

Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря
Сікорського
Факультет обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ
з розрахунково-графічної роботи
з навчальної дисципліни «Архітектура програмного забезпечення»

Тема: Архітектурні діаграми та бенчмарки

Виконали
Студенти 3 курсу ІП-94
Долгова Є.
Цасюк І.
Рекечинський Д.

Перевірив
Мазур Р. Ф.

Київ 2021

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.

Частина перша

Функція для підтвердження лінійного часу виконання функції:

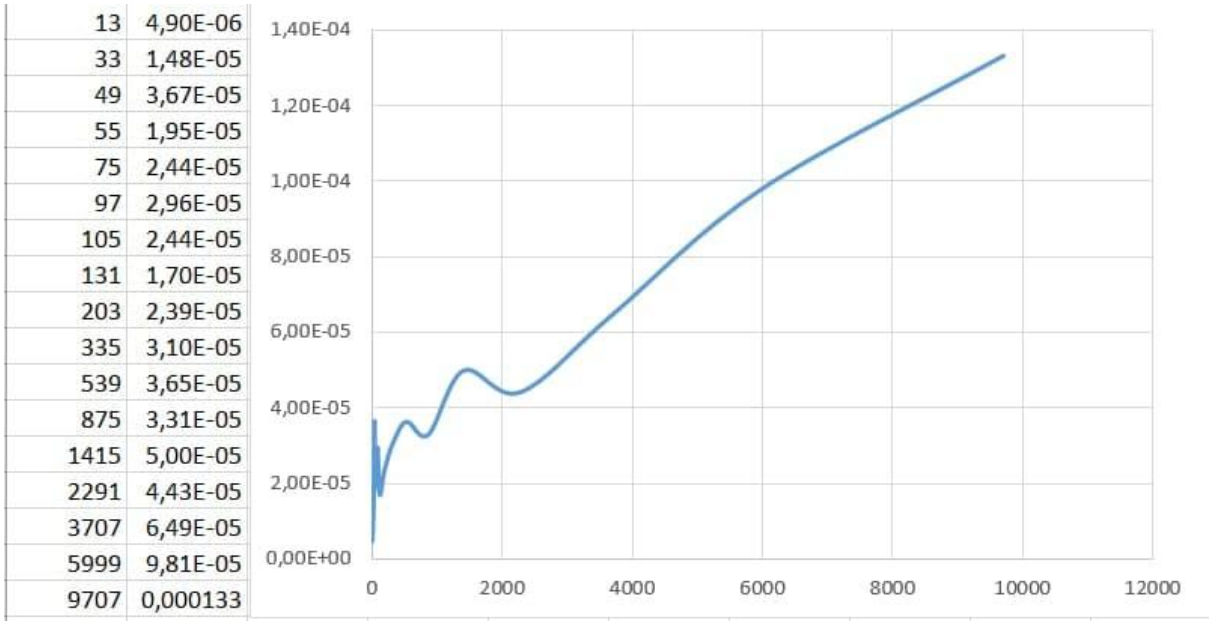
Скріншот коду:

```
1 package lab2
2
3 import (
4     "fmt"
5     "testing"
6 )
7
8 func BenchmarkPostfixToPrefix(b *testing.B) {
9     postfixes := []string{
10         "4 2 - 3 * 5 +",
11         "9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
12         "4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / +",
13         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
14         "1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 +",
15         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / +",
16         "4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
17         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4",
18         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 4 2 - 3 * 5 + 4 2 - 3",
19         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4",
20         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 4 2 - 3 * 5 + 4 2 - 3",
21         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4",
22         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 4 2 - 3 * 5 + 4 2 - 3",
23         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4",
24         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 4 2 - 3 * 5 + 4 2 - 3",
25         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4",
26         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 4 2 - 3 * 5 + 4 2 - 3",
27     }
28
29     count := len(postfixes)
30     for i := 0; i < count; i++ {
31         b.Run(fmt.Sprintf("len=%d", len(postfixes[i])), func(b *testing.B) {
32             PostfixToPrefix(postfixes[i])
33         })
34     }
35 }
```

Скріншот виконання:

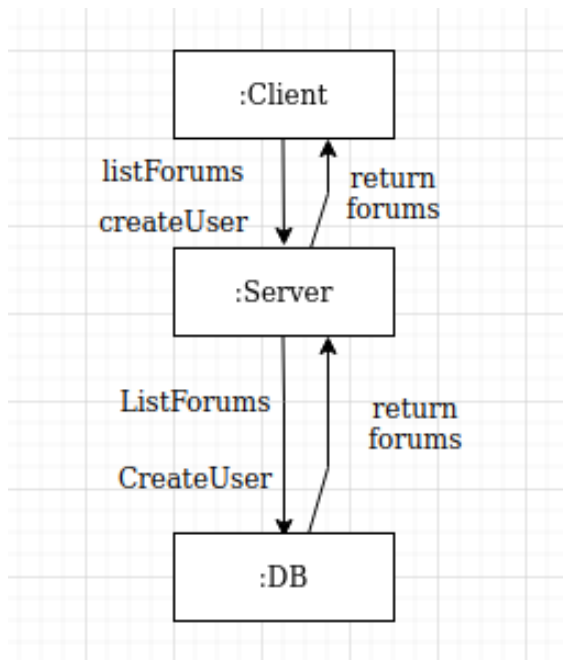
```
architecture-lab-2 : zsh — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Copy Paste Find
~/Gi/K/architecture-lab-2 maste go test -bench .
go: downloading github.com/stretchr/testify v1.7.0
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading github.com/pmezard/go-difflib v1.0.0
go: downloading gopkg.in/yaml.v3 v3.0.0-20210107192922-496545a6307b
goos: linux
goarch: amd64
pkg: software-architecture-2
cpu: AMD Ryzen 3 2200U with Radeon Vega Mobile Gfx
BenchmarkPostfixToPrefix/len=13-4      1000000000      0.0000143 ns/op
BenchmarkPostfixToPrefix/len=33-4      1000000000      0.0000095 ns/op
BenchmarkPostfixToPrefix/len=49-4      1000000000      0.0000175 ns/op
BenchmarkPostfixToPrefix/len=55-4      1000000000      0.0000247 ns/op
BenchmarkPostfixToPrefix/len=75-4      1000000000      0.0000161 ns/op
BenchmarkPostfixToPrefix/len=97-4      1000000000      0.0000198 ns/op
BenchmarkPostfixToPrefix/len=105-4     1000000000      0.0000133 ns/op
BenchmarkPostfixToPrefix/len=131-4     1000000000      0.0000185 ns/op
BenchmarkPostfixToPrefix/len=203-4     1000000000      0.0000257 ns/op
BenchmarkPostfixToPrefix/len=335-4     1000000000      0.0000243 ns/op
BenchmarkPostfixToPrefix/len=539-4     1000000000      0.0000291 ns/op
BenchmarkPostfixToPrefix/len=875-4     1000000000      0.0000244 ns/op
BenchmarkPostfixToPrefix/len=1415-4    1000000000      0.0000567 ns/op
BenchmarkPostfixToPrefix/len=2291-4    1000000000      0.0000624 ns/op
BenchmarkPostfixToPrefix/len=3707-4    1000000000      0.0000571 ns/op
BenchmarkPostfixToPrefix/len=5999-4    1000000000      0.0001073 ns/op
BenchmarkPostfixToPrefix/len=9707-4    1000000000      0.0001315 ns/op
PASS
ok      software-architecture-2 0.067s
~/Gi/K/architecture-lab-2 maste 14s
```

Графік результатів:



Частина друга

Діаграма взаємодії компонентів:



Система форумів. Користувач може створити нового користувача або побачити список форумів.

Клієнт надсилає запити до серверу. Сервер його обробляє та записує дані у базу або отримує дані з бази.

Частина третя

Функція для підтвердження лінійного часу роботи парсера команд:

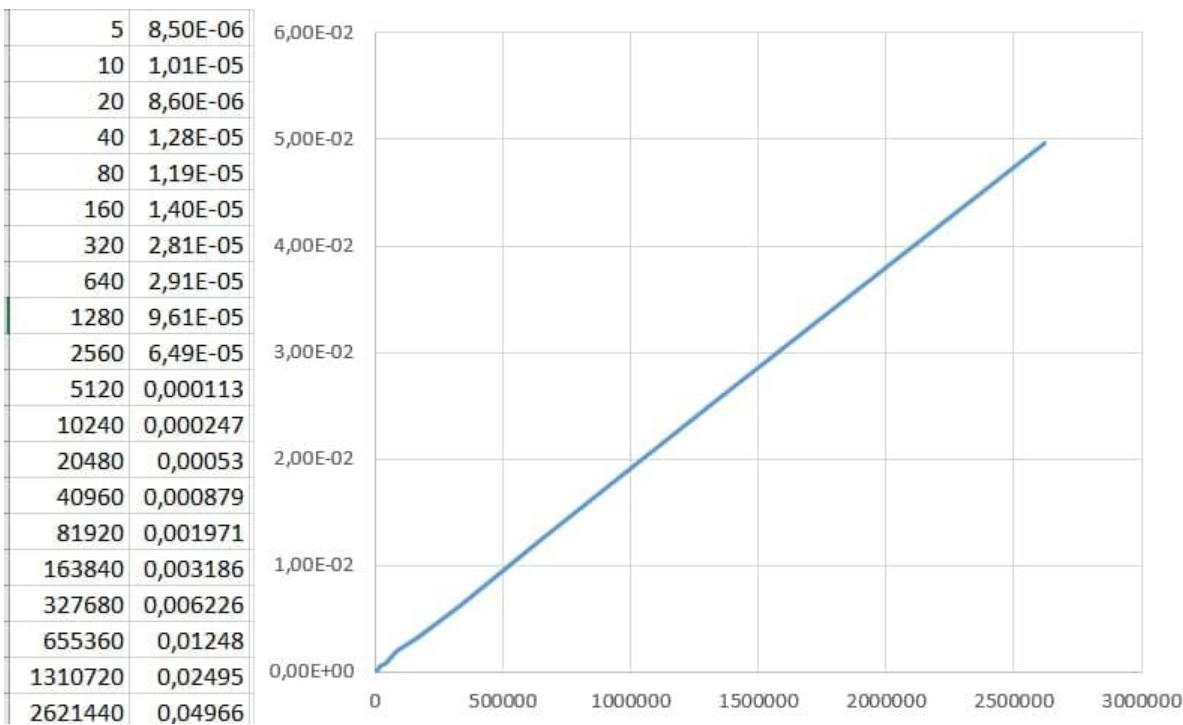
Скріншот коду:

```
1 package main
2
3 import (
4     "fmt"
5     "strings"
6     "testing"
7
8     . "github.com/asdf2107/architecture-lab-4/src/utils"
9 )
10
11 func BenchmarkParse(b *testing.B) {
12     commands := []string{
13         "print",
14         "palindrom",
15     }
16     const baseLen = 5
17
18     for i := 0; i < 20; i++ {
19         l := baseLen * 1 << i
20         in := make([]string, l)
21         in[0] = commands[i%2]
22         for k := 1; k < l; k++ {
23             in[k] = "a"
24         }
25         b.Run(fmt.Sprintf("len=%d", l), func(b *testing.B) {
26             Parse(strings.Join(in, ""))
27         })
28     }
29 }
```

