

GoNameYear.exe Hải 2000

Chương trình sẽ hiển thị

Chào Hải 20 tuổi

Chương trình ②: Mở rộng chương trình trên bằng cách hiển thị thêm một thông báo dạng như sau:

Chào Hải, từ năm bạn sinh ra (năm 2000) đến bây giờ có các năm chia hết cho 4 gồm: 2000, 4004, ...

(Phần ... là thông tin bạn phải liệt kê đầy đủ).

Chương trình ③: Mở rộng chương trình trên bằng cách hiển thị thêm một thông báo dạng như sau:

Ngày 2 – Cách viết một phần mềm đơn giản

Bài 1 – Phần mềm đầu tiên – Cài đặt phép toán cộng

Phần này sẽ giúp các bạn hình dung rõ hơn toàn bộ quá trình làm phần mềm theo công thức IPO.

Bước 1: Xác định và viết yêu cầu

Công việc phần này là bạn gặp Khách hàng để hiểu rõ Khách hàng muốn gì. Hoặc đơn giản hơn bạn chính là Khách hàng của dự án này. Để có thể hình dung ra được bối cảnh thì tôi tưởng tượng ra tình huống như sau để các bạn thực hành.

Tình huống

Bạn gặp một nhà đầu tư D yêu cầu bạn phát triển dự án “Cộng hai số lớn” với yêu cầu cụ thể như sau:

- Chương trình cho phép các học sinh tiểu học thực hiện được phép tính tổng hai số lớn (nói chung là cực kỳ lớn, về mặt lý thuyết là có thể dài vô tận).
- Ngoài chức năng nói trên thì các sản phẩm bàn giao phải có đầy đủ tài liệu mô tả Yêu cầu (Requirement), mô tả Thiết kế (Design), mô tả Kiểm thử mã nguồn² (Unit Testing), Kết quả Đánh giá mã nguồn (Checklist, Code Report).
- Mã nguồn phải trong sáng, dễ hiểu, dễ bảo trì. Tuân thủ theo chuẩn lập trình GOLANG. Tham khảo tài liệu của Google và cộng đồng GOLANG tại:
 - o <https://go.dev/blog/godoc>
 - o <https://github.com/smallnest/go-best-practices>
 - o <https://github.com/bahlo/go-styleguide>

Với yêu cầu cho dự án như trên của nhà đầu tư D thì bạn bắt tay vào công việc như thế nào?

Xác định High Level Requirement

Sau khi đã nghe nhà đầu tư D (xem như là Khách hàng của bạn) nói rõ yêu cầu như vậy thì việc đầu tiên là bạn mô tả lại tất cả nội dung đấy vào một văn bản gọi là High Level Requirement (Tài liệu yêu cầu mức cao hoặc Tài liệu yêu cầu tổng thể). Trong một số tình huống thì Khách hàng đã viết sẵn High Level

² Dịch sát nghĩa là Kiểm thử đơn vị (Unit Testing) nhưng tôi thích dùng Kiểm thử mã nguồn nghe xuôi tai hơn cho các bạn học nhập môn lập trình.

Requirement cho mình rồi. Vì thế tôi dùng từ “Xác định” cho mục này là vậy. Phần dưới đây là ngữ cảnh bạn giúp Khách hàng viết luôn High Level Requirement cho dự án.

Bạn có thể dùng phần mềm MS Word để trình bày. Tuy nhiên để tiện cho bạn theo dõi tài liệu này thì tôi sẽ dùng Excel để trình bày tài liệu. Cái tiện lợi của Excel là có nhiều sheets để trình bày nhiều loại tài liệu trong cùng một file. Bạn có thể lấy file tại link:

[https://github.com/mksgroup/myworkspace/blob/master/ebooks/jp/chapter-1/A2N High-level-requirement vi.xlsx](https://github.com/mksgroup/myworkspace/blob/master/ebooks/jp/chapter-1/A2N%20High-level-requirement%20vi.xlsx)

Biểu mẫu Excel để viết High Level Requirement gồm có 2 sheets.

Sheet thứ nhất “Record of change” dùng để ghi lại lịch sử thay đổi tài liệu.
Sheet thứ hai “Requirement” ghi lại các yêu cầu của dự án.

Template_High level requirement.xlsx - Excel

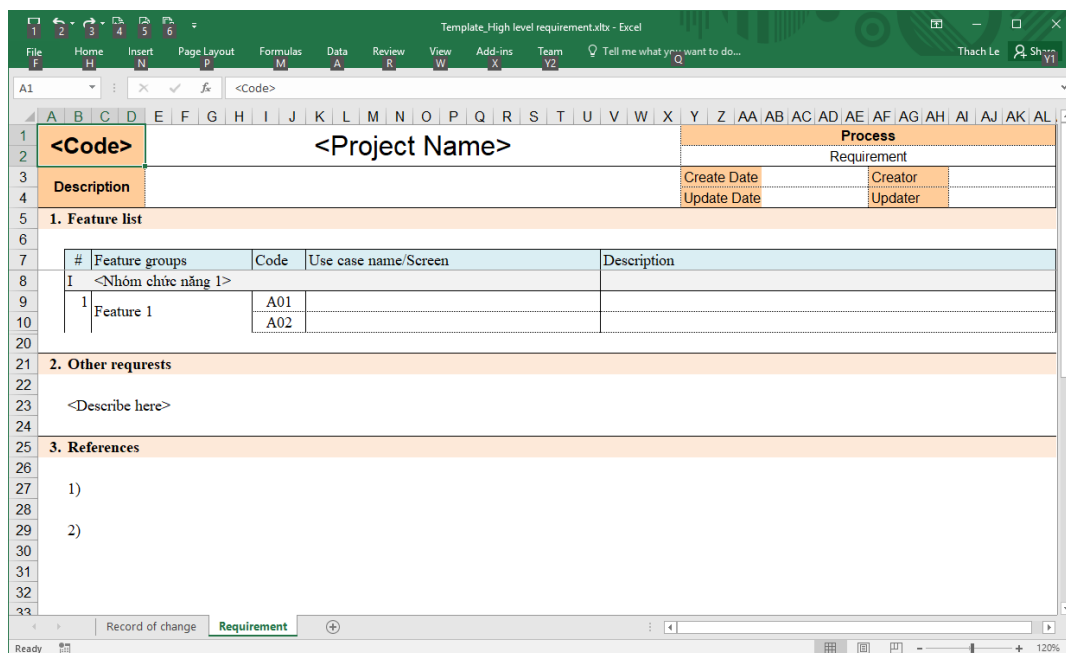
File Home Insert Page Layout Formulas Data Review View Add-ins Team Tell me... Thach Le Share

H17

	A	B	C	D	E
1					
2	*A - Add, M - Modify, D -Delete				
3					
4	Date	Changed item	A/M/D*	Description	Version
5	21-May-2017		A	The first version	0.
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					

Record of change Requirement

Ready



Sau khi viết đầy đủ thì được tài liệu High Level Requirement cho dự án Cộng hai số lớn như sau:

Requirement				
A2N	Cộng hai số lớn			Process
Description	Phần mềm cộng hai số lớn			Requirement
	Create Date	20-May-18	Creator	ThachLN
	Update Date		Updater	
1. Feature list				
Đây là phần mô tả chức năng cho phần mềm Cộng hai số lớn. Chương trình này giúp cho học sinh tiểu học hiểu và thực hiện từng bước của phép toán cộng.				
#	Feature groups	Code	Use case name/Screen	Description
I	<Nhóm chức năng 1>			
1	Feature 1	A01	Cộng hai số lớn	Chỉ hỗ trợ số nguyên dương.
		A02	Ghi lại các bước để thực hiện phép toán	Nhằm giúp cho người dùng hình dung được các bước cần thực hiện.
2. Other requests				
Sản phẩm bàn giao có đầy đủ:				
① Tài liệu mô tả Yêu cầu (Requirement)				
② Tài liệu mô tả Thiết kế (Design)				
③ Tài liệu mô tả Kiểm thử đơn vị (Unit Testing)				
④ Tài liệu báo cáo đánh giá mã nguồn (Review code report)				
3. References				
1) Tham khảo tài liệu Coding convention của Oracle				
2) Tham khảo tài liệu Coding convention của Google.				

Ghi nhớ

Việc đầu tiên là phải mô tả lại yêu cầu tổng thể của Khách hàng vào một tài liệu gọi là High Level Requirement.

Viết Software Requirement Specification

Bộ tài liệu thứ hai mà bạn³ cần viết là Software Requirement Specification, gọi tắt là SRS. Vì phần mềm này khá đơn giản và để bạn tập trung vào việc làm quen với lập trình thì tạm thời bỏ qua phần này.

³ Tôi dùng từ “Bộ” để bạn hình dung trong trường hợp phần mềm phức tạp thì có nhiều loại tài liệu, nhiều dạng (format) tài liệu khác nhau để làm nên Software Requirement Specification.

Bước 2: Làm thiết kế

Nghe đến từ Thiết kế cho phần mềm thì bạn có thể khóp và lo lắng không biết viết như thế nào. Phần này tôi sẽ giúp các bạn làm quen dần với cách trình bày Design. Việc đầu tiên là bạn cần luyện cách trình bày thuật toán cho dự án “Cộng hai số lớn” này.

Trước khi bạn học các kỹ năng, các công cụ nghe có vẻ cao siêu trong ngành Công nghệ phần mềm thì tôi chắc là các bạn có thể diễn đạt được ý tưởng, cách làm của mình bằng tiếng mẹ đẻ.

Bạn có thể vừa đọc phần này vừa thực hành bằng cách lấy giấy bút ra trình bày theo hướng dẫn:

Cho trước thông tin đầu vào gồm có hai số 123456 và 7890. Bạn hãy mô tả lại từng bước bạn cộng như thế nào. Nếu được thì hãy khái quát hoá lên thành phương pháp (gọi là Thuật toán). Ví dụ:

$$\begin{array}{r} 123456 \\ + \\ 7890 \\ \hline \end{array}$$

Tôi tin chắc là bạn có thể tự viết được cách làm như sau:

Thuật toán phiên bản 1

Bước 1: Lấy ra số cuối cùng của chuỗi 1 (số 6)

Bước 2: Lấy ra số cuối cùng của chuỗi 2 (số 0)

Bước 3: Thực hiện Cộng 2 số vừa lấy ($6 + 0 = 6$)

Bước 4: Ghi nhận kết quả là 6 (vì nhỏ hơn 10 nên không phải nhớ. Nếu kết quả Bước 3 lớn hơn 10 thì ghi nhận kết quả là phần đơn vị.

Lặp lại Bước 1 đến Bước 4 cho số bên trái tiếp theo. Cụ thể trong lần lặp thứ 2:

Bước 1: Lấy ra số 5

Bước 2: Lấy ra số 9

Bước 3: Thực hiện $5 + 9$ bằng 14

Bước 4: Vì 14 lớn hơn 10 nên lấy số 4 ghi vào kết quả (ghi vào bên trái của kết quả đang có là 6). Kết quả mới là 46. Phần chục trong số 14 được ghi nhớ (nhớ 1).

Lần lặp thứ 3:

Ghi nhớ

Để trình bày thuật toán thì bước đầu tiên hãy viết bằng lời như những gì bạn nói với người đối diện.

Bước 1: Lấy ra số 4

Bước 2: Lấy ra số 8

Bước 3: Thực hiện phép cộng

Thực hiện $4 + 8$ và cộng thêm nhớ 1 của lần lặp thứ 2. Kết quả là $4 + 8 + 1 = 13$.

Bước 4: Cập nhật kết quả

Lấy số 3 ghép vào bên trái của 46 ta được 346. Và tiếp tục nhớ 1.

Lần lặp thứ 4, 5, 6 sẽ tương tự như vậy. Tôi chắc là bạn tự làm được. Ở đây tôi chỉ ghi chú rõ thêm trường hợp chuỗi số hai khi đã lấy hết số (ở bước lặp thứ 5) thì số được lấy ra xem như là zero (0).

Viết ra giấy các bước này không phải là khó nhưng cũng không phải là dễ vì các bạn có thể nghĩ là nó quá quen thuộc. Nhưng nếu bạn không ghi ra giấy hoặc ít ra là nghĩ ra một cách rõ ràng trong đầu thì khả năng bạn làm sai trong các bước tiếp theo là rất lớn.

Sau khi đã có văn bản ở trên thì bạn đã hình dung được cách làm như thế nào? Tuy nhiên, bạn có thể cải tiến văn bản ở trên cho gọn hơn thành một văn bản có thể xem như là thuật toán.

Thuật toán phiên bản 2

Trong phiên bản 2 này thì bạn cần hiểu một số khái niệm trong máy tính:

- Các ngôn ngữ lập trình thường biểu diễn chuỗi số gồm nhiều kí tự liên tục và được đánh số thứ tự từ 0. Ví dụ chuỗi “123456” được thể hiện bằng các ô nhớ có chỉ số như bên dưới

	1	2	3	4	5	6
Chỉ số	0	1	2	3	4	5

- Biến: là một từ hoặc cụm từ nối với nhau bởi dấu gạch chân để chứa một giá trị nào đó. Ví dụ: $a = 0$ tức là biến a được gán giá trị zero (0).
- Cách viết $a = b$ gọi là phép gán giá trị b cho a .
- Sử dụng ngôn ngữ tự nhiên (tiếng Anh) để mô tả các hành động / hoạt động (gọi là hàm hoặc function). Ví dụ:
`length(s1)` là hàm tính độ dài của biến $s1$.

Cho thông tin đầu vào $s1$ và $s2$ là hai số dạng chuỗi (String). Thuật toán cộng $s1$ và $s2$ như sau:

Bước 1: Xác định độ dài của $s1$, $s2$ và độ dài lớn nhất của $s1$, $s2$.

$len1 = \text{length}(s1)$

$\text{len2} = \text{length}(s2)$

$\text{maxlen} = \max(\text{len1}, \text{len2})$

Bước 2: Duyệt từng kí tự

Cho chỉ số i lặp n lần (từ 0 đến $\text{maxlen} - 1$)⁴. Mỗi lần i tăng lên 1. Mỗi bước lặp thực hiện công việc tiếp theo từ bước 3.

Bước 3: Xác định chỉ số $i1, i2$ đánh dấu kí tự cần lấy của chuỗi $s1$ và $s2$.

Cách tính như sau:

$i1 = \text{len1} - i - 1$

$i2 = \text{len2} - i - 1$

Việc xác định công thức trên có thể có được từ việc viết ra giấy với ví dụ ở trên: $s1 = "123456"$, $s2 = "7890"$.

Độ dài lớn nhất của 2 chuỗi là: $\text{maxlen} = 6$

khi cho $i = 0$ thì $i1$ phải bằng 5 (trong lập trình kí tự tại vị trí 5 trong chuỗi $s1$ là kí tự '6').

khi cho $i = 1$ thì $i1$ phải bằng 4. Tức là khi i tăng lên thì $i1$ giảm đi 1.

khi cho $i = 5$ (tức là đến cuối chuỗi) thì $i1$ phải bằng 0.

Nên công thức $i1$ phụ thuộc vào i đâu đó như sau

$i1 = \text{len1} - i$

(Bên phải dấu bằng là biểu thức có trừ i để thể hiện tính nghịch đảo: khi i tăng thì $i1$ giảm. Và khi i xuất phát là không thì $i1$ phải xuất phát từ độ dài của chuỗi 1 bên phải lấy len1 trừ bớt cho i)

Bước 4: Lấy ra kí tự của tại vị trí $i1, i2$ tương ứng của chuỗi $s1, s2$.

$c1 = s1.\text{charAt}(i1)$

$c2 = s2.\text{charAt}(i2)$

Chú ý trường hợp $i1$ âm ($i1 < 0$) thì gán $c1 = 0$ luôn chứ không thực hiện charAt vì sẽ gây lỗi chỉ số chuỗi không lệ.

Và tương tự nếu $i2 < 0$ thì $c2 = 0$.

Bước 5: Xác định giá trị tương ứng của kí tự (character) $c1, c2$.

$d1 = c1 - '0'$

$d2 = c2 - '0'$

Trong máy tính thì các kí tự (char) thường được biểu diễn bằng một số nguyên, tương ứng mỗi số nguyên thì máy tính sẽ vẽ lên

⁴ Tuy theo Ngôn ngữ lập trình khác nhau thì nên lập từ 0 hoặc từ 1 cho hợp lý. Đa số các Ngôn ngữ lập trình hiện đại thì nên lập từ 0 do cách đánh chỉ số của các kí tự trong chuỗi hoặc trong mảng là từ 0 (zero).

màn hình cái chữ tương ứng để cho ta đọc⁵. Để chuyển một kí tự thành giá trị số (digit) thì bạn chỉ cần lấy giá trị mã ASCII của nó trừ cho mã ASCII của kí tự zero '0'.

Bước 6: Thực hiện cộng hai kí số d1 và d2

$$t = d1 + d2$$

Bước 7: Thực hiện lấy số hàng đơn vị của t và ghi nhận cờ nhớ

$$\text{resultTemp} = t \% 10 \text{ (chia lấy phần dư)}$$

$$\text{mem} = t / 10 \text{ (chia lấy phần nguyên)}$$

Bước 8: Ghép kết quả resultTemp vào kết quả cuối cùng

$$\text{finalResult} = \text{resultTemp} + \text{finalResult}$$

(chú ý ghép vào bên trái và giá trị ban đầu của finalResult là cuối rỗng)

Quay lại bước 2.

Bước 9: Khi kết thúc vòng lặp ở trên nếu biến nhớ mem có giá trị lớn hơn 0 thì ghép tiếp vào kết quả cuối cùng.

$$\text{Nếu mem} > 0 \text{ thì: } \text{finalResult} = \text{mem} + \text{resultTemp}.$$

Đến đây thì bạn có thể hình dung rõ hơn thuật toán cộng hai số như các bạn học sinh lớp 3 thường làm.

Tuy nhiên Thuật toán phiên bản 2 ở trên vẫn còn thiếu sót một chút ở bước 6. Do tôi cố trình bày theo luồng suy nghĩ bình thường để bạn hiểu thuật toán nên trong bước 6 chưa thực hiện phép cộng thêm biến nhớ nếu nó có giá trị.

Chúng ta sẽ cải tiến một chút ở bước 6, thêm bước 0 và bôi đậm các phần bổ sung để bạn tiện theo dõi trong thuật toán phiên bản 3 bên dưới.

Thuật toán phiên bản 3

Cho thông tin đầu vào s1 và s2 là hai số dạng chuỗi (String). Thuật toán cộng s1 và s2 như sau:

Bước 0: Chuẩn bị các biến và kiểu dữ liệu, giá trị ban đầu cho chương trình

String s1: chuỗi lưu số thứ nhất.

String s2: chuỗi lưu số thứ hai.

String finalResult = "" : Chuỗi lưu kết quả cuối cùng.

int len1: số nguyên lưu độ dài của s2.

int len2: số nguyên lưu độ dài của s1.

Ghi nhớ

Cứ trình bày thuật toán một cách tự nhiên như là nói. Sau đó sẽ cải tiến dần để có nội dung cô đọng hơn.

