Мета: закріплення навичок ілюстрації організації програмних систем та оцінки часу виконання алгоритмів.

Завдання

- Для 2-гої роботи, підтвердьте лінійний час виконання вашої функції перетворення чи обчислення вхідного виразу.
- Для 3-тої роботи, побудуйте діаграму взаємодії компонентів у вашій імплементації.
- Для 4-ої роботи, побудуйте діаграму взаємодії для вашої реалізації (на ній, скоріш за все, мають опинитися компоненти парсера, черги команд, ядра цикла) та підтвердьте лінійний час роботи вашого парсера команд.

Лабораторна робота №2

Лінійний час виконання функції перетворення вхідного виразу (з постфіксного у інфіксний)

Щоб підтвердити лінійний час виконання алгоритму виконаємо заміри часу алгоритму. Для цього створимо файл bench_test.go, у якому виконаємо запуск і заміряємо час виконання з різною довжиною вхідних параметрів.

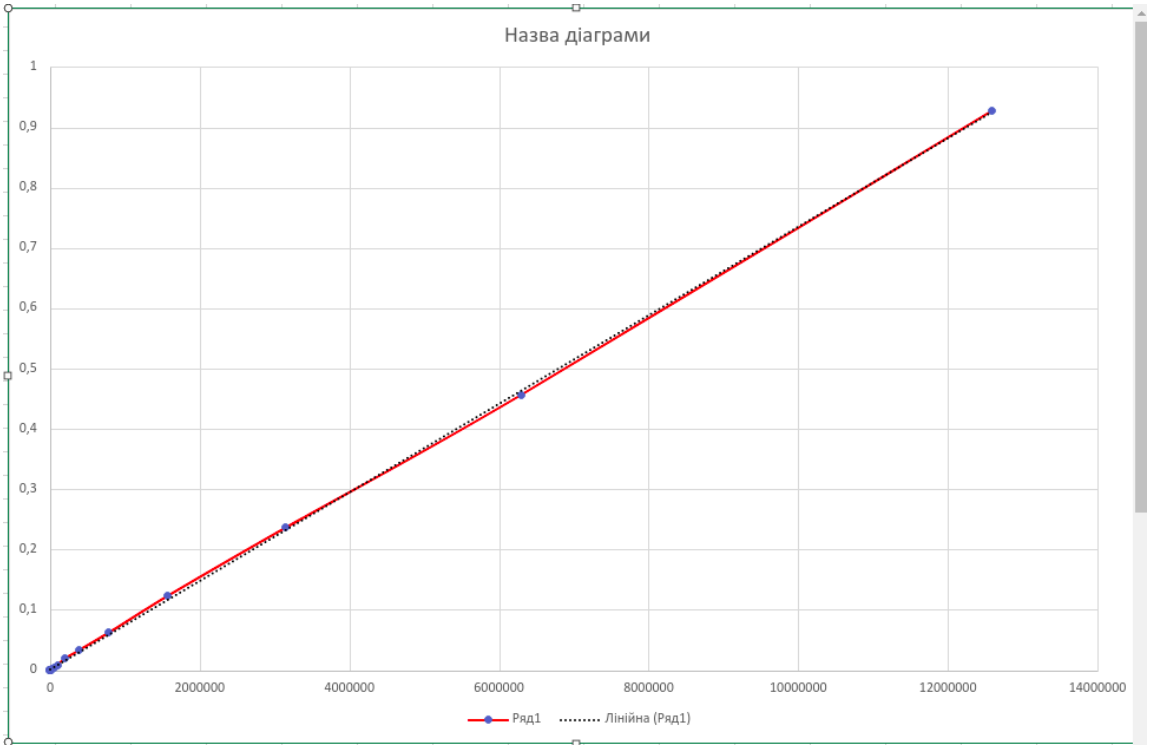
```
package APZ2

import (
    "fmt"
    "testing"
)

var cntRes string
var err error

func BenchmarkPrefixToInfix(b *testing.B) {
    const baseLen = 11 // number of operators in input
    inputLength := baseLen
    input := "+ 1 + 2 + 3 + 4 + 5 6"

    for i := 0; i < 20; i++ {
        input = "*" + input + input
        inputLength = inputLength*2 + 1
        b.Run(fmt.Sprintf("input length = %d", inputLength), func(b
*testing.B) {
            cntRes, err = PrefixToInfix(input)
        })

    }
}
```

