TESTY JEDNOSTKOWE

FizzBuzz

```
func FizzBuzz(n int) string {
    const (
        Fizz = "Fizz"
        Buzz = "Buzz"
    )

    var s string

    if n%3 == 0 {
        s += Fizz
    }

    if n%5 == 0 {
        s += Buzz
    }

    return s
}
```

TestFizzBuzz

```go
func TestFizzBuzz(t *testing.T) {
    tests := []struct {
        name string
        n    int
        want string
    }{
        {name: "3", n: 3, want: "Fuzz"},
        {name: "5", n: 5, want: "Buzz"},
        {name: "15", n: 15, want: "FizzBuzz"},
        {name: "-3", n: -3, want: "Fizz"},
        {name: "0", n: 0, want: "FizzBuzz"},
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := FizzBuzz(tt.n); got != tt.want {
                t.Errorf("FizzBuzz() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

go test

```
$ go test

PASS

ok      lyt/cmd/fizzbuzz        0.001s

$ go test

--- FAIL: TestFizzBuzz (0.00s)

    --- FAIL: TestFizzBuzz/3 (0.00s)

        main_test.go:20: FizzBuzz() = Fizz, want Fuzz
    FAIL
```

Fib

```
func Fib(n int) int {
    if n < 2 {
        return n
    }

    return Fib(n-1) + Fib(n-2)
}
```

TestFib

```
func TestFib(t *testing.T) {
    tests := []struct {
        name string
        n    int
        want int
    }{
        {"1", 1, 1}, {"2", 2, 1}, {"3", 3, 2},
        {"4", 4, 3}, {"5", 5, 5}, {"6", 6, 8},
        {"7", 7, 13},
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := Fib(tt.n); got != tt.want {
                t.Errorf("Fib() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

BENCHMARKING

FibNR

```go
func FibNR(n int) int {
    var (
        f1 = 1
        f2 = 0
    )

    for i := 0; i < n; i++ {
        f1, f2 = f2, f1+f2
    }

    return f2
}
```

TestFibNR

```go
func TestFibNR(t *testing.T) {
    tests := []struct {
        name string
        n    int
        want int
    }{
        {"1", 1, 1}, {"2", 2, 1}, {"3", 3, 2},
        {"4", 4, 3}, {"5", 5, 5}, {"6", 6, 8},
        {"7", 7, 13},
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := FibNR(tt.n); got != tt.want {
                t.Errorf("FibNR() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

BenchmarkFib

```go
var result int

// https://dave.cheney.net/2013/06/30/how-to-write-benchmarks-in-go
func BenchmarkFib(b *testing.B) {
    var r int
    for n := 0; n < b.N; n++ {
        // always record the result of Fib to prevent
        // the compiler eliminating the function call.
        r = Fib(10)
    }
    // always store the result to a package level variable
    // so the compiler cannot eliminate the Benchmark itself.
    result = r
}
```

BencharkFibNR

```
func BenchmarkFibNR(b *testing.B) {
    var r int
    for n := 0; n < b.N; n++ {
        r = FibNR(10)
    }
    result = r
}
```

go test -bench=.

```
$ go test -bench=.

goos: linux

goarch: amd64

pkg: cmd/fib

cpu: 12th Gen Intel(R) Core(TM) i5-1245U

BenchmarkFib-12           6132626               196.1 ns/op

BenchmarkFibNR-12       509490894                 2.321 ns/op
    PASS
```