⁵ Xem thêm bảng mã ASCII trong máy tính (<https://vi.wikipedia.org/wiki/ASCII>). Chú kí tự ở đây tôi viết bao bởi dấu nháy đơn để phân biệt với chuỗi được bao bởi dấu nháy đôi.

int maxlen: số nguyên lưu độ dài lớn nhất trong len1 và len2.

char c1: kí tự tạm thời trong quá trình duyệt chuỗi s1

char c2: kí tự tạm thời trong quá trình duyệt chuỗi s2

int d1: giá trị số nguyên của c1

int d2: giá trị số nguyên của c2.

int t: số nguyên lưu tổng tạm của từng kí số d1, d2.

int mem = 0: giá trị của cờ nhớ trong quá trình cộng. Khởi động bằng 0.

Bước 1: Xác định độ dài của s1, s2 và độ dài lớn nhất của s1, s2.

len1 = length(s1)

len2 = length(s2)

maxlen = max(len1, len2)

Bước 2: Cho chỉ số i lặp n lần (từ 0 đến maxlen – 1)⁶. Mỗi lần i tăng lên 1.

Bước 3: Xác định chỉ số i1, i2 đánh dấu kí tự cần lấy của chuỗi s1 và s2.

Cách tính như sau:

$i1 = len1 - i - 1$

$i2 = len2 - i - 1$

Việc xác định công thức trên có thể có được từ việc viết ra giấy với ví dụ ở trên: s1 = “123456”, s2 = “7890”.

Độ dài lớn nhất của 2 chuỗi là: maxlen = 6

khi cho i = 0 thì i1 phải bằng 5 (trong lập trình kí tự tại vị trí 5 trong chuỗi s1 là kí tự ‘6’).

khi cho i = 1 thì i1 phải bằng 4. Tức là khi i tăng lên thì i1 giảm đi 1.

khi cho i = 5 (tức là đến cuối chuỗi) thì i1 phải bằng 0.

Nên công thức i1 phụ thuộc vào i đâu đó như sau

$i1 = len1 - i$

(Bên phải dấu bằng là biểu thức có trừ i để thể hiện tính nghịch đảo: khi i tăng thì i1 giảm. Và khi i xuất phát là không thì i1 phải xuất phát từ độ dài của chuỗi 1 bên phải lấy len1 trừ bớt cho i)

Bước 4: Lấy ra kí tự của tại vị trí i1, i2 tương ứng của chuỗi s1, s2.

c1 = st1.charAt(i1)

Ghi nhớ

Cần phải viết ra giấy, vẽ hình minh họa ngay khi có thể để dễ kiểm chứng nội dung của thuật toán.

⁶ Tùy theo Ngôn ngữ lập trình khác nhau thì nên lập từ 0 hoặc từ 1 cho hợp lý. Đa số các Ngôn ngữ lập trình hiện đại thì nên lập từ 0 do cách đánh chỉ số của các kí tự trong chuỗi hoặc trong mảng là từ 0 (zero).


```
c2 = st2.charAt(i2)
```

Chú ý trường hợp $i1$ âm ($i1 < 0$) thì gán $c1 = 0$ luôn chứ không thực hiện `charAt` vì sẽ gây lỗi chỉ số chuỗi không lệ.

Và tương tự nếu $i2 < 0$ thì $c1 = 0$.

Bước 5: Xác định giá trị tương ứng của kí tự (character) $c1, c2$.

```
d1 = c1 - '0'
```

```
d2 = c2 - '0'
```

Trong máy tính thì các kí tự (char) thường được biểu diễn bằng một số nguyên, tương ứng mỗi số nguyên thì máy tính sẽ vẽ lên màn hình cái chữ tương ứng để cho ta đọc⁷. Để chuyển một kí tự thành giá trị số (digit) thì bạn chỉ cần lấy giá trị mã ASCII của nó trừ cho mã ASCII của kí tự zero '0'.

Bước 6: Thực hiện cộng hai kí số $d1$ và $d2$ và biến nhớ

```
t = d1 + d2 + mem
```

Bước 7: Thực hiện lấy số hàng đơn vị của t và ghi nhận cờ nhớ

```
resultTemp = t % 10 (chia lấy phần dư)
```

```
mem = t / 10 (chia lấy phần nguyên)
```

Bước 8: Ghép kết quả `resultTemp` vào kết quả cuối cùng

```
finalResult = resultTemp + finalResult
```

(chú ý ghép vào bên trái và giá trị ban đầu của `finalResult` là cuối rỗng)

Quay lại bước 2.

Bước 9: Khi kết thúc vòng lặp ở trên nếu biến nhớ `mem` có giá trị lớn hơn 0 thì ghép tiếp vào kết quả cuối cùng.

Nếu $mem > 0$ thì: `finalResult = mem + resultTemp`.

Đọc API document - Kiểm tra khả năng cài đặt thuật toán

Nếu bạn đã học lập trình thì phần này sẽ không mấy khó khăn. Tuy nhiên đối với các bạn mới làm quen lập trình, mới làm quen với GOLANG thì có thể hơi khó một chút.

⁷ Xem thêm bảng mã ASCII trong máy tính. Chữ kí tự ở đây tôi viết bao bởi dấu nháy đơn để phân biệt với chuỗi được bao bởi dấu nháy đôi.

Mục đích của phần này là giúp bạn luyện tập kỹ năng **đọc tài liệu lập trình API**⁸ để kiểm tra các thao tác (tức là các lệnh) trong thuật toán có thể thực hiện bằng ngôn ngữ lập trình GOLANG như thế nào.

Khi hãng Google và cộng đồng lập trình GOLANG cung cấp cho chúng ta thư viện thì họ cũng cung cấp luôn bộ tài liệu để tra cứu các method của thư viện. Bạn có thể tra cứu bằng cách search Google từ khoá “GOLANG API docs”, mở link “<https://golang.org/doc/>”.

Ghi nhớ

Cần lưu đường dẫn tài liệu và luyện tập cách tra cứu.

Ví dụ bạn tìm thông tin của package “string” thì sẽ thấy kết quả như sau:

The screenshot shows the Go.dev website for the 'strings' package. The browser address bar shows 'pkg.go.dev/strings'. The page has a dark header with the Go logo and a search bar. Below the header, it says 'strings' and 'package standard library'. It lists the version as 'go1.17.2 Latest', published on 'Oct 7, 2021', with a 'BSD-3-Clause' license, 7 imports, and imported by 842,090. There are tabs for 'Details' and 'Repository'. The 'Details' tab is active, showing 'Valid go.mod file', 'Redistributable license', 'Tagged version', and 'Stable version'. The 'Repository' tab shows 'cs.opensource.google/go/go'. On the left, there is a sidebar with 'Jump to ...' and a list of links: 'Documentation', 'Overview', 'Index', 'Constants', 'Variables', 'Functions', 'Types', 'Notes', and 'Source Files'. The 'Documentation' tab is selected, showing an 'Overview' section with the text 'Package strings implements simple functions to manipulate UTF-8 encoded strings.' and a link to 'https://blog.golang.org/strings'. Below the overview is an 'Index' section listing several functions: 'func Compare(a, b string) int', 'func Contains(s, substr string) bool', 'func ContainsAny(s, chars string) bool', 'func ContainsRune(s string, r rune) bool', 'func Count(s, substr string) int', 'func EqualFold(s, t string) bool', and 'func Fields(s string) []string'.

Đối với chương trình Cộng hai số này thì bạn cần tra cứu kỹ các hàm của string. API bạn cần tra thực hiện được các chức năng sau:

- Lấy ra kí tự tại một vị trí cho trước trong chuỗi (Returns the char value at the specified index).
- Lấy độ dài (số kí tự) của chuỗi (Returns the length of this string).

⁸ <https://pkg.go.dev/std>

Tham khảo tài liệu tại:

<https://go.dev/blog/strings>

Thiết kế giao diện (User Interface)

Trong yêu cầu của nhà đầu tư thì họ không đề cập đến giao diện tương tác người dùng. Có nhiều giải pháp để lựa chọn cho phần này. Tuy nhiên để vấn đề trở nên đơn giản nhất có thể và giúp bạn làm quen với quy trình phát triển một dự án phần mềm thì tôi tưởng tượng thêm câu chuyện với nhà đầu tư như sau:

Bạn đã trao đổi với nhà đầu tư để có sản phẩm sớm và đưa ra thử nghiệm thì cần giới hạn chức năng cho phiên bản 1.0. Trong đó phần giao diện tương tác giữa người dùng thì nên dùng lệnh thôi. Sau khi phần mềm được cài đặt phù hợp vào máy tính thì người dùng có thể gõ lệnh như bên dưới để chạy chương trình.

```
sum <n1> <n2>
```

Nếu gõ “sum” mà không có tham số hoặc có một tham số là /? thì xem được hướng dẫn sử dụng như sau:

```
sum /?  
Cú pháp sử dụng: sum <n1> <n2>  
<n1>: là số thứ nhất  
<n2>: là số hạng thứ hai  
Ví dụ: sum 1 2
```

Chú ý:

Như vậy trong giai đoạn hình dung thiết kế giao diện cho phần mềm, bạn đã gặp Khách hàng để thống nhất thêm về yêu cầu.

Bước 3: Lập trình và kiểm thử

Sau khi đã trình bày được thuật toán một cách rõ ràng, về cơ bản thì không có gì khuất mắt thì chúng ta bắt tay vào viết code.

Trong quá trình viết code thì có thể chia làm nhiều giai đoạn.

Giai đoạn một – cài đặt logic chính

Bạn nên tập thói quen tạo khung cho mã nguồn và ghi chú tài liệu cho method trước khi bắt tay viết code chi tiết. Ví dụ: Tạo thư mục Sum và file `sum.go` với khung nội dung sau:

```
// Copyright 2012 The Mentor. All rights reserved.  
// Use of this source code is governed by a BSD-style  
// license that can be found in the LICENSE file.
```

```
// Challenge #1 - Program 4: Sum of string-based integers

package main

import "fmt"

// Sum two string-base integers.
// Return: string-base integer.
func sum(n1 string, n2 string) string {
    return "0"
}

func main() {
    fmt.Println("Welcome to Sum program!")

    param1 := "0"
    param2 := "0"
    total := sum(param1, param2)

    fmt.Println("Result:", total)
}
```

Trong đoạn code trên tôi mô tả ngắn gọn cho package main (ở đầu file đầu file), và hàm sum để định hình trong đầu ý nghĩa của hàm mà chúng ta sẽ code. Nội dung của hàm sum lúc này chỉ có dòng return để không lỗi khi biên dịch.

Tiếp theo, chúng ta code vòng lặp để quét từng kí tự của hai chuỗi từ phải qua trái. Chú ý là trước khi code bạn nên ghi chú ý định (suy nghĩ) ra trước để làm.

```
public String sum(String s1, String s2) {
    String finalResult = "";

    // Quét các kí tự của chuỗi s1 và s2 từ phải qua trái

    //// Xác định độ dài của s1, s2 và độ dài lớn nhất của
    //// 2 chuỗi
    int len1 = s1.length();
    int len2 = s2.length();
    int maxLen = (len1 > len2) ? len1 : len2;

    int index1; // chỉ số của kí tự đang xét của chuỗi 1
    int index2; // chỉ số của kí tự đang xét của chuỗi 2

    //// Lặp maxLen lần
    for (int i = 0; i < maxLen; i++) {
```

```
        index1 = len1 - i - 1;
        index2 = len2 - i - 1;
    }

    return finalResult;
}
```

Đoạn code ở trên dùng một vòng lặp thực hiện maxLen lần. Mỗi lần lặp thì chỉ số i sẽ chạy từ 0 cho đến maxLen (chưa đến maxLen, tức là đến maxLen - 1).

Vì chúng ta cần lấy kí tự của chuỗi s1 và s2 từ phải sang trái. Tức là xuất phát từ **vị trí cuối cùng** của chuỗi rồi lần lượt giảm đi 1. Như vậy chúng ta cần tính 2 chỉ số index1 và index2 bởi công thức:

```
index1 = len1 - i - 1;
```

```
index2 = len2 - i - 1;
```

ý

Chú

Phong cách lập trình:

- Trước vòng lặp nên có dòng trắng.

Ghi nhớ

Trước vòng lặp nên có dòng trắng.

Sau khi đã có chỉ số index1 và index2 thì chúng ta sẽ lấy ra kí tự và tính kí số tương ứng.

```
public String sum(String s1, String s2) {
    String finalResult = "";

    // Quét các kí tự của chuỗi s1 và s2 từ phải qua trái

    //// Xác định độ dài của s1, s2 và độ dài lớn nhất của
    //// 2 chuỗi
    int len1 = s1.length();
    int len2 = s2.length();
    int maxLen = (len1 > len2) ? len1 : len2;

    int index1; // chỉ số của kí tự đang xét của chuỗi 1
    int index2; // chỉ số của kí tự đang xét của chuỗi 2
    char c1;    // kí tự tại vị trí index1 của chuỗi s1
    char c2;    // kí tự tại vị trí index2 của chuỗi s2
}
```

```
int d1; // kí số của c1
int d2; // kí số của c2

//// Lặp maxLen lần
for (int i = 0; i < maxLen; i++) {
    index1 = len1 - i - 1;
    index2 = len2 - i - 1;

    c1 = s1.charAt(index1);
    c2 = s2.charAt(index2);

    d1 = c1 - '0';
    d2 = c2 - '0';
}

return finalResult;
}
```

Sáu dòng code bôi vàng được bổ sung để xác định hai kí số d1 và d2. Để xác định kí tự tại vị trí cho trước trong chuỗi thì dùng method `charAt(vị trí)`. Để chuyển một kí tự (char) thành giá trị nguyên tương ứng với nó thì chúng ta sử dụng ý nghĩa của bảng mã ASCII. Cụ thể là ta lấy kí tự trừ đi kí tự zero '0' thì sẽ ra giá trị nguyên của kí tự. Ví dụ: '1' - '0' sẽ được 1.

Sau khi đã có d1 và d2 rồi thì việc cộng từng kí số là không có gì khó khăn.

Chú ý	Phong cách lập trình:
	<ul style="list-style-type: none">Trong quá trình code thì sẽ có nhu cầu khai báo thêm biến. Thì không nên khai báo biến trong vòng lặp. Ở đây tôi khai báo d1 và d2 phía trước vòng lặp for.Đoạn code tính toán d1 và d2 mới thêm vào có ý nghĩa logic với đoạn code đang có. Vì vậy nên sử dụng một dòng trắng để cách ra. Cái này sẽ giúp cho code dễ đọc (readable).

Ghi nhớ
Không khai báo biến trong vòng lặp
Giữa các khối lệnh thực hiện các nghiệp vụ / logic khác nhau nên có dòng trắng.

Chúng ta sẽ bổ sung tiếp code để thực hiện cộng hai kí số.

Code version 1

```
public String sum(String s1, String s2) {
    String finalResult = "";

    // Quét các kí tự của chuỗi s1 và s2 từ phải qua trái

    //// Xác định độ dài của s1, s2 và độ dài lớn nhất của
    //// 2 chuỗi
    int len1 = s1.length();
    int len2 = s2.length();
    int maxLen = (len1 > len2) ? len1 : len2;

    int index1;    // chỉ số của kí tự đang xét của chuỗi 1
    int index2;    // chỉ số của kí tự đang xét của chuỗi 2
    char c1;       // kí tự tại vị trí index1 của chuỗi s1
    char c2;       // kí tự tại vị trí index2 của chuỗi s2
    int d1;        // kí số của c1
    int d2;        // kí số của c2
    int t;         // tổng tạm của d1 và d2;
    int mem = 0;   // nhớ nếu t lớn hơn hoặc bằng 10

    //// Lặp maxLen lần
    for (int i = 0; i < maxLen; i++) {
        index1 = len1 - i - 1;
        index2 = len2 - i - 1;

        c1 = s1.charAt(index1);
        c2 = s2.charAt(index2);

        d1 = c1 - '0';
        d2 = c2 - '0';

        t = d1 + d2;

        // Lấy hàng đơn vị của t ghép vào phía trước kết quả
        finalResult = (t % 10) + finalResult;
        mem = t / 10;
    }
```

```
// Kết thúc vòng lặp, nếu biến nhớ mem có giá trị thì  
// ghép thêm mem vào phía trước kết quả  
if (mem > 0) {  
    finalResult = mem + finalResult;  
}  
  
return finalResult;  
}
```

Phần code màu vàng ở trên thực hiện lấy hàng đơn vị của tổng (tạm) của 2 kí số ghép vào kết quả finalResult.

Để xác định giá trị biết nhớ nếu kết quả t lớn hơn hoặc bằng 10 thì chúng ta lợi dụng phép chia lấy phần nguyên (div). Trong Java sử dụng phép toán $t / 10$ và gán cho biến mem có giá trị nguyên (int).

Khi kết thúc vòng lặp thì có khả năng biến nhớ mem có giá trị. Vì vậy chúng ta phải kiểm tra nếu $mem > 0$ thì ghép tiếp vào kết quả. Chú ý là ghép vào bên trái chuỗi kết quả.

Như vậy đến thời điểm này thì chúng ta đã thực hiện xong ý tưởng chính của thuật toán. Tức là logic chính đã được cài đặt.

Nếu tinh ý thì bạn sẽ thấy đoạn code ở trên có thể chạy được với trường hợp hai chuỗi s1 và s2 truyền vào có độ dài bằng nhau. Nếu s1 và s2 có độ dài khác nhau sẽ gây ra lỗi khi chạy đến lệnh `charAt(index)` vì index có thể nằm ngoài phạm vi chỉ số các kí tự của chuỗi.

“Chạy thử”

Trước khi hoàn thiện các logic phụ thì tại sao chúng ta không “chạy thử” đoạn code ở trên nhỉ? Để xem thành quả của chúng ta ra sao. Chữ “chạy thử” ở đây tôi để trong dấu ngoặc kép có nghĩa chính xác là kiểm thử (Testing). Chính xác hơn là Unit Testing vì phần code này chỉ là một method, method này được xem là một đơn vị (Unit) trong phần mềm. Vì vậy kiểm thử một method gọi là Unit Testing⁹.