Запустимо бенчмарк, щоб отримати часові результати виконання:

```
goos: windows
goarch: amd64
pkg: APZ2
cpu: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
BenchmarkPrefixToInfix
BenchmarkPrefixToInfix/input_length_=_23
BenchmarkPrefixToInfix/input_length_=_23-8              1000000000
BenchmarkPrefixToInfix/input_length_=_47
BenchmarkPrefixToInfix/input_length_=_47-8              1000000000
BenchmarkPrefixToInfix/input_length_=_95
BenchmarkPrefixToInfix/input_length_=_95-8              1000000000
BenchmarkPrefixToInfix/input_length_=_191
BenchmarkPrefixToInfix/input_length_=_191-8             1000000000
BenchmarkPrefixToInfix/input_length_=_383
BenchmarkPrefixToInfix/input_length_=_383-8             1000000000
BenchmarkPrefixToInfix/input_length_=_767
BenchmarkPrefixToInfix/input_length_=_767-8             1000000000
BenchmarkPrefixToInfix/input_length_=_1535
BenchmarkPrefixToInfix/input_length_=_1535-8            1000000000
BenchmarkPrefixToInfix/input_length_=_3071
BenchmarkPrefixToInfix/input_length_=_3071-8            1000000000
 0.0009984 ns/op
BenchmarkPrefixToInfix/input_length_=_6143
BenchmarkPrefixToInfix/input_length_=_6143-8            1000000000
 0.0009970 ns/op
BenchmarkPrefixToInfix/input_length_=_12287
BenchmarkPrefixToInfix/input_length_=_12287-8           1000000000
BenchmarkPrefixToInfix/input_length_=_24575
BenchmarkPrefixToInfix/input_length_=_24575-8           1000000000
 0.002655 ns/op
```

```
BenchmarkPrefixToInfix/input_length_=_49151-8          1000000000
 0.003617 ns/op
BenchmarkPrefixToInfix/input_length_=_98303
BenchmarkPrefixToInfix/input_length_=_98303-8          1000000000
 0.007470 ns/op
BenchmarkPrefixToInfix/input_length_=_196607
BenchmarkPrefixToInfix/input_length_=_196607-8         1000000000
 0.01696 ns/op
BenchmarkPrefixToInfix/input_length_=_393215
BenchmarkPrefixToInfix/input_length_=_393215-8         1000000000
 0.03590 ns/op
BenchmarkPrefixToInfix/input_length_=_786431
BenchmarkPrefixToInfix/input_length_=_786431-8         1000000000
 0.06584 ns/op
BenchmarkPrefixToInfix/input_length_=_1572863
BenchmarkPrefixToInfix/input_length_=_1572863-8        1000000000
 0.1194 ns/op
BenchmarkPrefixToInfix/input_length_=_3145727
BenchmarkPrefixToInfix/input_length_=_3145727-8        1000000000
 0.2245 ns/op
BenchmarkPrefixToInfix/input_length_=_6291455
BenchmarkPrefixToInfix/input_length_=_6291455-8        1000000000
 0.4553 ns/op
BenchmarkPrefixToInfix/input_length_=_12582911
BenchmarkPrefixToInfix/input_length_=_12582911-8               1        10725887
00 ns/op
PASS
```

На основі даних збудуємо графік:

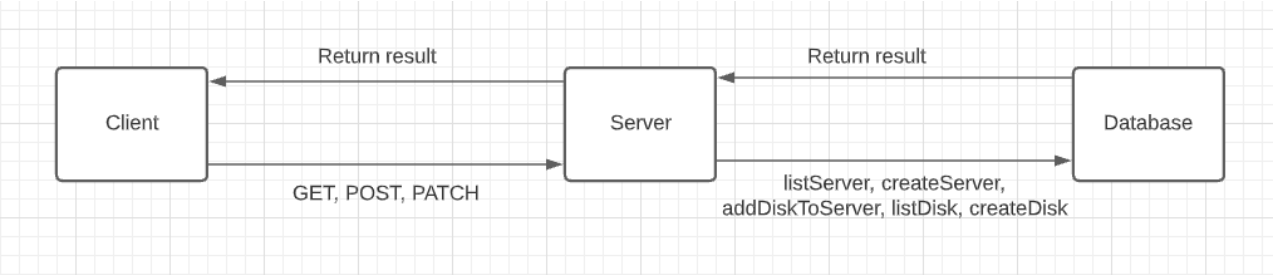


На графіку є дві лінії: червоною суцільною показано графік, що випливає з замірів часу виконання функції, а чорною пунктирною - його лінійне наближення. Оскільки графіки майже сходяться, що час виконання функції є лінійним. Проте ці значення співпадають не завжди, адже є і випадкові чинники, що впливають на час виконання (навантаження процесора).

Лабораторна робота №3

Діаграма взаємодії компонентів

Створимо діаграму взаємодії компонентів:



Клієнт надсилає запити до серверу, який, у залежності від запиту, надсилає певний інший запит до бази даних, яка повертає результат цих дій серверу, який в сіою чергу передає цей результат клієнту.

Лабораторна робота №4

Тип 1: підтвердити лінійний час виконання вашого алгоритму

Як і в 2. лабораторній роботі, створимо файл тестування, і запустимо його

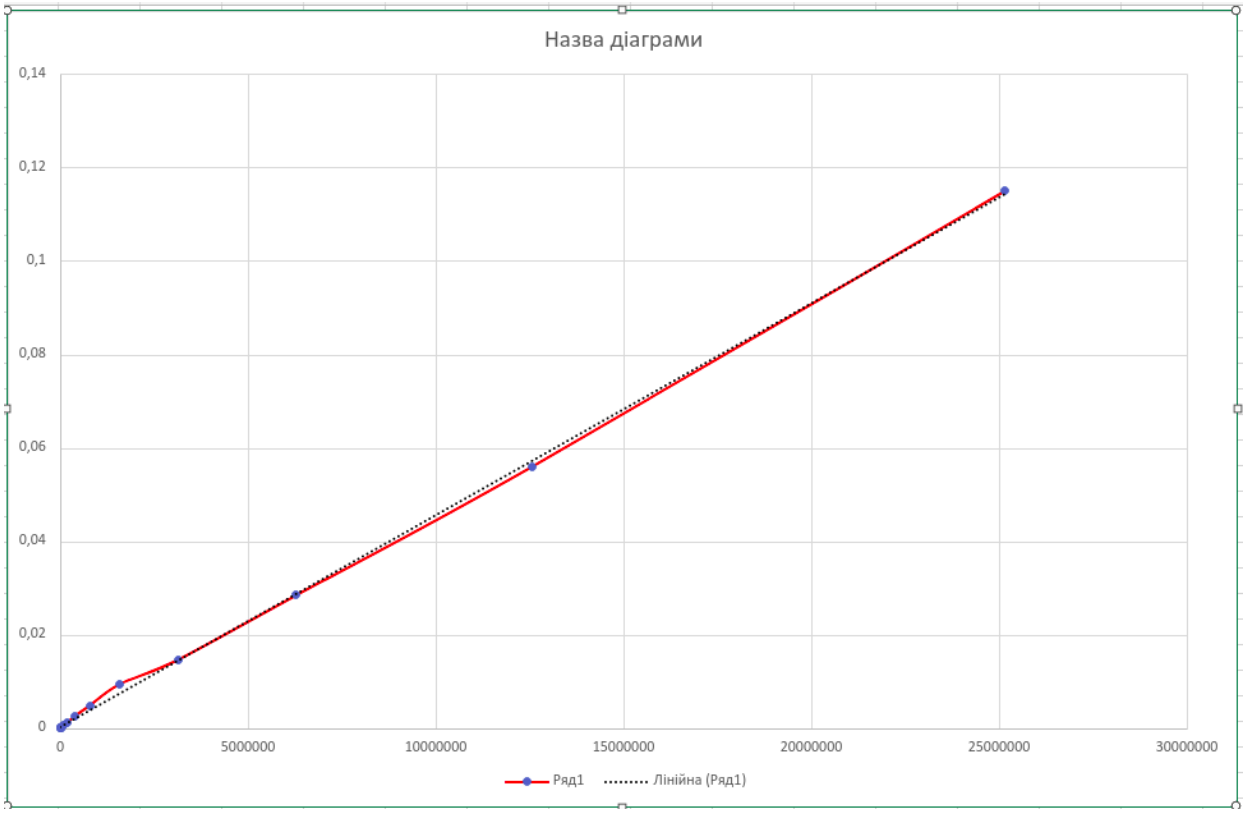
```
package engine

import (
   "fmt"
   "strings"
   "testing"
)

var constInput = "cat " //base function
var command Command

func BenchmarkCount(b *testing.B) {
   repeatNum := 1
   for i := 0; i < 20; i++ {
       repeatNum = 2 * repeatNum
       input := constInput
       input     +=     strings.Repeat("abracadabrasomethingelse",
repeatNum) + " " + strings.Repeat("abracadabrasomethingelse", repeatNum)
```

```
            b.Run(fmt.Sprintf("len=%d",      24*repeatNum),      func(b
*testing.B) {
                command = Parse(input)
            })
        }
    }
```

