git init - 저장소 만들기 .git파일

git pull

git push origin main

git push -u origin main

git remote add origin site

git remote set-url origin site

git remote -v

git status

go run ./파일명

go mod init 디렉터리명/

go build ./파일명

go fmt

go version

ls

-a

-l

-al

cd

-..

pwd

git add .

git commit -m ""

git config --global user.name

git config --global user.email

git config --global --list

git log -- commit 과정

echo

echo " ">> ???

cat ??? cat > 파일명

git branch -m main

mkdir

rmdir

git checkout

--branch가 여러 개일 때 branch 바꿈

**Go language**

* 미사용 메모리를 자동 해제한다.
* 편리한 동시성 코드를 작성한다
* 멀티코어 프로세서를 지원한다.

1. Go는 정적 타입의 언어이며 Garbage Collection 기능을 제공하고 CSP(Communicating Sequential Processes) 스타일의 Concurrent 프로그래밍을 지원한다.

Package 패키지

* 함수들이 소속된 곳

package : 문자열 서식 기능 모음이나 이미지 그리기 기능 모음과 같은 유사한 기능을 수행하는 코드

import : 다른 패키지에 있는 코드를 사용하기 위해선 먼저 해당 패키지를 가져와야 한다

-미사용 패키지를 불러오면 오류남!

-패키지의 함수를 쓸 때

ex) fmt.Println()으로 .을 붙여줌

--fmt -> format

function : 코드의 어딘가에 호출할 수 있는 한 줄 이상의 코드로 이루어진 코드의 집합

- first find main!

세미콜론을 써도 괜찮고 안써도 괜찮다

Println 출력 후 한 줄 넘어가기

-괄호는 필수

-하나 이상의 인자를 받을 수 있음 ","로 구분

* 1. 주석

/\* \*/

------------

2. 변수 선언

Var 변수명 변수타입

Var 변수명 변수타입 = 변수 초기값

Var 변수명 = 변수 초기값

----

변수 지정의 또 다른 방식 short Assingment Statement(:=) 를 사용할 수 있다.

* Var i = 1 -> i:=1로 대신 사용이 가능하다

----------------------

상수 선언

Const 상수명 상수타입 = 상수값

* 상수는 값이 변할 수 없으므로 반드시 처음 지정할 때 반드시 지정해줘야 함

여러 개의 상수를 묶어서 지정 가능

* Const(

Visa = “Visa”

Master = “MasterCard”

Amex = “American Express”

)

Itoa라는 identifier를 사용하면 첫 값이 0을 할당받고 나머지 순서

변수와 상수는 함수 밖에서도 사용할 수 있다.

3. Go programming type

1. Bool type
2. String : string은 한 번 생성되면 수정될 수 없는 immutable type임
3. Int, int8, int16, int 32, int64

Unit, unit8, unit16 unit32, unit64, unitptr

1. Float및 복소수 type

Float32, float64, complex64, complex128

1. 기타 타입

Byte : unit8과 동일하며 바이트 코드에 사용

Rune : int32와 동일하며 유니코드 코드 포인터에 사용

\*\*Go에서 파일을 읽고 쓰기 위해 Go 표준 패키지인 os 패키지를 사용할 수 있다.

4. 문자열 리터럴은 Back Quote(` `)혹은 이중인용부호(“ “)를 사용하여 표현할 수 있다.

* (` `)안에 문자열은 Raw String Literal이라고 불리며 별도 해석 없이 Raw String 그대로의 값을 가진다.
* (“ “)안에 문자열은 Interpreted String Literal이라 부르는데, 복수 라인에 걸쳐 쓸 수 없고 문자열이 해석된다.

5. 데이터 타입 변환

* 하나의 데이터 타입에서 다른 데이터 타입으로 변환하기 위해서는 T와 같이 표현,
* 타입간 변환은 명시적으로 지정해 주어야 한다는 점
  + Var I int = 100
  + Var u unit = unit(i)

6. Go의 연산자

- 산술연산자(+,-,/,\*,%), (++,--)

- 관계연산자(==, <, >, <=, >=, !=)

- 논리연산자(!, &&, ||)

- Bitwise 연산자

- 비트단위 연산을 위해 사용되는데, 바이너리, AND, OR, XOR 바이너리 시프트 연산자가

있다. ->ex) c = (a & b) << 5

- 할당연산자

-값을 할당하는 = 연산자 외에 사칙연산, 비트연산을 축약한 (+=, &=, <<=)같은 연산자

- 포인터연산자

* C++와 같이 &혹은 \*을 사용해 해당 변수의 주소를 얻어내거나 이를 반대로 Dereference 할 떄 사용한다.

-- var k int = 10

-- var p = &k -> k의 주소를 할당

-- println(\*p) -> p가 가리키는 주소에 있는 실제 내용 출력

7. if문

- 조건이 맞으면 { }안의 내용을 실행한다.

- 조건식은 반드시 Boolean 식으로 표현되어야 한다.

8. switch문

- 여러 값을 비교해야 하는 경우, 다수의 조건식을 체크해야 하는 경우

- 하나의 변수를 지정하고, case 문에 해당 변수가 가질 수 있는 값들을 지정하여 내려갈 수 있으며 하나의 case에 여러 개의 값을 넣어 놓을 수 있다.

9. for문

- for 초기값; 조건식; 증감;{ }의 형식을 따른다.

* Go 에서 for 루프는 초기값과 증감식을 생략하고 조건식만 사용할 수 있다.
* 초기값, 증감식, 조건식을 모두 제거하면 무한 루프가 된다.