⁹ Nhưng nói ngược lại Unit Testing là kiểm thử method thì chưa đầy đủ nhé! Nội dung Unit Testing sẽ được bàn trong các chương kế tiếp

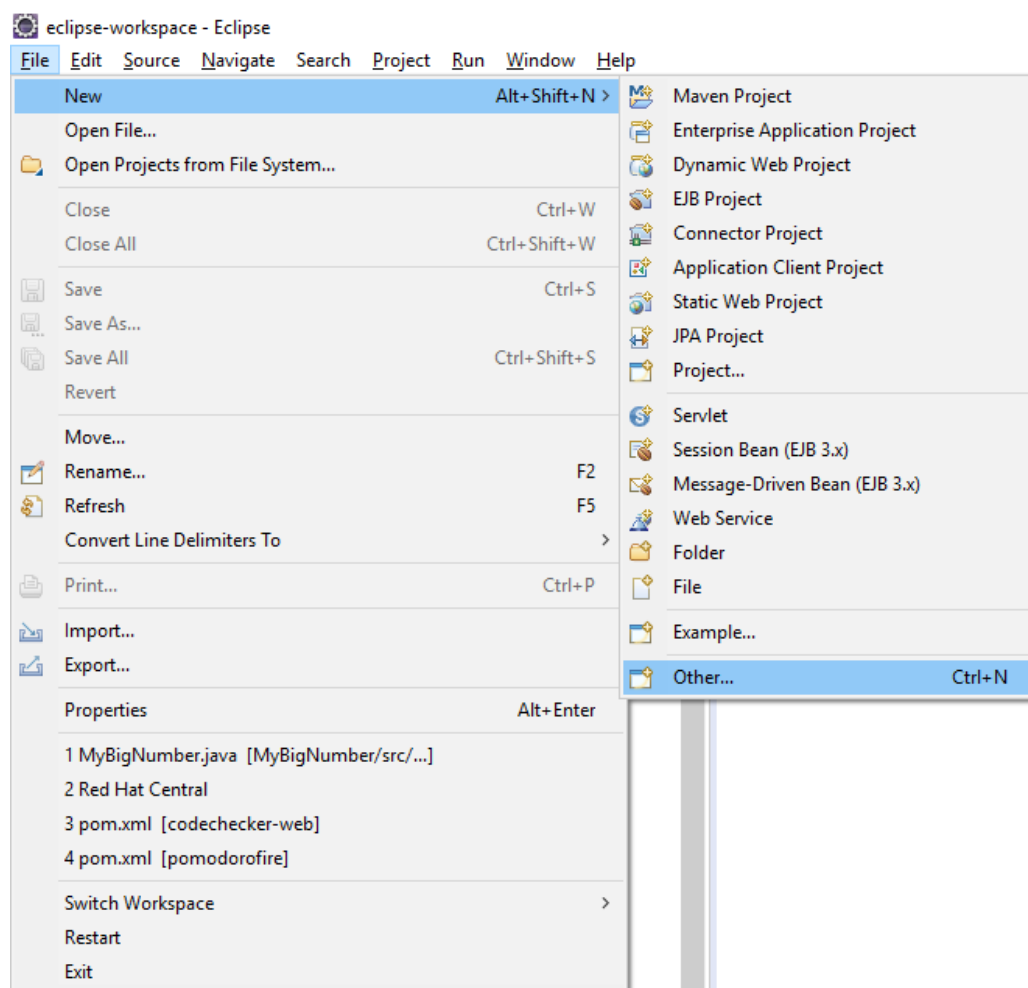
Thông thường thói quen của các bạn mới học lập trình là viết method main để chạy thử theo đúng nghĩa đen là thử. Tuy nhiên để rèn luyện kỹ năng lập trình chuyên nghiệp thì không nên chạy thử mà nên thực hiện Unit Testing để kiểm tra các tình huống theo ý đồ của chúng ta. Để làm quen với kỹ thuật Unit Testing khi lập trình Java với Eclipse thì phần tiếp theo tôi sẽ hướng dẫn các bạn test thử đoạn code cộng hai số dạng chuỗi ở trên.

Tạo Project

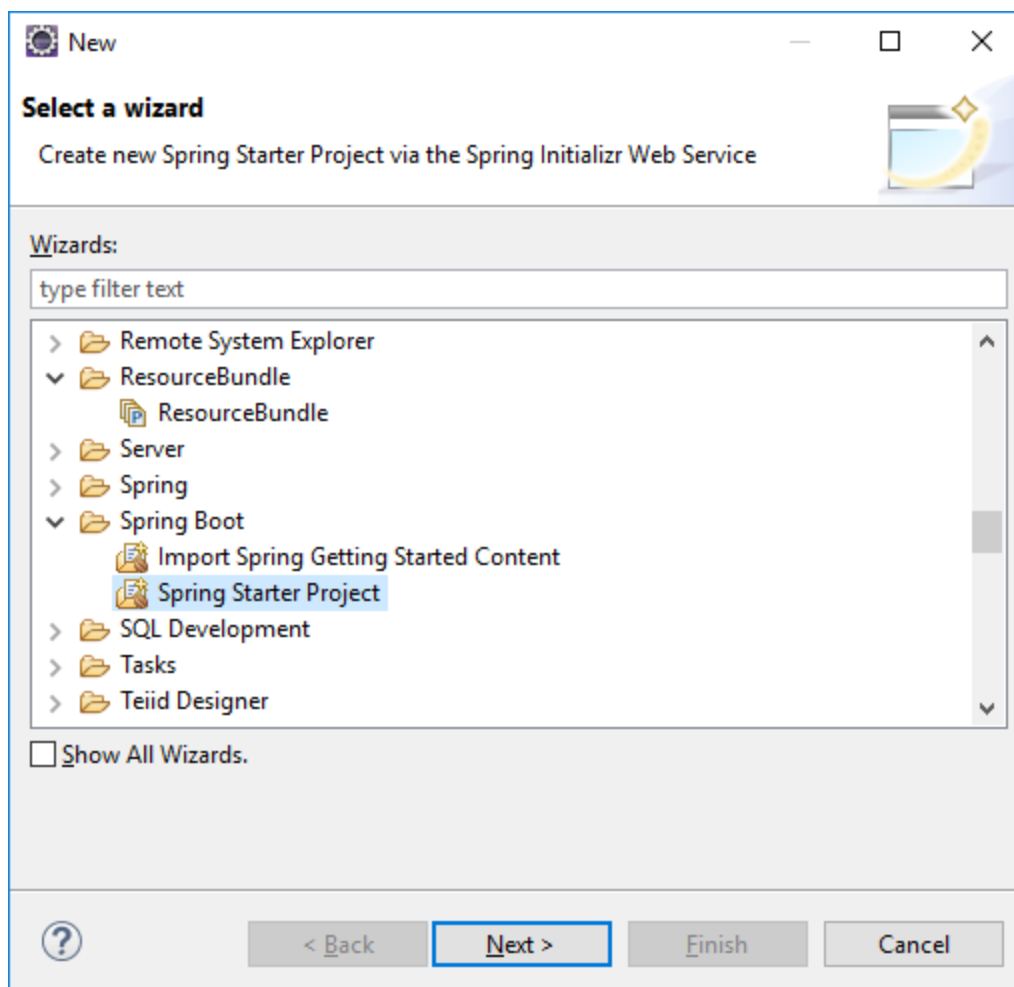
Giả định bạn đã có bộ Eclipse đã cài gói Spring Tool Suite (hoặc sử dụng luôn bộ Spring Tool Suite IDE).

Khởi tạo project bằng Springboot.

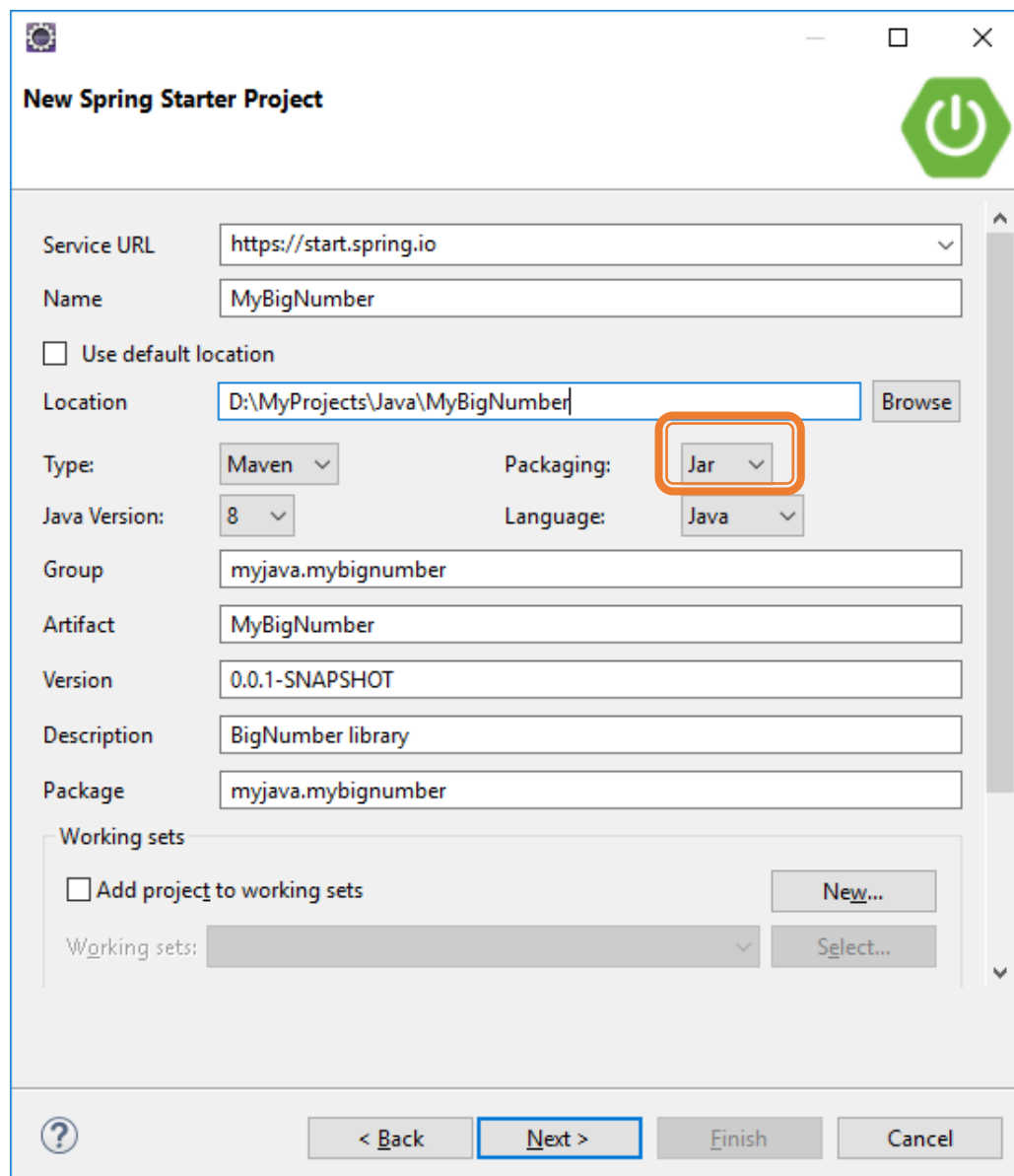
Trong Eclipse, vào menu File > New > Other...



Hoặc nhấn phím tắt Ctrl + N để hiển thị hộp thoại tạo Project.



Chọn mục Spring Boot > Spring Starter project. Sau đó nhấn nút “Next”.



New Spring Starter Project

Service URL:

Name:

☐ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

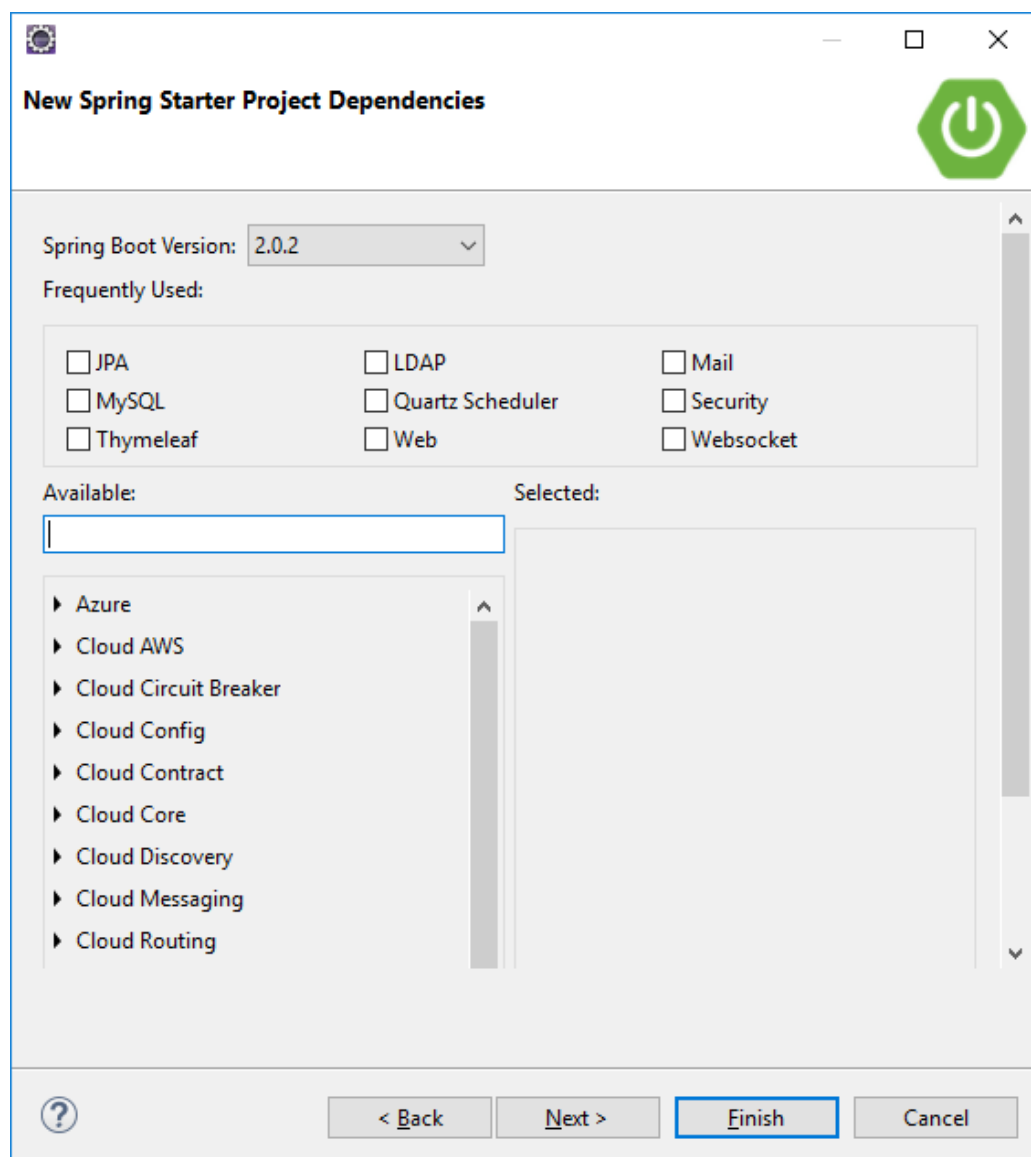
Package:

Working sets

☐ Add project to working sets

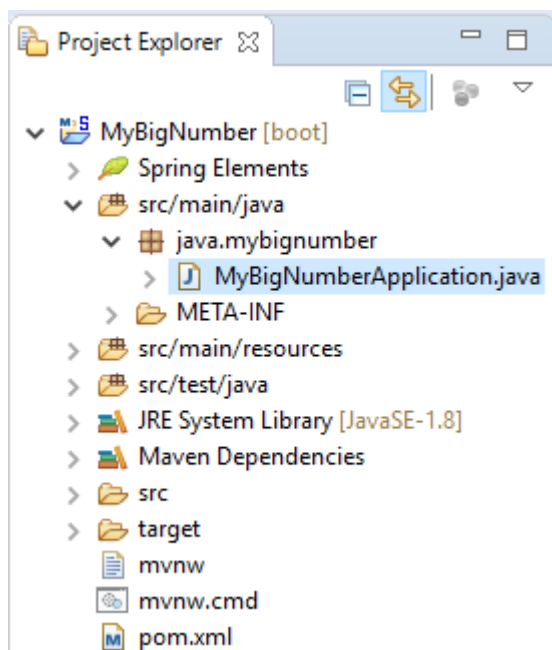
Working sets:

Máy tính của bạn cần phải có kết nối Internet để Eclipse kết nối tới địa chỉ <https://start.spring.io> để khởi tạo dự án. Bạn điền các thông số như trên hình. Sau đó nhấn nút “Next”.



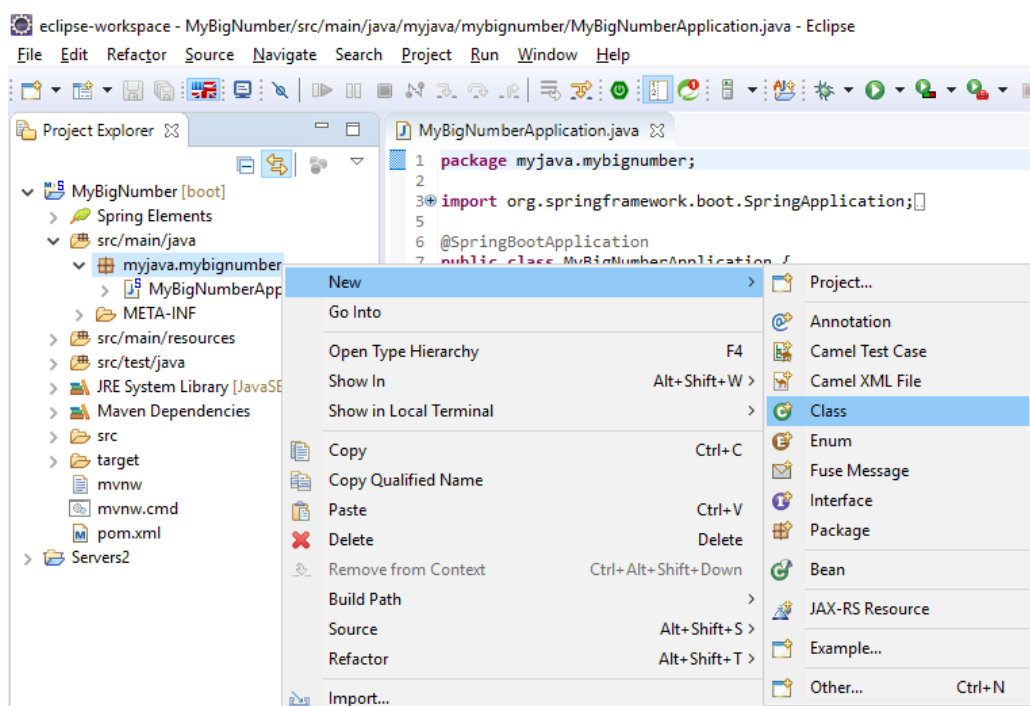
Phiên bản Spring Boot Version tại thời điểm viết tài liệu này là 2.0.2. Bạn nhấn tiếp nút “Next” và “Finish”.