Результат запуску тестування:

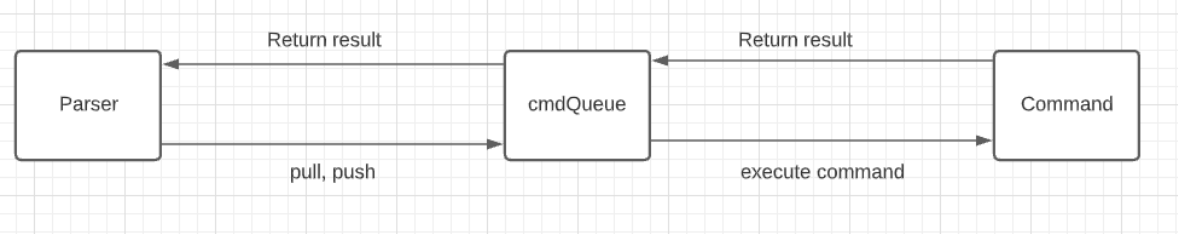
```
pkg: github.com/HohenzoIIern/APZ234/Lab4/engine
BenchmarkCount/len=48-8                 1000000000               0.000003 ns/op
BenchmarkCount/len=96-8                 1000000000               0.000003 ns/op
BenchmarkCount/len=192-8                1000000000               0.000004 ns/op
BenchmarkCount/len=384-8                1000000000               0.000005 ns/op
BenchmarkCount/len=768-8                1000000000               0.000010 ns/op
BenchmarkCount/len=1536-8               1000000000               0.000012 ns/op
BenchmarkCount/len=3072-8               1000000000               0.000022 ns/op
BenchmarkCount/len=6144-8               1000000000               0.000046 ns/op
BenchmarkCount/len=12288-8              1000000000               0.000072 ns/op
BenchmarkCount/len=24576-8              1000000000               0.000143 ns/op
BenchmarkCount/len=49152-8              1000000000               0.000283 ns/op
BenchmarkCount/len=98304-8              1000000000               0.000573 ns/op
BenchmarkCount/len=196608-8             1000000000               0.00116 ns/op
BenchmarkCount/len=393216-8             1000000000               0.00257 ns/op
BenchmarkCount/len=786432-8             1000000000               0.00478 ns/op
BenchmarkCount/len=1572864-8            1000000000               0.00932 ns/op
BenchmarkCount/len=3145728-8            1000000000               0.0146 ns/op
BenchmarkCount/len=6291456-8            1000000000               0.0284 ns/op
BenchmarkCount/len=12582912-8           1000000000               0.0562 ns/op
BenchmarkCount/len=25165824-8           1000000000               0.115 ns/op
PASS
```

Побудуємо графік, залежності часу виконання функції від довжини вхідних даних:



На графіку зображено як функцію (суцільна лінія червоного кольору), та її лінійне наближення штрихованою лінія чорного кольору. Лінії майже співпадають, отже функція має лінійний час виконання.

Тип 2: побудувати діаграму взаємодії



Парсер послідовно зчитує рядки, парсить їх (виділяє команду та її вхідні дані) та додає до черги команд. Ця черга команд чекає виконання усіх команд - для цього вона послідовно направляє дані по командах і очікує на результат. Коли всі команди виконались, виводить результат.