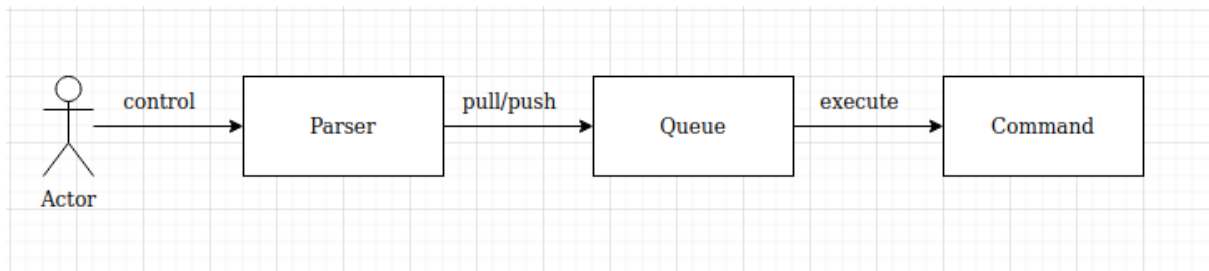
Скріншот виконання:

```
architecture-lab-4 : zsh - Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Copy Paste Find
~/Gi/K/architecture-lab-4 maste go test -bench .
goos: linux
goarch: amd64
pkg: github.com/asdf2107/architecture-lab-4
cpu: AMD Ryzen 3 2200U with Radeon Vega Mobile Gfx
BenchmarkParse/len=5-4      1000000000      0.0000085 ns/op
BenchmarkParse/len=10-4     1000000000      0.0000101 ns/op
BenchmarkParse/len=20-4     1000000000      0.0000086 ns/op
BenchmarkParse/len=40-4     1000000000      0.0000128 ns/op
BenchmarkParse/len=80-4     1000000000      0.0000119 ns/op
BenchmarkParse/len=160-4    1000000000      0.0000140 ns/op
BenchmarkParse/len=320-4    1000000000      0.0000281 ns/op
BenchmarkParse/len=640-4    1000000000      0.0000291 ns/op
BenchmarkParse/len=1280-4   1000000000      0.0000961 ns/op
BenchmarkParse/len=2560-4   1000000000      0.0000649 ns/op
BenchmarkParse/len=5120-4   1000000000      0.0001127 ns/op
BenchmarkParse/len=10240-4  1000000000      0.0002474 ns/op
BenchmarkParse/len=20480-4  1000000000      0.0005301 ns/op
BenchmarkParse/len=40960-4  1000000000      0.0008788 ns/op
BenchmarkParse/len=81920-4  1000000000      0.001971 ns/op
BenchmarkParse/len=163840-4 1000000000      0.003186 ns/op
BenchmarkParse/len=327680-4 1000000000      0.006226 ns/op
BenchmarkParse/len=655360-4 1000000000      0.01248 ns/op
BenchmarkParse/len=1310720-4 1000000000      0.02495 ns/op
BenchmarkParse/len=2621440-4 1000000000      0.04966 ns/op
PASS
ok      github.com/asdf2107/architecture-lab-4 1.181s
~/Gi/K/architecture-lab-4 maste
```

Графік результатів:



Діаграма взаємодії:



Інтерпретатор читає рядки послідовно, парсить команду та додає до черги де виконуються команди.

У екземлярі EventLoop є методи Start (створює чергу), Post (додає команду до черги) та AwaitFinish (очікує, поки всі заплановані команди завершаться). У інтерфейсі Command є метод Execute, що надає можливість виконувати додаткові команди. EventLoop виводить результат.

Висновки:

При виконанні розрахунково-графічної роботи було закріплено навички ілюстрації програмних систем та оцінки часу виконання алгоритмів.

Було розроблено бенчмарки та діаграми взаємодії.