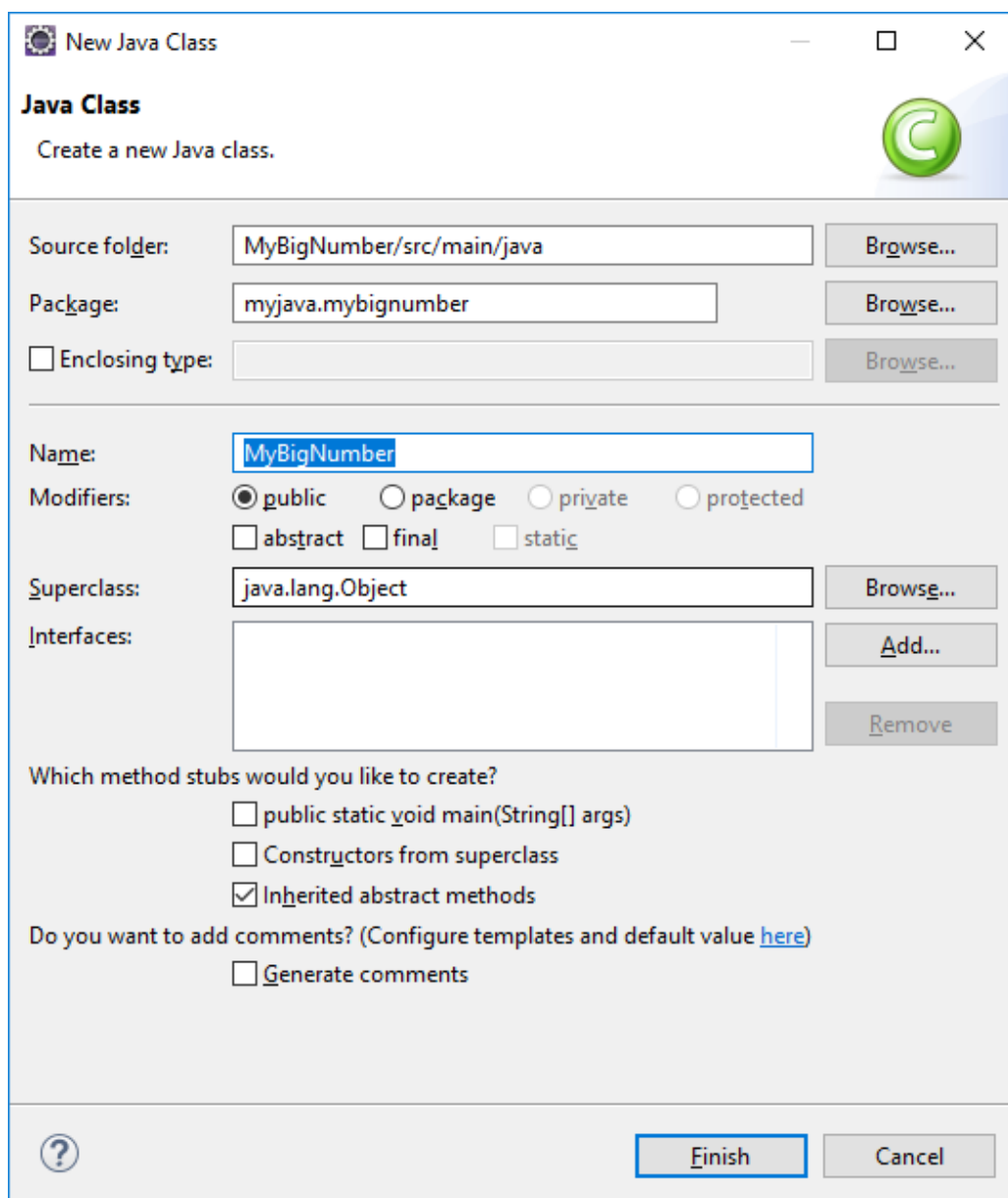
Eclipse sẽ khởi tạo project với khung code mẫu có một file mã nguồn “MyBigNumberApplication.java” như bên dưới:



Tạm thời bạn chưa quan tâm đến file mã nguồn này mà tập trung vào khởi tạo lớp `MyBigNumber` của chúng ta.

Bạn nhấp phải chuột vào package `myjava.mybignumber` ở khung bên trái, chọn menu `New > Class`.



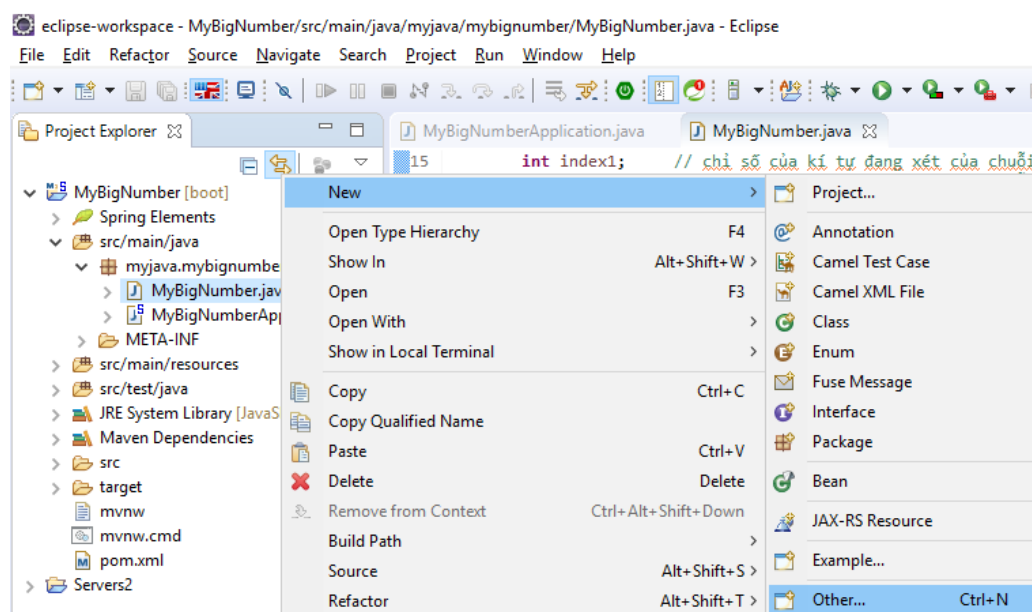


Điền tên class sẽ tạo vào mục Name: MyBigNumber.

Sau đó nhấn nút “Finish” và bắt đầu code method sum như tôi đã hướng dẫn ở trên. Bạn có thể copy và paste từ [source version 1](#).

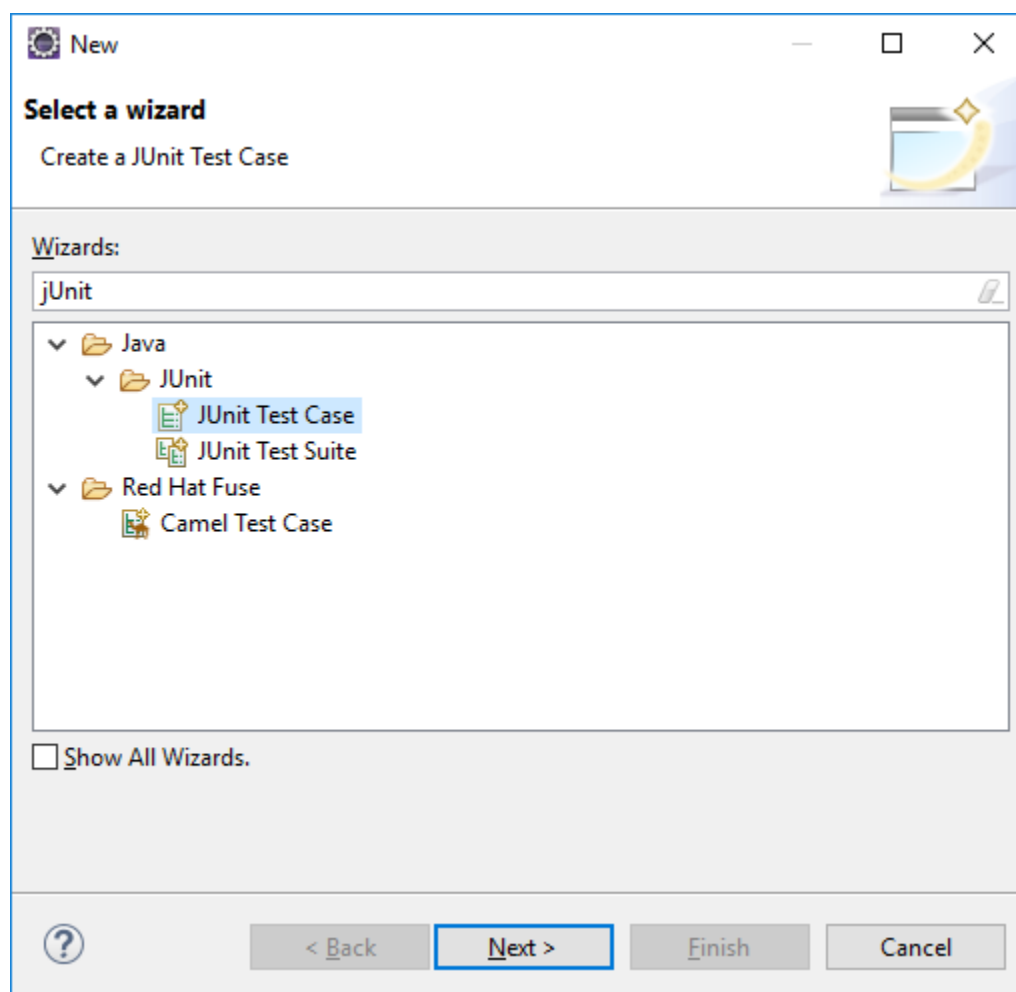
Tạo Test Case

Lúc này bạn đã có source của của method sum(s1, s2). Để tạo ra code kiểm thử thì bạn nhấn phải chuột vào tên file “MyBigNumber.java” ở khung Project Explorer bên trái, chọn menu New > Other...

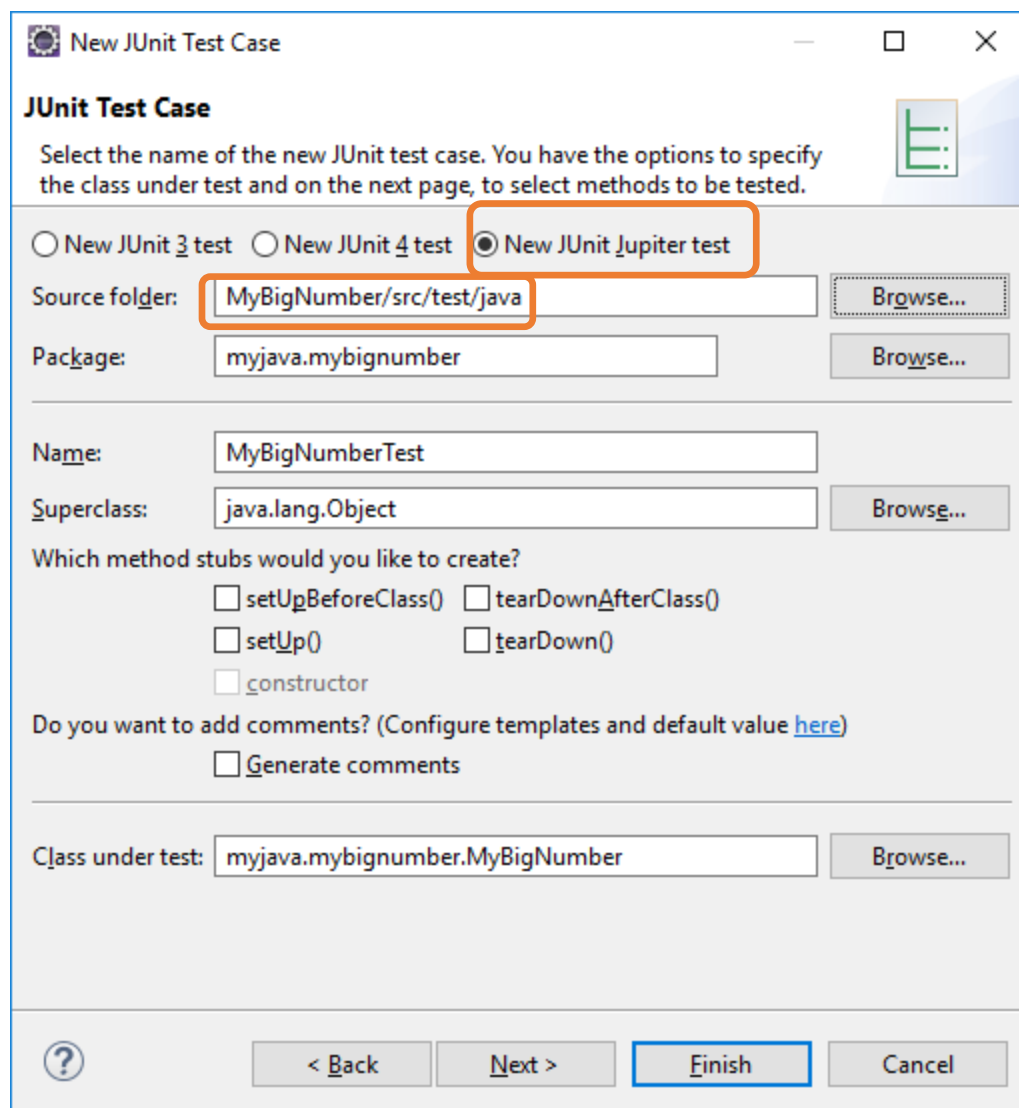


Hoặc nhấn phím tắt “Ctrl + N”.

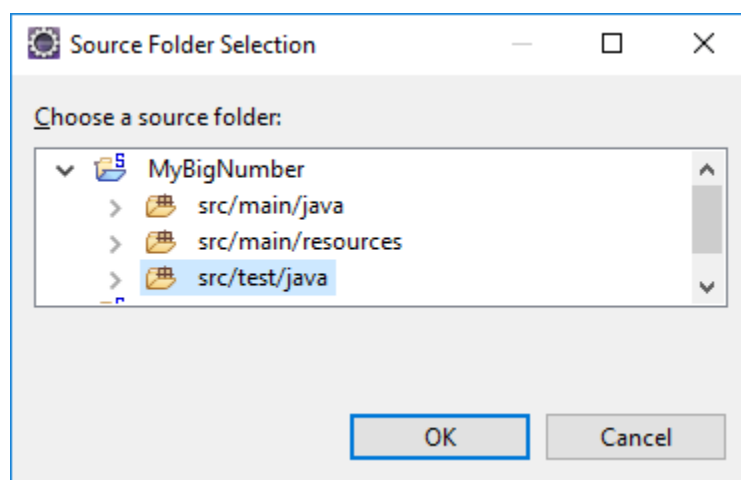
Trong hộp thoại “New”, chọn mục Java > JUnit > JUnit Test Case.



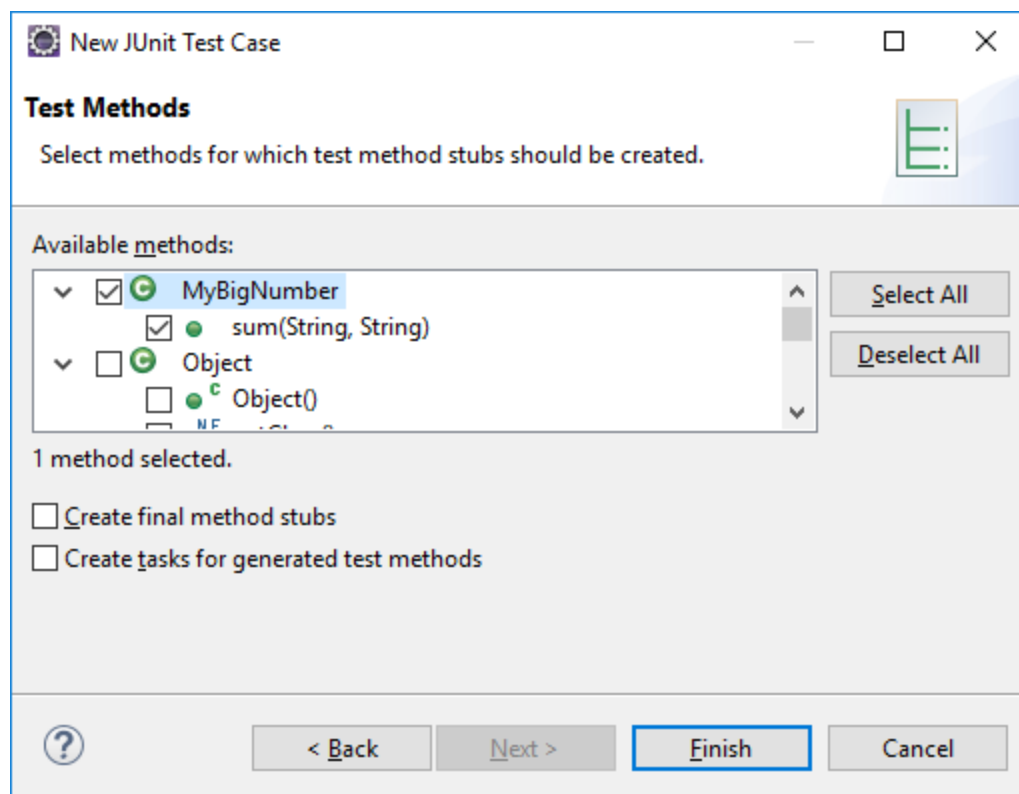
Nhấn nút “Next” để tiếp tục.



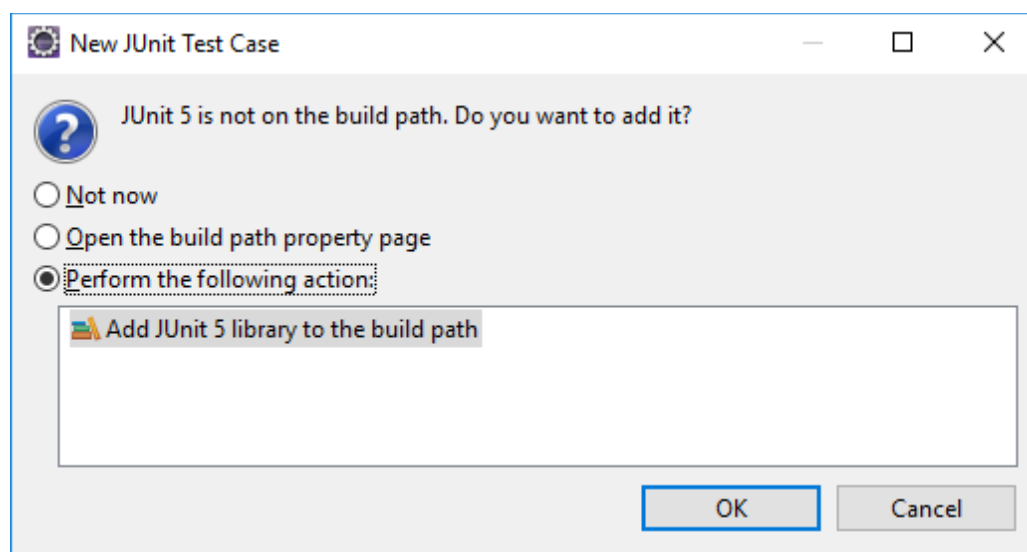
Trong hộp thoại “New JUnit Test Case” bạn chọn mục “New JUnit Jupiter test”. Mục Source folder, bạn bấm nút “Browser...” bên phải để chọn thư mục “src/test/java”.



Sau đó nhấn nút “Next”.



Đánh dấu vào mục `sum(String, String)` rồi nhấn nút Finish.



Nhấn nút “OK” để xác nhận Eclipse sẽ dùng thư viện JUNIT 5 cho project.

