2020년 10월 22일 목요일 오후 9:01

```
package main

import (
    "fmt"
    "mathcalculator"
    "os"
)

func main()  {
    for x := 1; true; x++ {
        fmt.Print("1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : ")
        // 1~10번까지 출력해주면서 하고 싶은 기능을 선택한다.
        n := mathcalculator.Input()
        //1번 절대값 계산으로 Input을 사용하여 mathcalculator에서 Absolute를 이용하여 계산해준다
```

main.go

```
    if n == 1 {
        fmt.Print("정수 입력(절대값 계산) : ")
        fmt.Println(mathcalculator.Absolute(mathcalculator.Input()))
    }
```

mathcalculator.go

```
func Absolute(n int) int {
    if n < 0 {
        return n * -1
    }
    return n
}
```

설명

int인 n을 받아와
n이 음수면 n*(-1)을 해서 양수로 만들어주고
정수면 n으로 리턴해준다

결과

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 1
정수 입력(절대값 계산) : 5
5
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 1
정수 입력(절대값 계산) : -5
5
```

main.go

```
    else if n == 2 {
        fmt.Print("정수 입력(팩토리얼 계산) : ")
        fmt.Println(mathcalculator.Factorial(mathcalculator.Input()))
    }
```

mathcalculator.go

```
func Factorial(n int) int {
    if n == 0 {
        return 1
    }
    return n * Factorial(n-1)
}
```

설명

int인 n을 받아와 팩토리얼 계산을 해주는데
n이 0이면 1을 리턴해주고 아닐 경우
팩토리얼 이므로 n*(n-1)!로 진행되어
n*Factorial(n-1)을 리턴해준다

결과

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 2
정수 입력(팩토리얼 계산) : 3
6
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 2
정수 입력(팩토리얼 계산) : 0
1
```

main.go

```
    else if n == 3 {
        fmt.Print("정수 입력(피보나치 출력) : ")
        f := mathcalculator.Input()
        fmt.Println(mathcalculator.Fibonacci(f))
    }
```

mathcalculator.go

```
func Fibonacci(n int) int {
    if n == 1 || n == 2 {
        return 1
    } else {
        return Fibonacci(n-1) + Fibonacci(n-2)
    }
}
```

설명

피보나치수열은

$$F_n = F_{n-1} + F_{n-2}$$

로 계산하는데 이 식에 n이 1이거나 2면 Fn-(1or 2)이
음수가 되므로 1로 리턴해주고 아닌 경우 Fn-1과 Fn-2
로 리턴하여 Fn을 구하도록 해준다.

결과

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 3
정수 입력(피보나치 출력) : 5
5
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 3
정수 입력(피보나치 출력) : 8
21
```

```
    else if n == 4 {
```

```
func Power(b int, e int) int {
```

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 4
```

```
 else if n == 4 {
    fmt.Print("Base 입력 : ")
    b := mathcalculator.Input()
    fmt.Print("Exponent 입력 : ")
    e := mathcalculator.Input()
    fmt.Println(mathcalculator.Power(b, e))
}
```

```
    }

    func Power(b int, e int) int{
        r := 1
        for i := 1; i<=e ; i++{
            r = r * b
        }
        return r
    }
```

거듭제곱은 밑(b)을 지수만큼(e) 곱하는 것이므로
밑,지수 2가지 정수를 입력 받아, 지수를 횟수로 for문을 돌려 밑을 지수번 곱하여 계산해준다(r= r * b)

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 4
Base 입력 : 3
Exponent 입력 : 5
243
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 4
Base 입력 : -5
Exponent 입력 : 3
-125
```

```
else if n == 5 {
    fmt.Print("정수 입력(삼각수 계산) : ")
    t:= mathcalculator.Input()
    fmt.Println(mathcalculator.Triangular(t))
}
```

```
    func Triangular(n int) int {
        n = (n*n+n)/2
        return n
    }
```

이 계산은 n번째삼각수의 값을 얻는 것으로 n*(n+1)/2를 하여도 되고 for문을 이용하여 1부터 n까지 더해도 같은 값이 나온다.

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 5
정수 입력(삼각수 계산) : 2
3
```

```
else if n == 6 {
    fmt.Print("정수 입력(오각수 계산) : ")
    f:= mathcalculator.Input()
    fmt.Println(mathcalculator.Pentagonal(f))
}
```

```
    func Pentagonal(n int) int {
        //n = n*(n*3-1)/2
        n =Triangular(3*n-1)/3
        return n
    }
```

n번째 오각수는 3n−1번째 삼각수의 1/3과 같아서 Trianglur()를 사용하여 계산하였다

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 6
정수 입력(오각수 계산) : 3
12
```

```
 else if n == 7 {
    var num string
    fmt.Print("정수 입력(팩토리온 확인) : ")
    fmt.Scanln(&num)
    fmt.Println(" is",mathcalculator.Factorion(num),".")
}
```