변수 명을 이미 있는 패키지나 함수 등으로 사용하게 된다면 원래의 의미와 이름을 잃으므로 조금이라도 다르게 선언해야 한다.

ex) Fatal - fatal

키보드로 처음 입력받은 값은 문자열이다.

- 사용자가 값을 입력하고 엔터를 누를 때 추가되는 줄 바꿈 문자를 없애야 한다.

- 문자열을 부동 소수점 숫자 타입으로 변환해야 한다.

--------------------------------------------------

return 반환 값

-함수 안에서 사용

-마지막 값을 반환해주는 것

-선언한 반환 타입과 반환 값의 타입은 같아야함

--------------------------------------------------

사용자 지정 함수

func <함수명>(매개변수명 매개변수타입) 리턴 타입{

}

->매개변수는 여러개 지정 가능

사용자 지정 함수 네이밍 규칙

-이름은 문자로 시작해야하며 임의 개수의 문자와 숫자로 구성될 수 있다.

-함수명이 대문자로 시작하면 외부로 노출되어 외부 패키지에서 호출 가능

-이름이 여러 단어 일 때는 캐멀

----------------------------------

string : 텍스트 문자를 나타내는 일련의 바이트

literal : 입력 값

rune : 단일 문자를 나타내는데 사용, 유니코드, 문자를 입력하면 유니코드 문자 코드가 나옴

* 작은 따옴표 사용, 표준 유니코드를 사용하기 때문에 go 프로그램에서는 지구상에 존재하는 거의 모든 언어에 포함된 거의 모든 문자를 사용할 수 있음.

boolean : true, false를 가지는 type

산술 연산자 : +, -, \*, /

비교 연산자 : ==, !=, <, >, <=, >= 결과가 true, false로 나옴

type이 바뀌면 에러 메세지가 뜬다.

-> go는 staticlly typed이기에 실행되기 전 값의 타입을 미리 알아낼 수 있고 함수는 특정 타입의 인자를 받고 특정 타입의 값을 반환

append : append([]string{}, "Espanol")

-> [espanol]

---------------------------------------

Type

-int

-int32

-float32

-float64

->완벽하지는 않지만 상당한 정밀도가 있음

-bool

-string

---------------------------------------------------

변수 선언

var <variable name> <variable's type>

or

var <variable name> <variable's type> = <variable's result>

or

<variable> := variable's result

-> 변수의 타입은 자동 지정

-여러 개 나열 가능

----------------------------------------------------------------

zero value : 값을 할당하지 않고 선언한 변수

Naming rule's

-이름은 문자로 시작해야 하며 임의 개수의 문자와 숫자로 구성될 수 있음

-대문자로 시작하면 외부로 노출되어 외부 패키지에서 접근 가능 [fmt.Println()]

Type Change

-<변활할 타입>(변환값)

ex) var myInt int = 2

float(myInt)

-사용 할 때마다 변환해주고 사용해야함

주석

// 한 줄 주석

/\* \*/여러 줄 주석

------------------------------------------------------------

for x:=?; x>=?; x++{

}

continue

break

condition

-------------------------------------------------------

-if

if <conditional word> {}

-true인 경우 실행

-false면 실행하지 않음

-----------------------------------------------------------------

scope : 코드에서 변수가 보이는 (변수에 접근할 수 있는) 영역의 범위

단축 변수 선언에서 변수는 단 한 번만 새로 선언된다.

-> 동일한 스코프에서 동일한 이름의 변수를 두 번 선언하면 컴파일 에러가 발생

->단축 변수 선언에서 하나 이상의 변수가 새로운 변수인 경우에는 가능

----------------------------------------------

printf : print, with formmatting

Sprintf

-형식동사

문자열에 나타나는 대부분의 문자는 문자 그대로 출력 but 백분율 기호%로 시작하는 형식동사

%f -> 여섯 번 째 자리

%d

%s

%t (boolean)

%v 값의 타입에 따라 적절히 형식화된 값

%#v Go 프로그램 코드에 나타나는 그대로 형식화된 값

%T 제공된 값의 타입

%% 백분율 기호

--------------------------

함수 사용법

-> <package name>.<function name>

메서드 사용법

-> <package name>.<function name>.<method>

\*함수 안에 함수 사용가능

\*호출한 함수로부터 결괏값을 반환받아야 하는 경우도 있기 때문에 대부분의 프로그래밍 언어에서 함수는 리턴 값, 함수를 호출한 호출자에게 반환하는 값을 가질 수 있다.

ex) math.Floor, strings.Title

package

-fmt

-Print -> 출력

-Println -> 출력 후 다음줄로

-Scan -> 정수 입력받음

-strings

-Title -> 각 단어 첫번째 대문자로

-Replacer -> 바꿈

-Replace

-TrimSpace -> 필요 없는 줄 띄움 없애기

-math

-Floor -> 내림

-Ceil -> 올림

-Round -> 반올림

-reflect

-Typeof -> 타입

-time

-Time -> 시간

-Year

-Now ->

-Unix

-os

-Stdin -> 표준입력

-Open -> 열기

-create ->만들기

-Read

-Write

-exit -> 나가기

-bufio

-Reader -> 문자열로 입력받기

-ReadString

-log

-Fatal -> 오류 검출

-append

-strconv

-ParseFloat -> float64로 바꿔줌

->strconv.ParseFloat(반환 문자열, 결괏값의 정밀도 비트 수)

-math/rand

-Intn ->(?) 0부터 ?-1까지 중 하나

-Seed ->

-errors

-New