Source code JUNIT được tạo ra như sau:

```
package myjava.mybignumber;

import static org.junit.jupiter.api.Assertions.*;

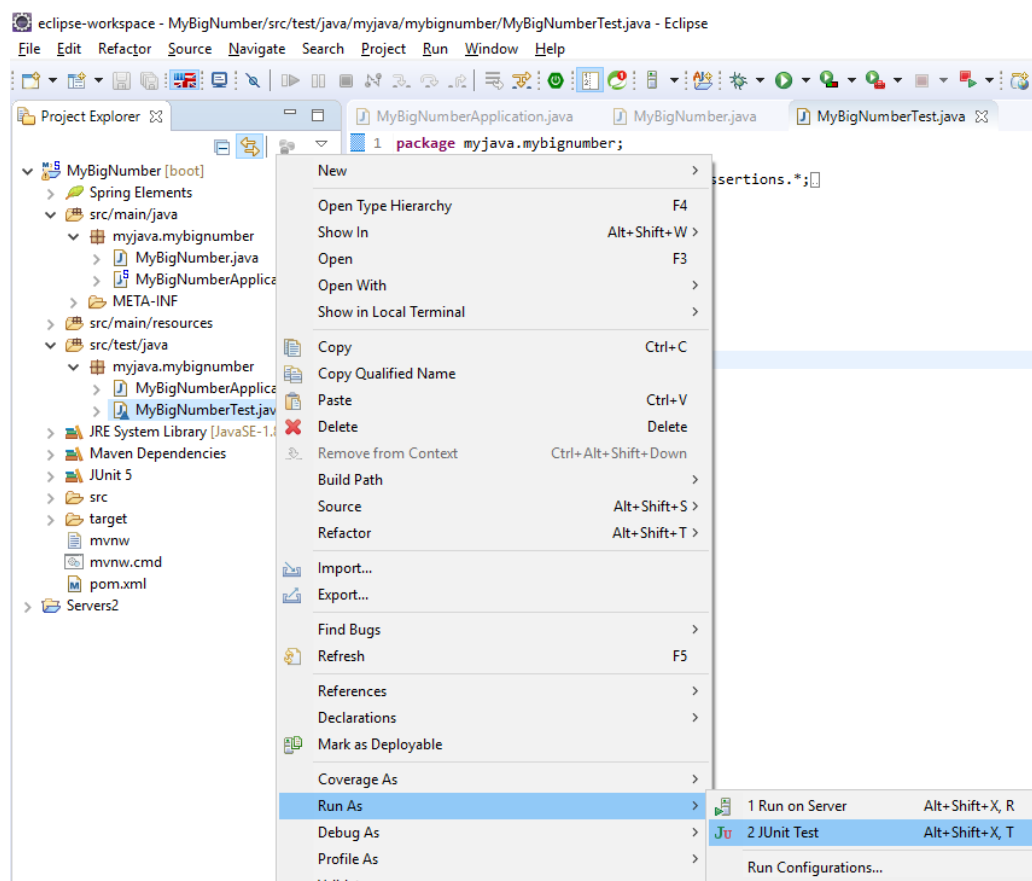
import org.junit.jupiter.api.Test;
```



```
class MyBigNumberTest {  
  
    @Test  
    void testSum() {  
        fail("Not yet implemented");  
    }  
}
```

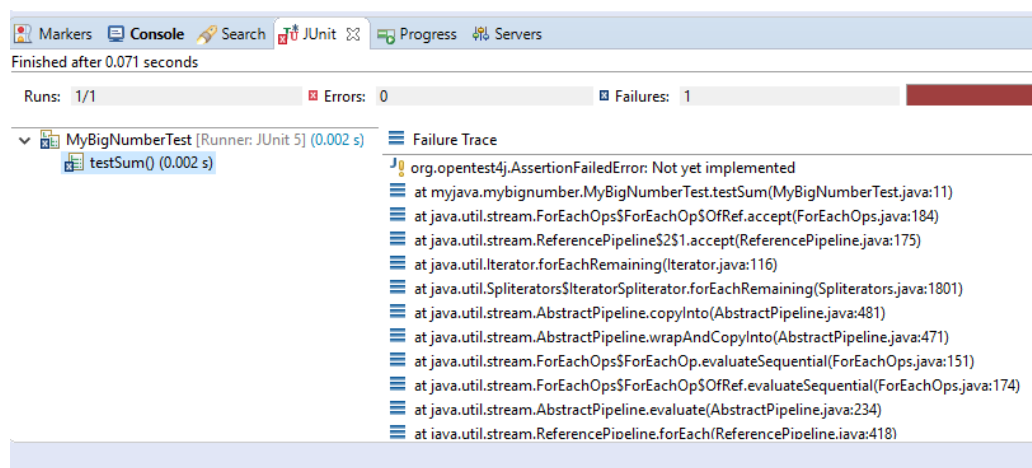
Cú pháp annotation `@Test` phía trước method `void testSum()` có nghĩa là method `testSum` là test case, sẽ được thư viện JUNIT gọi chạy.

Để chạy Test Case này, bạn nhấn phải chuột vào file `MyBigNumberTest.java`, chọn menu `Run As > JUnit Test`.



Hoặc bấm trái chuột vào file `MyBigNumberTest.java` rồi nhấn phím tắt “`Alt + Shift + X`”, sau đó nhấn tiếp phím `T`.

Kết quả thực hiện Test Case sẽ hiện ra trong tab JUnit ở phía đáy màn hình Eclipse.



Do code JUNIT do Eclipse sinh ra có một Test Case “testSum”.

```
@Test
void testSum() {
    fail("Not yet implemented");
}
```

Nội dung test case là lệnh fail(“Not yet implemented”);

Bạn cần xoá dòng này đi và viết lại Test case để gọi method sum(s1, s2) trong class MyBigNumber.

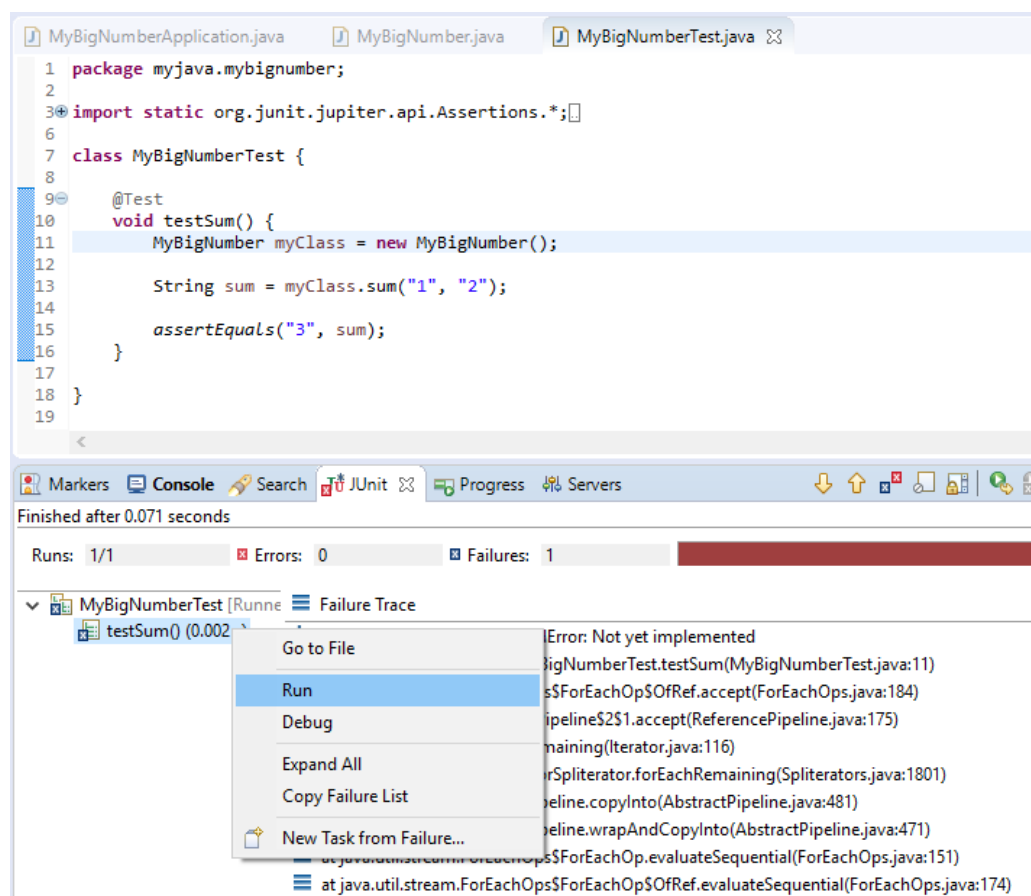
```
@Test
void testSum() {
    MyBigNumber myClass = new MyBigNumber();
    String sum = myClass.sum("1", "2");
    assertEquals("3", sum);
}
```

Nội dung test case “testSum” được viết lại gồm có 3 dòng. Hai dòng đầu là khởi tạo lớp cần test và gọi method để test. Cụ thể là ta cần test thử method sum với 2 chuỗi “1” và “2”.

Dòng thứ ba dùng method assertEquals để kiểm tra kết quả. assertEquals ở đây có 2 tham số. Tham số thứ nhất là giá trị mong đợi (1 cộng với 2 sẽ mong đợi kết quả là 3. Vì là kiểu chuỗi nên ta để trong cặp nháy đôi “3”). Tham số thứ hai là biến cần so sánh với giá trị mong đợi.

Chạy lại test case sau khi đã chỉnh bằng phím tắt Alt + Shift + X, T.

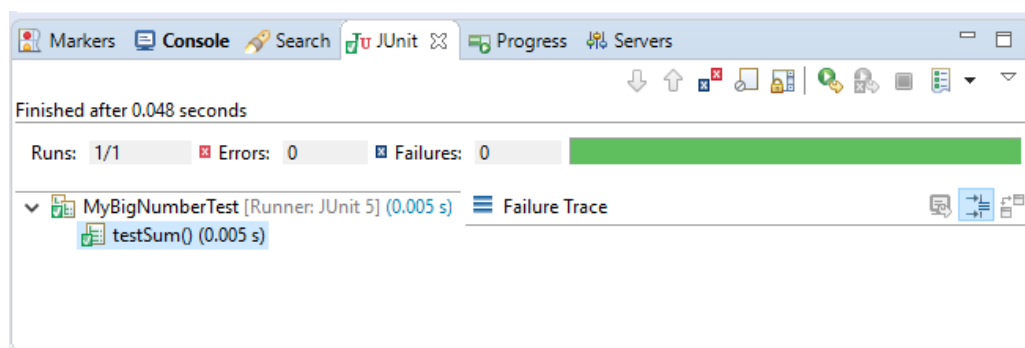
Một cách khác để chạy lại test case là trong cửa sổ JUnit, bấm phải chuột vào test case (tên method), chọn menu Run.



Ghi nhớ

Code xong một method thì nên viết Test Case để kiểm tra.

Kết quả ra như sau:



Dấu stick màu xanh bên trái test case “testSum()” cho biết là đoạn code test đã chạy không có lỗi. Tức là kết quả sum(“1”, “2”) trả lại đúng bằng “3”.

Bạn có thể thử thêm vài phép cộng khác với độ dài 2 chuỗi bằng nhau. Ví dụ tôi thêm hai dòng bôi vàng bên dưới để thử cộng 2 số lớn có nhớ.

```
@Test
void testSum_N_1() {
    MyBigNumber myClass = new MyBigNumber();
    String sum = myClass.sum("1", "2");

    assertEquals("3", sum);
}
```

```
        sum = myClass.sum("8", "9");

        assertEquals("17", sum);
    }
}
```

Trong đoạn code test ở trên tôi có sửa lại tên method là testSum_N_1(). Tên này giúp cho chúng ta gọi nhớ: N là Normal (tức là test trường hợp bình thường, tức là dữ liệu đúng), số 1 là trường hợp thứ nhất. Với tên test case như vậy bạn có thể copy thành nhiều test case khác như sau. Chú ý copy cả anotation @Test.

```
class MyBigNumberTest {

    @Test
    void testSum_N_1() {
        MyBigNumber myClass = new MyBigNumber();
        String sum = myClass.sum("1", "2");

        assertEquals("3", sum);

        sum = myClass.sum("8", "9");

        assertEquals("17", sum);
    }

    @Test
    void testSum_N_2() {
        MyBigNumber myClass = new MyBigNumber();

        String sum = myClass.sum("123", "9");

        assertEquals("132", sum);
    }
}
```

Bạn hãy chạy lại 2 test case này bằng phím tắt Alt + Shift + X, R. Kết quả như thế nào?

The screenshot shows an IDE with three tabs: `MyBigNumberApplication.java`, `MyBigNumber.java`, and `MyBigNumberTest.java`. The `MyBigNumber.java` tab is active, showing the following code:

```
16 int index2; // chỉ số của kí tự đang xét của chuỗi 2
17 char c1; // kí tự tại vị trí index1 của chuỗi s1
18 char c2; // kí tự tại vị trí index2 của chuỗi s2
19 int d1; // kí số của c1
20 int d2; // kí số của c2
21 int t; // tổng tạm của d1 và d2;
22 int mem = 0; // nhớ nếu t lớn hơn hoặc bằng 10
23
24 /// Lặp maxLen lần
25 for (int i = 0; i < maxLen; i++) {
26     index1 = len1 - i - 1;
27     index2 = len2 - i - 1;
28
29     c1 = s1.charAt(index1);
30     c2 = s2.charAt(index2);
31
32     d1 = c1 - '0';
33     d2 = c2 - '0';
34
35     t = d1 + d2;
36
37     // Lấy hàng đơn vị của t ghép vào phía trước kết quả
38     finalResult = (t % 10) + finalResult;
39     mem = t / 10;
```

Below the code editor, the `JUnit` tab shows the test results. The `testSum_N_10` test passed, but the `testSum_N_20` test failed. The `Failure Trace` for the failed test is as follows:

```
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.lang.String.charAt(String.java:658)
    at myjava.mybignumber.MyBigNumber.sum(MyBigNumber.java:30)
    at myjava.mybignumber.MyBigNumberTest.testSum_N_2(MyBigNumberTest.java:26)
    at java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps.java:184)
    at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.java:175)
    at java.util.Iterator.forEachRemaining(Iterator.java:116)
    at java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:1801)
    at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:481)
    at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:471)
    at java.util.stream.ForEachOps$ForEachOp.evaluateSequential(ForEachOps.java:151)
    at java.util.stream.ForEachOps$ForEachOp$OfRef.evaluateSequential(ForEachOps.java:178)
    at java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
    at java.util.stream.ReferencePipeline.forEach(ReferencePipeline.java:59)
```

Ghi nhớ

Phải đọc kỹ từng dòng, từ chữ của thông báo lỗi.

Phân tích lỗi Unit Testing

Kết quả test case “testSum_N1()” là OK nhưng test case “testSum_N_2()” bị báo đỏ. Khung “Failure Trace” bên phải cung cấp thông tin chi tiết hơn về lỗi. Cụ thể:

- Dòng đầu tiên:
`java.lang.StringIndexOutOfBoundsException: String index out of range: -1`

20. Có nghĩa là lỗi index vượt ngoài giới hạn. `index = -1`.

- Dòng thứ 2:
`at java.lang.String.charAt(String.java:658)`
Đây là lỗi tại dòng 658 trong lớp `String` của Java, tại method `charAt`.

Chúng ta sẽ bỏ qua, không phân tích lỗi xuất hiện trong source code của Java.

- Dòng thứ 3: đây là lỗi của chúng ta
21. `at myjava.mybignumber.MyBigNumber.sum(MyBigNumber.java:30)`

Trong source code MyBigNumber.java tại dòng 30 gây ra lỗi index vượt ngoài giới hạn. Bạn double-click vào dòng thứ ba này để Eclipse mở source code cho chúng ta xem.

Vì chúng ta đang test thử sum("123", "9") thì rõ ràng khi chỉ số index1 chạy từ phải sang tới ký tự '2' thì đối với chuỗi s2 đã hết chuỗi (tức là index2 đã lùi về tới -1, bị âm). Điều này gây ra lỗi trong test case số 2.

Tóm tắt:

Như vậy đến thời điểm này bạn đã code được luồng logic chính để cộng hai số dạng chuỗi. Bạn cũng đã biết cách viết Test Case dùng JUNIT để test chương trình thay vì viết method main như thói quen thông thường.

Bạn cũng đã đưa vào một test case để test method sum với 2 tham số có độ dài khác nhau. Kết quả đương nhiên là FAILED.

Việc tiếp theo của chúng ta là tiếp tục nâng cấp code để giải quyết tiếp các tình huống phụ.

Giai đoạn hai – cài đặt các logic phụ

Tôi có gắng trình bày thanh hai phần logic chính và logic phụ là để bạn có thói quen khi lập trình nên tập trung vào chức năng chính trước, hình thành được khung code giải quyết được các vấn đề quan trọng; trình bày code trong sáng, dễ hiểu.

Sau đó mới cải tiến tiếp để giải quyết các luồng xử lý phụ khác.

Cụ thể trong source code ở trên, chúng ta sẽ cần nâng cấp tiếp khi chỉ số index1 và index2 đã lùi về bên trái đến hết chuỗi.

Đoạn code tính c1, c2 được tinh chỉnh lại như sau:

```
c1 = (index1 >= 0) ? s1.charAt(index1) : '0';  
c2 = (index2 >= 0) ? s2.charAt(index2) : '0';
```

Dùng toán tử 3 ngôi cho gọn thay vì dùng if...else.

Một số điểm bạn có thể cải tiến tiếp:

- d1, d2 là các biến trung gian để cho code dễ hiểu. Việc gán c1, c2 bằng ký tự '0' có thể hơi thừa. Nếu muốn tối ưu code thêm thì có thể gán d1, d2 bằng zero luôn khi index1, index2 bị âm.

Sau khi sửa được lỗi khi index 1, index âm thì bạn chạy lại Test Case sẽ phát hiện test case số 2 vẫn bị FAILED. Bạn tập phân tích kết quả test case và phát hiện dòng lệnh `t = d1 + d2` quên cộng thêm biến nhớ “mem”. Source code của vòng for sẽ sửa thêm một chút:

```
for (int i = 0; i < maxLen; i++) {
    index1 = len1 - i - 1;
    index2 = len2 - i - 1;

    c1 = (index1 >= 0) ? s1.charAt(index1) : '0';
    c2 = (index2 >= 0) ? s2.charAt(index2) : '0';

    d1 = c1 - '0';
    d2 = c2 - '0';

    t = d1 + d2 + mem;

    // Lấy hàng đơn vị của t ghép vào phía trước kết quả
    finalResult = (t % 10) + finalResult;
    mem = t / 10;
}
```

Tự trải nghiệm

- Thêm các test case cộng số có nhiều chữ số hơn.

Nâng cấp version 2

Để method `sum(String s1, String s2)` có thể cung cấp được cho người khác sử dụng thì chúng ta cần phải xử lý nhiều tình huống hơn. Cụ thể là cần bổ sung thêm 3 tính năng:

- 1) Nếu các tham số `s1`, `s2` truyền vào là null hoặc emty (rỗng) thì xem như là zero. Ví dụ `sum("", "2")` hoặc `sum(null, "2")` sẽ cho kết quả giống như `sum("0", "2")`.
- 2) Nếu các tham số `s1`, `s2` truyền vào có chứa các ký tự không phải là ký số '0'..'9' thì method sẽ throw (văng ra / ném ra) Exception với các thông chi tiết như sau:

- Exception class: `java.lang.NumberFormatException (String msg)`
- msg: là nội dung của Exception có dạng:

Lỗi ở tham số <param> tại vị trí <index>: <invalid char>.

- `<param>`: là s1 hoặc s2
- `<index>`: là vị trí chứa kí tự lỗi tính theo index của String trong java.
- `<invalid char>`: là kí tự gây lỗi.

Ví dụ: `sum("123", "1a2")` sẽ throw ra `NumberFormatException`("Lỗi ở tham số s2 tại vị trí 1: a").

Nâng cấp version 3

Bổ sung thêm tính năng cho version 2.