```
    func Factorion(num string) string {
        ma:= map[string]int{"0":1,"1":1,"2":2,"3":6,"4":24,
        "5":120,"6":720,"7":5040,"8":40320,"9":362880}
        var s int
        for i :=0;i<len(num);i++{
            n := num[i:i+1]
            s +=ma[n]
        }
        intn := strconv.Itoa(s)
        fmt.Print(num,"(",s,")")
        if intn == num {
            return "factorion"
        }
        return "not factorion"
    }
```

각 자릿수의 계승의 합이 자기 자신이 되는 정수를 팩토리온이라고 불린다.
이 때 팩토리얼을 0~9까지만 쓰기 때문에 경우가 10가지로 map을 이용하기로 했다.
Scanln을 이용하여 문자열(num)로 받아와서 for문을 num의 길이만큼 반복하여, n에 "숫자" 형식으로 넣고, s에 더해준다.
s를 정수형으로 바꿔 만약 num과 intn이 같으면 fatorion으로 리턴이 되고 아닌 경우는 not fatorion으로 리턴된다.
ex)
1=1!1=1!
2=2!2=2!
145=1!+4!+5!145=1!+4!+5!
40585=4!+0!+5!+8!+5!

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 7
정수 입력(팩토리온 확인) : 2
2(2) is factorion .
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 7
정수 입력(팩토리온 확인) : 153
153(127) is not factorion .
```

```
 else if n == 8 {
    fmt.Print("소수 판정  입력 : ")
    n :=mathcalculator.Input()
```

```
func Prime(value int) bool {
    for i := 2; i <= int(math.Floor(float64(value) / 2)); i++ {
        if value%i == 0 {
```

소수인지 아닌지를 판정해 주는 것으로 정수 value 값을 받아 2부터 for문을 value/2까지 돌린다.
만약 value가 i로 나누어지면 소수가 아니므로 false를 출력해주고

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 8
소수 판정  입력 : 13
13 is prime number
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 8
```

```
        else if n == 8 {
            fmt.Print("소수 판정  입력 : ")
            p :=mathcalculator.Input()
            fmt.Print(p," is")
            if mathcalculator.Prime(p) ==true{
                fmt.Println(" prime number ")
            }else{
                fmt.Println(" not prime number ")
            }
        }
```

```
func Prime(value int) bool {
    for i := 2; i <= int(math.Floor(float64(value) / 2)); i++ {
        if value%i == 0 {
            return false
        }
    }
    return value > 1
}
```

소수인지 아닌지를 판정해 주는 것으로 정수 value 값을 받아 2부터 for문을 value/2까지 돌린다.
만약 value가 i로 나누어지면 소수가 아니므로 false를 출력해주고 나누어지지 않으면 true로 출력해준다.(value>1)

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 8
소수 판정  입력 : 13
13 is prime number
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 8
소수 판정  입력 : 15
15 is not prime number
```

```
        else if n == 9 {
            fmt.Print("n  : ")
            n:=mathcalculator.Input()
            fmt.Print("r  : ")
            r:=mathcalculator.Input()
            fmt.Println(mathcalculator.Combination(n,r))
        }   else if n == 10 {
            os.Exit(3)
        } else {
            fmt.Print("잘못 된 입력 값입니다. 1~10사이의 수를 입력하세요!")
        }
    }
}
```

```
    func  Combination(n int,r int)int{
        result :=Factorial(n)/Factorial(r)
        return result
    }
```

조합은 이미 팩토리얼 함수가 있으므로
nCr= n!/r! = Factorial(n)/Factorial(r)로 계산하여 리턴한다

```
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 9
n  : 5
r  : 3
20
1) 절대값 2) 팩토리얼 3) 피보나치 4) 거듭제곱 5)삼각수 6)오각수 7)팩트리온 8)소수 9)조합 10) 종료 : 9
n  : 4
r  : 4
1
```