- 3) Nếu tham số s1, s2 truyền vào là các kí số nhưng có dấu trừ ở đầu, tức là số âm thì method `sum(s1, s2)` sẽ throw ra `java.lang.NumberFormatException(String msg)` với msg có dạng:

Chưa hỗ trợ số âm `<parm>`: `<value>`

- `<param>`: là s1 hoặc s2
- `<value>`: là giá trị tương ứng của param

Ví dụ: `sum("-123", "45")` sẽ throw ra `NumberFormatException`("Chưa hỗ trợ số âm s1: -123").

Nâng cấp version 4

Lớp `NumberFormatException` sử dụng message làm tham số trong Constructor không được tốt lắm về mặt thiết kế.

Chúng ta sẽ nâng cấp một chút bằng cách tự viết một lớp `ExNumberFormatException(Integer errorPos)` với tham số là Integer để chỉ vị trí lỗi của kí tự không hợp lệ.

Code test cũng sẽ nâng cấp một chút bằng cách kiểm tra 2 nội dung:

- Kiểm tra đúng loại Exception
- Kiểm tra thông tin trong Exception chính xác. Ở đây là `errorPos`

Bước 3.1: Kiểm thử mã nguồn

Trong bước 3 ở trên bạn đã làm quen với cách dùng JUNIT để tạo ra test code gọi method cần test để thực thi. Bạn cũng biết cách dùng method `assertEquals` trong JUNIT để kiểm tra kết quả có đúng mong đợi hay không.

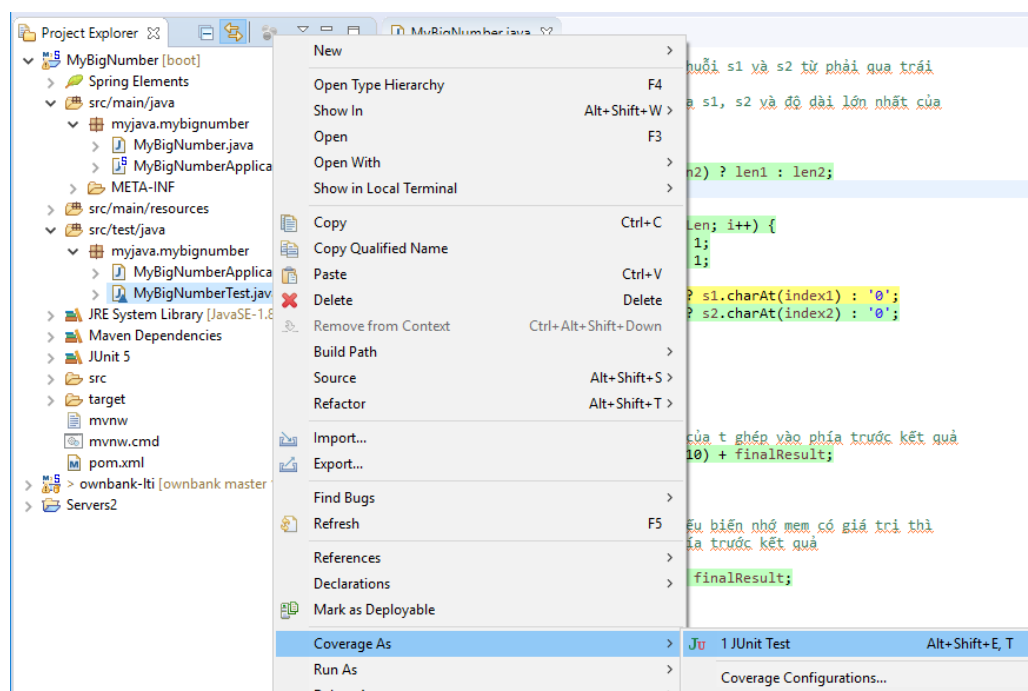
Câu hỏi?

Các test case bạn viết như vậy đã kiểm tra đủ tất cả các tình huống chưa? Bạn viết bao nhiêu test case là đủ?

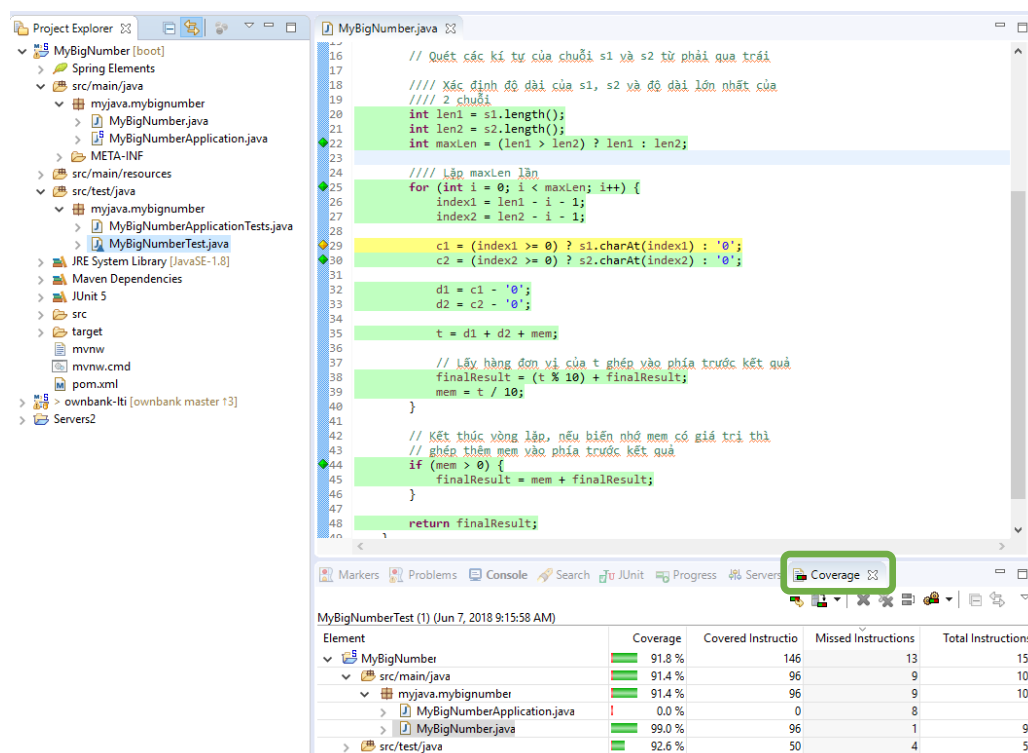
Trả lời của bạn:

Để trả lời câu hỏi trên, bạn hãy thử dùng chức năng “Coverage” trong Eclipse để kiểm tra mức độ bao phủ mã nguồn của các test case. Cách làm như sau:

Bấm phải chuột lên file Test Case “MyBigNumberTest.java” ở khung Project Explorer bên trái. Chọn menu Coverage As > JUnit Test (phím tắt là Alt + Shift + E, T)



Kết quả sẽ được trình bày trong tab “Coverage” như hình bên dưới:



Trong tab “Coverage” cho bạn biết các thông tin sau: Với bộ test case, `MyBigNumberTest.java` thì class `MyBigNumber.java` đã được thực thi (xem cột Element), độ bao phủ (cột Coverage) là 99.0%.

Ghi nhớ
Cố gắng đảm bảo
Coverage là 100%

Coverage được tính bằng: lấy số dòng lệnh đã được thực thi chia cho tổng số dòng lệnh có trong class.

$$\text{Coverage} = \frac{\text{Số dòng lệnh đã thực thi}}{\text{Tổng số dòng lệnh đã viết}} \times 100$$

Ý nghĩa 3 cột tiếp theo có nghĩa là:

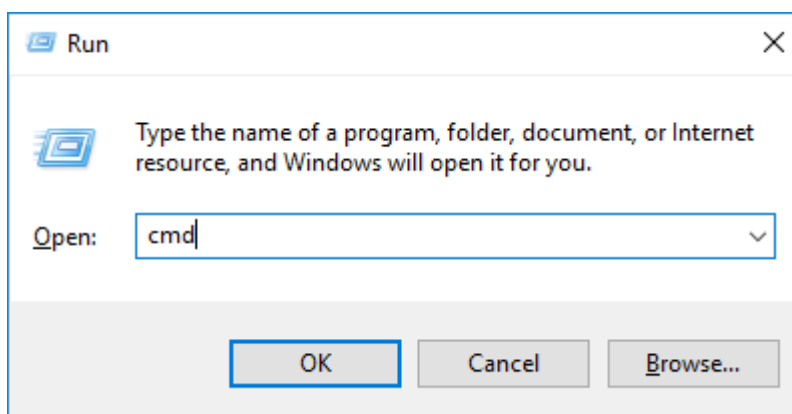
- Covered Instructions: số dòng lệnh đã thực thi (đã chạy).
- Missed Instructions: số dòng lệnh chưa chạy (xem các dòng bôi đỏ, bôi vàng trong khung source code).
- Total Instructions: tổng số dòng lệnh.

Như vậy một định hướng cho chúng ta là cần phải bổ sung test case sao cho khi thực thi thì Coverage là 100%.

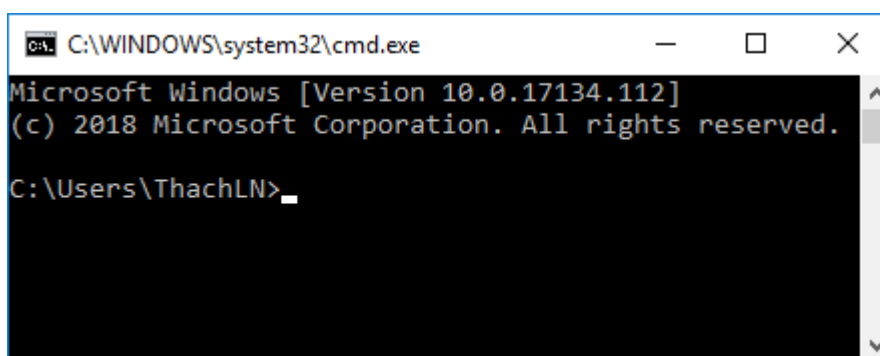
Bước 3.2: Hoàn thiện giao diện

Trong bước 2 tôi có thiết kế giao diện của ứng dụng đơn giản là console. Tức là phần mềm “sum” cho phép người dùng gõ lệnh và chạy trong cửa sổ lệnh của hệ điều hành. Tùy theo hệ điều hành thì cửa sổ lệnh có tên gọi khác nhau.

Trong Windows, để mở cửa sổ lệnh thì nhấn phím tắt Windows + R (Hầu hết các bàn phím trên máy PC hoặc laptop mới thì có phím có cửa sổ - biểu tượng của Microsoft Windows, gần phím Alt bên trái) để mở hộp thoại Run:



Trong mục Open, bạn gõ chữ “cmd” (không có dấu nháy) rồi nhấn Enter. Cửa sổ lệnh sẽ hiện ra như bên dưới:



Quay lại phần mềm Cộng hai số, bạn cần tạo một class có method main để thực thi chương trình. Nếu bạn dùng Springboot thì sẽ có sẵn class này.

Nhiệm vụ của bạn

Viết code cho method main để lấy các tham số, khởi tạo đối tượng MyBigNumber và gọi method sum để tính toán.

Bước 4: Kiểm thử hệ thống

Sau khi bạn đã lập trình và kiểm thử xong trong bước 3. Bây giờ là lúc bạn đóng gói và thử triển khai phần mềm của mình lên máy của một người khác để kiểm tra lại. Trong công nghệ phần mềm, bước này gọi là Integration Test hoặc System Testing. Hai chủ đề này không nằm trong phạm vi của cuốn sách này. Trong một phần mềm rất nhỏ như Cộng hai số thì hai thuật ngữ này có thể xem là một.

Nhiệm vụ của bạn trong bước này là đóng gói phần mềm và cài đặt phần mềm vào một máy khác để thực hiện System Testing. Để đơn giản thì chúng ta có thể gọi là chạy thử.

Đóng gói

Trong bước 3, bạn đã tạo Project dùng Springboot. Chúng ta cần phần mềm maven để biên dịch và đóng gói chương trình.

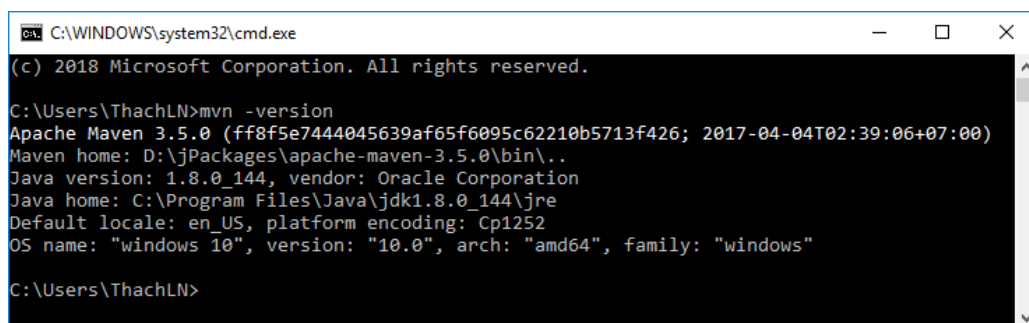
Cài đặt maven

Để cài đặt maven (<http://maven.apache.org>) bạn thực hiện các bước sau:

- Tải gói binary. Vd: tải gói
“<http://mirrors.viethosting.com/apache/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.tar.gz>” về thư mục “D:\jPackages”.
- Extract maven. Vd: dùng phần mềm 7-zip (<https://7-zip.org>) để extract ra thư mục “D:\jPackages\apache-maven-3.5.0” sao cho thư mục “bin” có đường dẫn là “D:\jPackages\apache-maven-3.5.0\bin”.
- Thêm biến môi trường “M2_HOME” vào trong hệ điều hành. M2_HOME có giá trị là:
D:\jPackages\apache-maven-3.5.0
- Cập nhật biến môi trường PATH của hệ điều hành bằng cách thêm đường dẫn: %M2_HOME%\bin.
- Kiểm tra lại maven đã cài đặt đúng chưa bằng cách mở cửa sổ lệnh gõ:

22. `mvn -version`

Kết quả tương tự như sau:



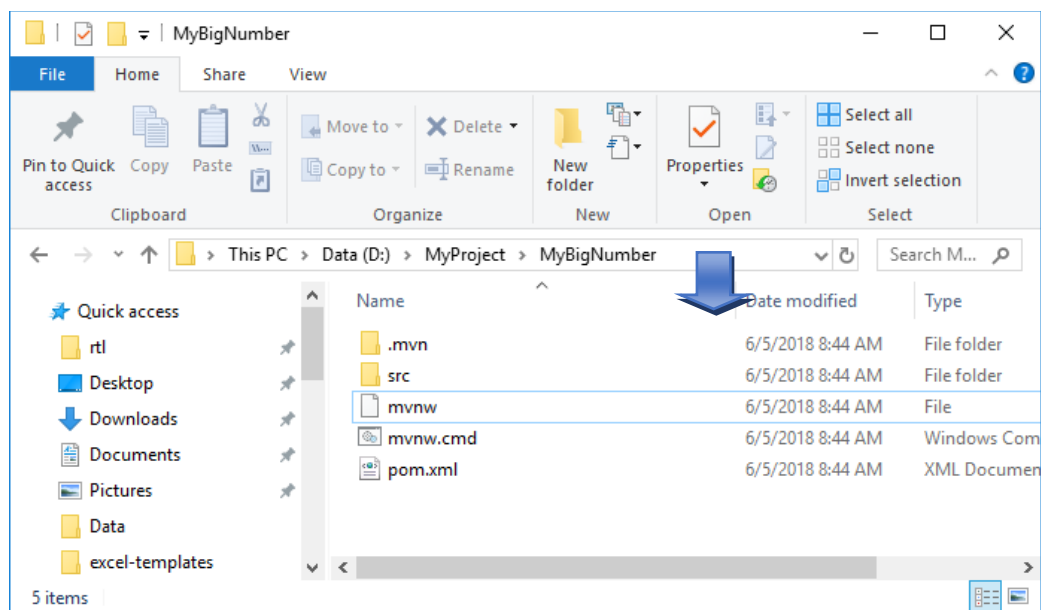
```
C:\WINDOWS\system32\cmd.exe
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ThachLN>mvn -version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-04T02:39:06+07:00)
Maven home: D:\jPackages\apache-maven-3.5.0\bin\..
Java version: 1.8.0_144, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_144\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

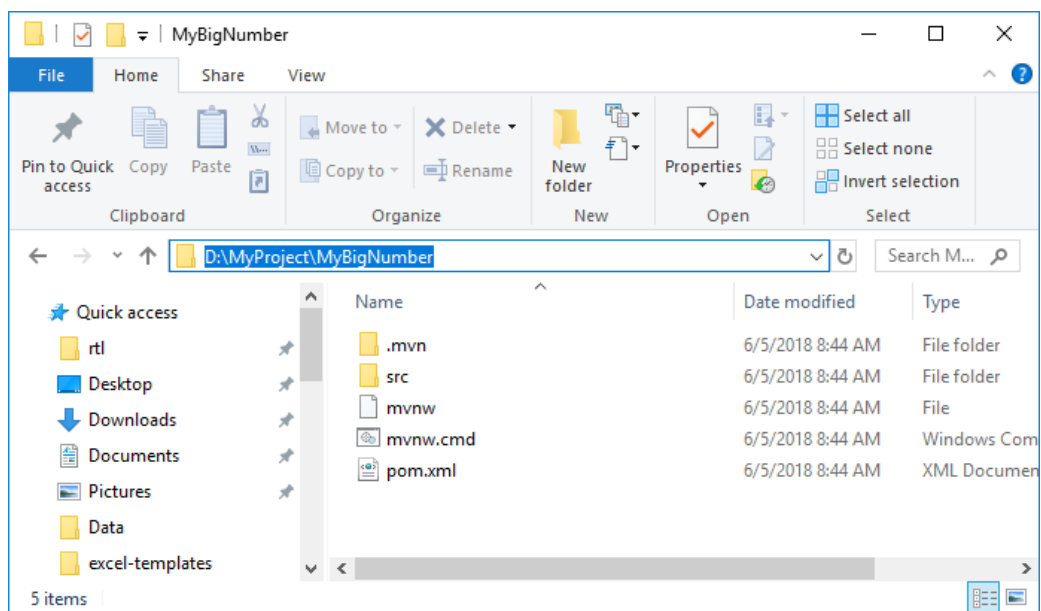
C:\Users\ThachLN>
```

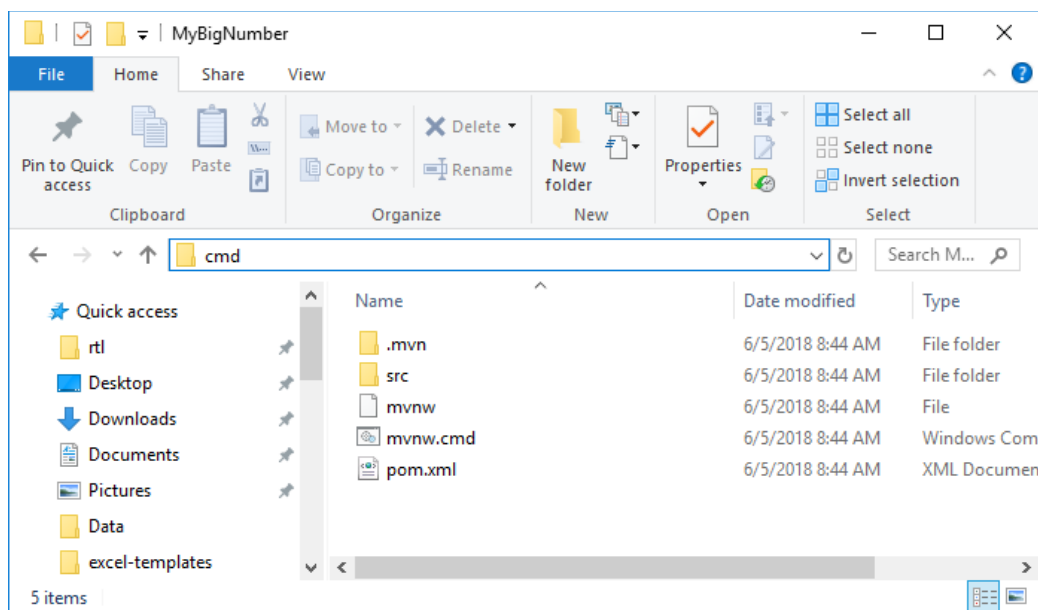
Tạo file .jar

Bạn cần mở cửa sổ lệnh với thư mục làm việc (working directory) là thư mục mã nguồn của dự án. Trong Windows Explorer đang mở thư mục dự án, bạn có thể mở nhanh cửa sổ lệnh bằng cách bấm chuột vào thanh Address – nhớ bấm vào chỗ trống bên phải tên thư mục.

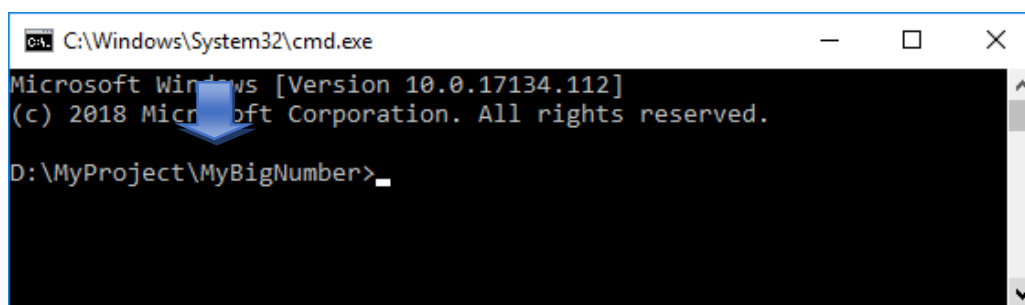


Lúc đó đường dẫn thư mục của dự án được bôi xanh, bạn gõ luôn chữ rồi gõ “cmd” (không có dấu nháy). Sau đó nhấn Enter.





Cửa sổ lệnh hiện ra với đường dẫn của dự án như bên dưới:



Tiếp theo, bạn gõ lệnh “mvn clean package” rồi nhấn Enter để biên dịch và đóng gói dự án. Kết quả của lệnh này là

Trong trường hợp quá trình biên dịch bị lỗi thì bạn thử lệnh:

```
mvn clean package -Dmaven.test.skip
```

Tham số “-Dmaven.test.skip” có nghĩa là bỏ qua quá trình testing.

Sau khi có kết quả “**BUILD SUCCESS**” thì bạn lấy kết quả file jar tại thư mục “.\target\MyBigNumber-0.0.1-SNAPSHOT.jar” tính tại thư mục bạn đang làm việc.

Chạy thử chương trình trong file .jar

Thực thi chương trình bằng đánh lệnh:

```
java -jar ./target/MyBigNumber-0.0.1-SNAPSHOT.jar "1" "2"
```

Như vậy bạn đã biên dịch, tạo file .jar cho chương trình và có thể thực thi chương trình thông qua gói .jar. Tuy nhiên việc gõ lệnh để chạy chương trình như trên sẽ gây khó khăn cho người dùng.

Bản hãy suy nghĩ cách để cải tiến cách đóng gói sản phẩm cho người dùng tiện lợi hơn xem!

Triển khai

Bạn tạo một thư mục “Release\sum” để chuẩn bị đóng gói sản phẩm cho khách hàng theo các bước như sau:

- Copy file `MyBigNumber-0.0.1-SNAPSHOT.jar` vào thư mục “Release\sum”.
- Trong thư mục “Release\sum”, tạo file `sum.cmd` có nội dung sau:

```
java -jar D:/sum/MyBigNumber-0.0.1-SNAPSHOT.jar %1 %2
```

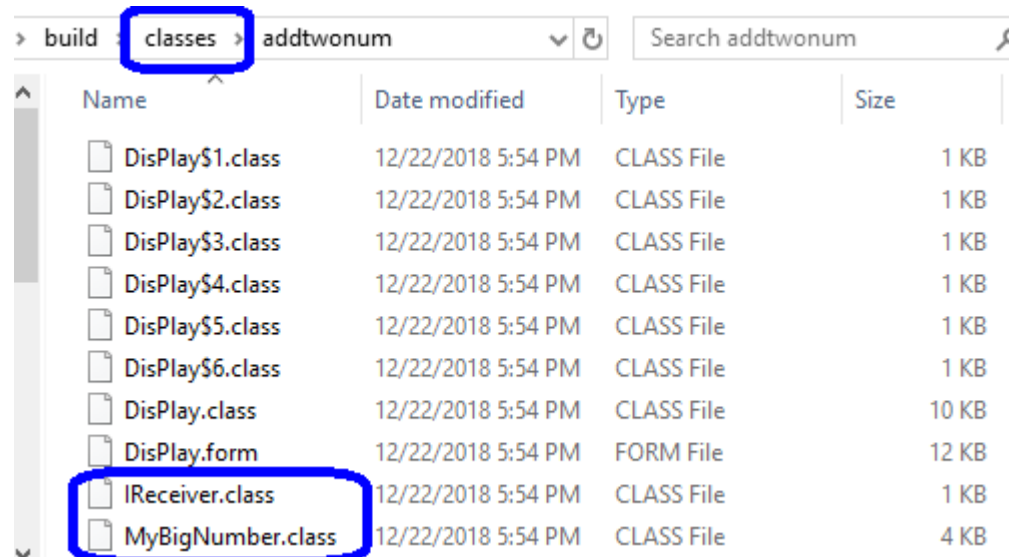
Để triển khai chương trình Cộng hai số bạn thực hiện 2 bước sau:

- Copy thư mục “sum” ở trên vào ổ đĩa D: của máy khách hàng.
- Thêm đường dẫn “`D:/sum`” vào biến môi trường `PATH` trên máy khách hàng.
- Mở cửa sổ lệnh gõ thử lệnh: `sum “1” “2”`

Nâng
cao

Khi muốn chủ động đóng gói vài file `.class` vào thành thư viện `.jar` thì bạn có thể dùng lệnh “`jar`”.

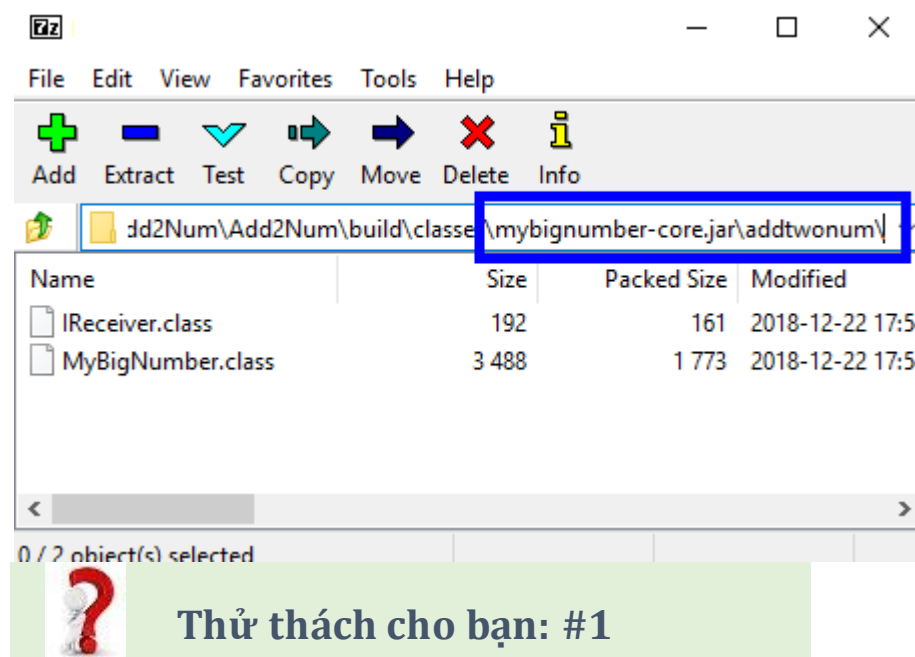
Ví dụ trong hình bên dưới bạn muốn đóng gói 2 file `IReceiver.class` và `MyBigNumber.class` thành thư viện `mybignumber-core.jar`?



Mở cmd trong thư mục “classes”, thực thi lệnh sau (viết trên 1 dòng):

```
jar cvf mybignumber-core.jar .\addtwonum\MyBigNumber.class  
.\addtwonum\IReceiver.class
```

Sau đó dùng 7-zip ở open file “`mybignumber-core.jar`” trong thư mục “classes” để kiểm tra lại nội dung file:



Chủ đầu tư có nhu cầu nâng cấp phần mềm này cho phép cộng hai số lớn được lưu trong file text.

Cú pháp sử dụng như sau:

```
sum <file path 1> <file path 2> -t file
```

<file path 1>: là đường dẫn của file văn bản chứa các kí số đầu tiên.

<file path 2>: là đường dẫn của file văn bản chứa các kí số thứ hai.

-t file: cho biết là kiểu file (t: viết của chữ type)

Gợi ý

- Áp dụng tính Overload trong lập trình Java để bổ sung method cho source code có sẵn.

Thử thách cho bạn: #2

Áp dụng các kiến thức, kỹ năng đã học hãy viết một ứng dụng loại bỏ dấu tiếng Việt (Remove Tone Marks) trong một văn bản có sẵn. Hỗ trợ tiếng Việt với bảng mã UTF-8.

Phiên bản 1: Hỗ trợ truyền văn bản trên dòng lệnh

```
rtm "Chúng tôi là Lập trình viên Java chuyên nghiệp."
```


Vd: lệnh

```
rtm "Chúng tôi là Lập trình viên Java chuyên nghiệp."
```

sẽ in ra màn hình câu bên dưới (không có kẻ ô chữ nhật):

```
Chung toi la Lap trinh vien Java chuyen nghiep.
```

Phiên bản 2: Hỗ trợ truyền đường dẫn file đầu vào và file đầu ra trên dòng lệnh

```
rtm <input file> <output file> -t file
```

<input file>: là đường dẫn của file văn bản chứa chứa tiếng Việt UTF-8.

<output file>: là đường dẫn của file kết quả sau khi đã loại bỏ tiếng Việt.

-t file: cho biết là kiểu file (t: viết của chữ type)

Gợi ý

- Luyện tập kỹ năng viết yêu cầu ra file High Level Requirement.
- Luyện tập kỹ năng viết thuật toán ra giấy hoặc ra file trên máy tính. Suy nghĩ kỹ cách làm thật đơn giản, dễ hiểu như cách suy nghĩ thông thường – chưa cần phải tối ưu.
- Luyện tập cách trang Javadoc API JDK để tìm method sử lý chuỗi (String) phù hợp.

Tham khảo thêm source code bằng CSharp:

```
static string sum(string number1, string number2)
{
    int number1Len = number1.Length, number2Len = number2.Length;
    int lenMax = number1Len > number2Len ? number1Len : number2Len;
    string result = String.Empty;
    int temp;
    int idx1, idx2;
    char c1, c2;
    int d1, d2;
    int mem = 0;

    for (int i = 0; i < lenMax; i++)
    {
        idx1 = number1Len - i - 1;
        idx2 = number2Len - i - 1;
        c1 = (idx1 >= 0) ? number1[idx1] : '0';
```

```
        c2 = (idx2 >= 0) ? number2[idx2] : '0';

        d1 = c1 - '0';
        d2 = c2 - '0';
        temp = d1 + d2 + mem;
        mem = temp / 10;
        temp = temp % 10;
        result = temp + result;
    }

    if (mem > 0) { result = mem + result; }

    return result;
}
```

Tham khảo source code bằng C

```
char* addBigNumber(char* number1, char* number2)
{
    size_t number1Len = strlen(number1);
    size_t number2Len = strlen(number2);
    size_t temp;
    int idx1, idx2;
    char c1, c2;
    size_t d1, d2;
    size_t mem = 0;

    size_t lenMax = number1Len > number2Len ? number1Len : number2Len;

    char* result = new char[lenMax + 2];

    memset(result, ' ', lenMax);
    bool remember = false;

    for (int i = 0; i < lenMax; i++)
    {
```

```
        idx1 = number1Len - i - 1;
        idx2 = number2Len - i - 1;

        c1 = (idx1 >= 0) ? number1[idx1] : '0';
        c2 = (idx2 >= 0) ? number2[idx2] : '0';

        d1 = c1 - '0';
        d2 = c2 - '0';

        temp = d1 + d2 + mem;

        mem = temp / 10;
        temp = temp % 10;

        result[lenMax - i] = temp + '0'
    }

    result[lenMax + 1] = '\0';

    if (mem > 0) {
        result[0] = '1';
    }

    return result;
}
```

Tham khảo thêm source code bằng Python

```
def addBigNumber(number1, number2):
    number1Len = len(number1)
    number2Len = len(number2)
    lenMax = max(number1Len, number2Len)
    result = ""
    mem = 0

    for i in range(lenMax):
```

```
idx1 = number1Len - i - 1
idx2 = number2Len - i - 1

c1 = number1[idx1] if idx1 >= 0 else '0'
c2 = number2[idx2] if idx2 >= 0 else '0'

d1 = int(c1)
d2 = int(c2 )
temp = d1 + d2 + mem
mem = int(temp / 10)
temp1 = int(temp % 10)
result = str(temp1) + result

if mem > 0:
    result = str(mem) + result

return result
```

Thử thách cho ngày 2

Chương trình ReadCSV_EmployeeV1

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: ?
 - 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).

Gợi ý:

- Sử dụng cấu trúc (struct) để mô tả dữ liệu của nhân viên.

Chương trình ReadCSV_EmployeeV2

Nâng cấp chương trình ReadCSV_EmployeeV1 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 sao cho hợp lý (tối ưu bộ nhớ, tốc độ truyền dữ liệu) với các chức năng sau:
 - o Đọc file csv vào danh sách (bộ nhớ máy tính).
 - o Thêm 1 nhân viên mới vào danh sách.
 - o Xóa 1 nhân viên khỏi danh sách theo Mã nhân viên.
 - o Cập nhật thông tin nhân viên trong danh sách.
 - o Tìm kiếm nhân viên có mã số chứa một chuỗi ký tự cho trước, không phân biệt chữ hoa và chữ thường.
 - o Lưu file csv ra thư mục được chỉ định.

- Lấy danh sách nhân viên có ngày sinh nhật trong tuần tới (tuần được tính từ CN đến Thứ Hai).
- Chương trình hỗ trợ file CSV (dùng mã UTF-8 hợp lý).

Gợi ý:

- Sử dụng cấu trúc (struct) để mô tả dữ liệu của nhân viên.
- Sử dụng slice để chứa danh sách nhân viên.

Chương trình ReadCSV_EmployeeV2

Nâng cấp chương trình ReadCSV_EmployeeV1 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 backend 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 sao cho hợp lý (tối ưu bộ nhớ, tốc độ truyền dữ liệu) với các chức năng sau:

Ngày 3: Biểu diễn thông tin phức hợp

Bài 1 – Biểu diễn thông tin phức hợp với GO

Cấu trúc (Structure)

Khám phá đoạn code sau:

```
package main

import (
    "fmt"
)

type aStructure struct {
    person string
    height int
    weight int
}

func main() {
    p1 := aStructure{"Thạch", 165, 72}

    fmt.Println(p1)
}
```

```
{Thạch 165 72}
```

Kết hợp Slice và Structure

Khám phá đoạn chương trình sau:

```
package main

import (
    "fmt"
)

type aStruct struct {
    person string
    height int
    weight int
}
```



```
}

func main() {
    pSlice := [2]aStruct{}

    pSlice[0] = aStruct{"Thạch", 165, 72}
    pSlice[1] = aStruct{"Ngọc", 170, 77}

    fmt.Println(pSlice)
}
```

```
[{Thạch 165 72} {Ngọc 170 77}]
```

Ví dụ trên minh họa khai báo một slice của struct có độ dài 2 phần tử. Sau đó gán mỗi phần tử bởi một cấu trúc.

Mảng động cho cấu trúc

Nâng cấp một chút cho ví dụ trên để minh họa cách thêm phần tử (cấu trúc) cho mảng không giới hạn kích thước (nhắc lại trong GOLANG gọi là kiểu Slice).

```
package main

import (
    "fmt"
)

type aStruc struct {
    person string
    height int
    weight int
}

func main() {
    pSlice := []aStruct{}

    pSlice = append(pSlice, aStruc{"Thạch", 165, 72})
    pSlice = append(pSlice, aStruc{"Ngọc", 170, 77})
}
```

```
    fmt.Println(pSlice)
}
```

Ghi chú:

- Cú pháp `[]aStruct{}` khai báo Slice của struct “aStruct”.
- Hàm `append(slice, struct)` trả lại slice mới sau khi thêm struct vào cuối slice.

Con trỏ (Pointer)

Nếu bạn nào đã học lập trình C thì sẽ biết đến khái niệm con trỏ. Trong trường hợp bạn nghe khái niệm con trỏ (Pointer) lần đầu thì hiểu như sau: Khi bạn khai báo một biến (variable) thì có nghĩa là máy tính sẽ tạo một vùng nhớ trong thanh RAM và đặt tên vùng nhớ đó dưới dạng một cái tên (tên biến) để cho bạn lưu trữ dữ liệu tạm trong quá trình chương trình thực thi (Xem lại Bài 2).

Thông thường thì bạn chỉ cần biết tên biến và lấy dữ liệu của biến đó, hoặc thiết lập dữ liệu vào biến đó. Tuy nhiên trong vài tình huống đặc biệt thì bạn lại cần truy cập đến địa chỉ của vùng nhớ (memory address). GOLANG cung cấp 2 cú pháp để bạn làm việc với con trỏ:

- *: dấu sao để lấy giá trị của vùng nhớ mà con trỏ đang chỉ đến
- &: để lấy địa chỉ của vùng nhớ (memory address) của biến bình thường (biến không phải con trỏ)

Phân tích đoạn chương trình sau:

```
package main

import "fmt"

func main() {
    i := -10
    j := 25
    pI := &i
    pJ := &j
    fmt.Println("pI memory:", pI)
    fmt.Println("pJ memory:", pJ)
    fmt.Println("pI value:", *pI)
    fmt.Println("pJ value:", *pJ)
}
```

Kết quả:

```
pI memory: 0xc000012090
pJ memory: 0xc000012098
```

```
pI value: -10  
pJ value: 25
```

Hãy thử phép gán sau và in ra giá trị của biến i sau đó:

```
*pI = 11
```

Con trỏ đến cấu trúc

Phân tích chương trình sau đây:

```
package main  
  
import "fmt"  
  
type aPerson struct {  
    Name  string  
    Weight int // kg  
    Height int // cm  
}  
  
func changeInfo(p aPerson) {  
    p.Height += 1  
}  
  
func main() {  
    hai := aPerson{"Nguyễn Văn Hải", 74, 168}  
  
    fmt.Println(hai)  
  
    changeInfo(hai)  
  
    fmt.Println("Sau khi gọi hàm changeInfo")  
    fmt.Println(hai)  
}
```

Kết quả:

```
{Nguyễn Văn Hải 74 168}
```

```
Sau khi gọi hàm changeInfo
```

```
{Nguyễn Văn Hải 74 168}
```

Nhận xét:

- Nếu có nhu cầu viết một hàm để thay đổi giá của của struct được truyền từ tham số thì cách truyền biến thông thường sẽ không có tác dụng.

Cách xử lý vấn đề trên như sau.

Phân tích chương trình sau:

```
package main  
  
import "fmt"  
  
type aPerson struct {
```

```
Name    string
Weight  int // kg
Height  int // cm
}

func changeInfo(p *aPerson) {
    (*p).Height += 1
}

func main() {
    hai := aPerson{"Nguyễn Văn Hải", 74, 168}

    fmt.Println(hai)

    changeInfo(&hai)

    fmt.Println("Sau khi gọi hàm changeInfo")
    fmt.Println(hai)
}
```

Kết quả:

```
{Nguyễn Văn Hải 74 168}
Sau khi gọi hàm changeInfo
{Nguyễn Văn Hải 74 169}
```

Chương trình này có một chút cải tiến:

- Sử dụng con trỏ cho tham số trong hàm `changeInfo`:

```
func changeInfo(p *aPerson)
```

- Để thay đổi giá trị của tham số `p` thì dùng cú pháp dấu `*` để lấy ra giá trị của struct. Sau đó thay đổi giá trị của thuộc tính của struct bằng phép gán bình thường:

```
(*p).Height += 1
```

Phép `+=` có nghĩa là cộng cho chính nó. Tức là lệnh trên có nghĩa là cộng thêm cho thuộc tính (property, gọi là biến – variable cũng được) `Height` của struct `aPerson` (mà biến `p` trỏ đến) lên 1 đơn vị.

Tuples (Bộ dữ liệu)

Trong ngày 2, bạn đã làm quen với khái niệm hàm (function), lúc đó chỉ là hàm `calAge` đơn giản nhận một tham số là năm sinh và trả về một số nguyên là số tuổi.

```
func calAge(birthYear int) int {
    return 2020 - birthYear
}
```

Nhu cầu thực tế có thể phức tạp hơn như yêu cầu hàm trả nhiều thông tin hơn. Khảo sát đoạn chương trình sau:

```
package main

import (
    "fmt"
)

func calAge(birthYear int) (int, string) {
    return 2020 - birthYear, "Đinh Ty"
}

func main() {

    myAge, moonAge := calAge(1977)

    fmt.Println(myAge, " ", moonAge)
}
```

```
43 Đinh Ty
```

Bạn để ý lúc này hàm `calAge` không phải trả lại một số nguyên (tuổi) nữa mà có thêm một chuỗi cho biết tuổi theo 12 con giáp. Cú pháp của kết quả trả về của hàm được bao đóng trong cặp dấu ngoặc như thế này:

(int, string)

Các viết lệnh `return` trong hàm cũng có chút khác biệt là

`return value1, value2`

Bộ giá trị `value1, value2` (có thể có nhiều giá trị hơn nữa) gọi là `tuple` (bộ)

Mình họa hàm `strconv.Atoi`

Hàm `Atoi` trong thư viện `strconv` sẽ chuyển một chuỗi các kí tự thành số. Bạn có thể tra cứu thư viện này tại "<https://golang.org/pkg/strconv/>".

Bạn sẽ thấy rằng hàm `atoi` sẽ trả lại một bộ gồm 2 giá trị với cú pháp sử dụng như sau:

```
n, err := strconv.Atoi(string)
```

Trong đó tham số `string` là biến có kiểu `string` hoặc là `literal string` bao đóng với cặp nháy đôi.

n và err lần lượt là 2 biến kết xuất: giá trị số bạn cần nhận, thông báo lỗi (nếu có)

Khảo sát ví dụ sau:

```
package main

import (
    "fmt"
    "strconv"
)

func main() {

    n, err := strconv.Atoi("123A")
    fmt.Println("Lỗi: ", err)
    fmt.Println("n: ", n)
}
```

```
Lỗi: strconv.Atoi: parsing "123A": invalid syntax
n: 0
```

Hãy sử dụng tham số "123A" thành "123" thì kết quả sẽ như sau:

```
Lỗi: <nil>
n: 123
```

Vài nhận xét:

- Khi dữ liệu hợp lệ thì err sẽ bằng nil. Đây là giá trị đặc biệt có nghĩa là "không có gì cả". Các ngôn ngữ lập trình khác như C, C++, Java, Python gọi là Null.
- Khi dữ liệu không hợp lệ thì err sẽ chứa thông báo cụ thể. Tức là khác nil

Đọc thêm và thực hành

Chuỗi (String)

Có thể xem String là thông tin phức hợp vì nó được tạo thành từ các kí tự (char).

Trong Bài 5, bạn đã làm quen với kiểu chuỗi với vài thao tác đơn giản. Phần này sẽ giúp các bạn mở rộng thêm kiến thức của mình trong việc khai thác kiểu dữ liệu chuỗi.

Chuyển đổi kiểu chuỗi thành số

Một tình huống đặt ra cho các bạn là khi viết chương trình cần nhận tham số đầu vào từ dòng lệnh có ý nghĩa là số như ví dụ sau: Bạn cần viết chương trình tính toán năm sinh dương lịch để hiển thị ra năm âm lịch theo con giáp. Ví dụ năm 1984 là năm Giáp Tý. Cách chạy chương trình bằng lệnh như sau:

```
amlich 1984
```

Chương trình sẽ hiển thị ra chữ:

```
Giáp Tý
```

Như vậy bạn cần áp dụng kiến thức của Bài 4 để biết cách lấy tham số từ dòng lệnh bằng cách truy xuất mảng `os.Args`. Tuy nhiên khi lấy tham số được truyền từ dòng lệnh như `os.Args[1]` thì kết quả là một String.

Để chuyển từ string sang kiểu số thì dùng thư viện `strconv` (viết tắt của string conversion). Bạn tập xem tài liệu tại:

<https://godoc.org/strconv>

Hãy tra cứu thêm tài liệu tại trang "<http://buaphep.net/2020/02/06/cach-chuyen-doi-nam-duong-lich-sang-nam-am-lich/>" để hoàn thành chương trình Âm Lịch ở trên.

Sử dụng các hàm thông dụng về String

Regular expressions and pattern matching

Tạm dịch mục này là Biểu thức chính qui và so trùng chuỗi. Hơi khó hiểu phải không? Hãy xem nhu cầu sau đây:

Đôi lúc bạn cần tìm kiếm hoặc nhận diện một phần của chuỗi theo một quy tắc nào đó. Các quy tắc được biểu diễn dưới dạng biểu thức gọi là biểu thức chính quy (Regular expression). Ví dụ:

Biểu thức “H\d” ý nói là một chuỗi có dạng H1 hoặc H2 ... hoặc H9. Tức là sau chữ H là một kí số. Kí hiệu \d ý nói là 1 digit character.

Kí tự xuyệt trái (back slash) \ là một kí tự đặc biệt trong chuỗi. Dấu xuyệt này thường được dùng để kết hợp với một kí tự tiếp theo để thể hiện một ý nghĩa đặc biệt nào đó. Ví dụ: \n là biểu diễn kí tự xuống hàng.

Trong trường hợp một chuỗi có nội dung là “\n” tức gồm 2 kí tự là dấu xuyệt và n thì làm sao? Vì nếu viết lệnh `fmt.Println("\n")` thì máy tính sẽ in ra kí tự xuống hàng (kí tự này bạn không thấy bằng mắt mà thực sự là con trỏ đánh dấu chỗ hiển thị kí tự trên màn hình sẽ xuống 1 hàng để chuẩn bị hiển thị nội dung cho các lệnh Print tiếp theo). Để giải quyết tình huống này thì tác giả ngôn ngữ lập trình GO qui ước là dùng thêm một kí tự xuyệt trái nữa. Cụ thể là kí tự ‘\’ được thể hiện trong chuỗi là “\\”. Chuỗi “\n” được biểu diễn là “\\n”

Đoạn chương trình sau đây sử dụng hàm **MatchString** trong thư viện `regexp` để kiểm tra một chuỗi trong biến `st` có xuất hiện chuỗi H0 hoặc H2 ... hoặc H9 không (sau H là một kí số từ 0 đến 9)

```
package main

import (
    "fmt"
    "regexp"
)

func main() {
    st := "H1"
    match, _ := regexp.MatchString("H\\d", st)
    fmt.Println(match)
}
```

Kết quả là biến `match` sẽ trả lại là: `true`

Hãy thử thay đổi biến `st` với các giá trị sau và quan sát kết quả của biến `match`:
aH1, H1b, aH1b, h1, ah1, ah1b

Từ đó rút ra nhận xét cho riêng mình.

Hãy đọc tài liệu về Regular Expression trong GO tại trang web:

<https://golang.org/pkg/regexp/>

Trong đó hàm `MatchString` được giải thích như sau:

func MatchString

```
func MatchString(pattern string, s string) (matched bool, err error)
```

MatchString reports whether the string *s* contains any match of the regular expression pattern. More complicated queries need to use *Compile* and the full *Regexp* interface.

Hãy đọc và thử thực hành các hàm khác.

Bài 2 – Viết hàm cho cấu trúc

Khảo sát đoạn chương trình sau:

```
package main

import (
    "fmt"
)

type Employee struct {
    name    string
    height  int
    weight  int
}

func (e Employee) calculateBMI() float64 {
    heightMet := float64(e.height) / 100.0
    return float64(e.weight) / (heightMet * heightMet)
}

func main() {
    p1 := Employee{"Thạch", 165, 72}

    fmt.Println(p1)
    bmi := p1.calculateBMI()

    fmt.Printf("BMI of %s is %f", p1.name, bmi)
}
```

Phân tích hàm calculateBMI cho struct Employee

Chú ý cách gọi hàm tính chỉ số BMI cho biến p1:

```
bmi := p1.calculateBMI()
```

Biến p1 có kiểu là Employee tức diễn đạt theo ngôn ngữ tự nhiên p1 là một Nhân viên có tên là Thạch, cân nặng là 72kg, chiều cao là 1m65. Để tính chỉ số BMI của nhân viên thì gọi hàm calculateBMI của nhân viên đó. Đây chính là tư tưởng của lập trình hướng đối tượng (Object Oriented Programming). Việc gọi hàm chính là gửi thông điệp (send message) cho đối tượng (ở đây là biến p1).

Để lập trình được hàm cho cấu trúc như trên thì bạn chú ý cách viết hàm calculateBMI như sau:

```
func (e Employee) calculateBMI() float64 {
}
```

Cú pháp **func** dùng để định nghĩa hàm như bình thường.

Điểm mới ở đây là cú pháp giống như khai báo biến trước tên hàm:

(e Employee)

Chỗ này có nghĩa là biến `e` có kiểu là `Employee`.

Phần tiếp theo giống khi khai báo hàm bình thường:

`calculateBMI() float64`

Tóm lại ý nghĩa của hàm trên diễn đạt như sau: khi báo hàm cho biến mang tính đại diện tên là `e` có kiểu là cấu trúc `Employee`; tên hàm là `calculateBMI`; kết quả hàm trả lại là số thực 64 bit.

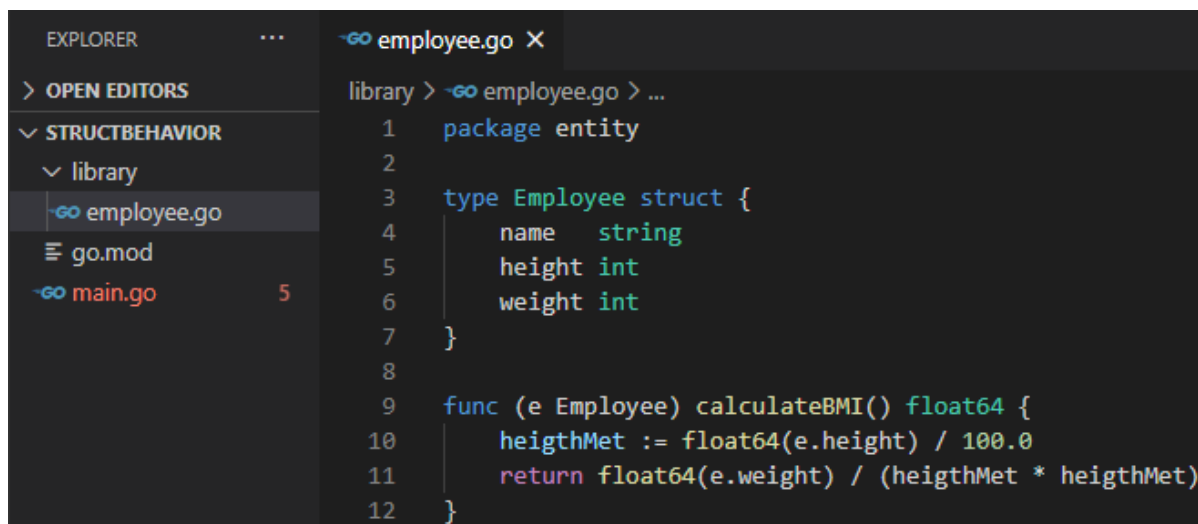
Tổ chức thành thư viện (module)

Tình huống tiếp theo cho bạn là cần tổ chức mã nguồn của cấu trúc `Employee` thành module để có thể dùng lại.

Đúng trong thư mục của dự án, tạo file `go.mod` bằng lệnh sau:

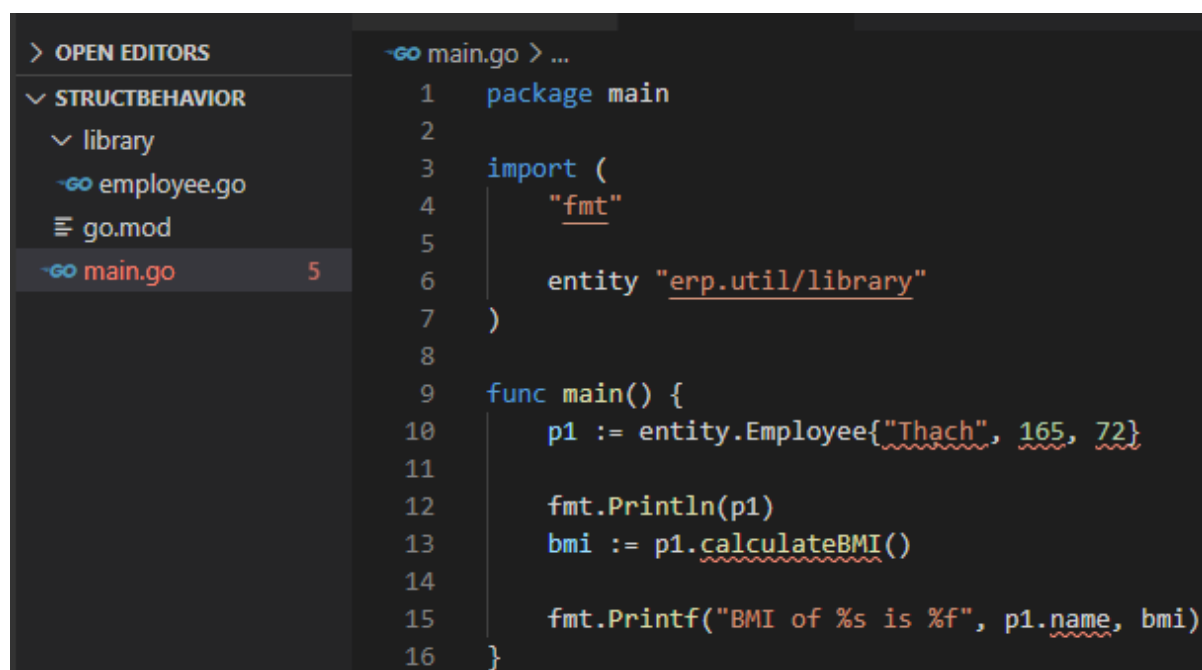
```
go mod init erp.util
```

Tiếp theo tạo thư mục “library” và file `library\employee.go`. Chuyển mã nguồn của khai báo cấu trúc `Employee` và hàm `calculateBMI` từ file `main.go` vào file `library\employee.go` như sau:



```
EXPLORER
> OPEN EDITORS
STRUCTBEHAVIOR
  library
    -go employee.go
    go.mod
    -go main.go 5
library > -go employee.go > ...
1 package entity
2
3 type Employee struct {
4     name    string
5     height  int
6     weight  int
7 }
8
9 func (e Employee) calculateBMI() float64 {
10     heighthMet := float64(e.height) / 100.0
11     return float64(e.weight) / (heighthMet * heighthMet)
12 }
```

Mã nguồn của file `main.go` khai báo sử dụng package “entity” trong thư viện “erp.util/library” như sau:



```
> OPEN EDITORS
STRUCTBEHAVIOR
  library
    employee.go
    go.mod
    main.go 5
main.go
1 package main
2
3 import (
4     "fmt"
5
6     entity "erp.util/library"
7 )
8
9 func main() {
10     p1 := entity.Employee{"Thach", 165, 72}
11
12     fmt.Println(p1)
13     bmi := p1.calculateBMI()
14
15     fmt.Printf("BMI of %s is %f", p1.name, bmi)
16 }
```

Về mặt cấu trúc mã nguồn thì tạm ổn. Tuy nhiên bạn sẽ thấy mã nguồn `main.go` bị lỗi ở các dòng 10, 13, 15 trong việc sử dụng cấu trúc và hàm của cấu trúc. Lý do là các biến của cấu trúc và tên hàm bắt đầu bằng **chữ thường**.

Để truy cập được các biến của cấu trúc và tên của hàm thì bạn cần điều chỉnh lại phạm vi (scope) của cấu trúc bằng cách sử dụng kí tự Hoa đầu tiên.

Cụ thể mã nguồn của file `employee.go` được chỉnh lại như sau:

```
package entity

type Employee struct {
    Name    string
    Height  int
    Weight  int
}

func (e Employee) CalculateBMI() float64 {
    heighthMet := float64(e.Height) / 100.0
    return float64(e.Weight) / (heighthMet * heighthMet)
}
```

Chú ý những chỗ viết Hoa và in đậm.

Quy tắc sử dụng kí tự in Hoa đầu tiên cho các biến, tên hàm (gọi chung là **identifer**) trong các module được khái quát lên như sau:

Trong GOLANG, để các định danh (tên cấu trúc, tên biến, tên hàm, v.v...) trong một module được truy cập được từ bên ngoài thì cần viết Hoa kí tự đầu tiên.

Code trong file `main.go` sửa lại như sau:

```
package main
```

```
import (  
    "fmt"  
  
    entity "erp.util/library"  
)  
  
func main() {  
    p1 := entity.Employee{"Thạch", 165, 72}  
  
    fmt.Println(p1)  
    bmi := p1.CalculateBMI()  
  
    fmt.Printf("BMI of %s is %f", p1.Name, bmi)  
}
```

Bài 3 – Dữ liệu dạng JSON

JSON là một dạng tài liệu văn bản rất phổ biến để giúp các lập trình viên thực hiện xử lý dữ liệu trong JavaScripts.

Đọc dữ liệu JSON