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.

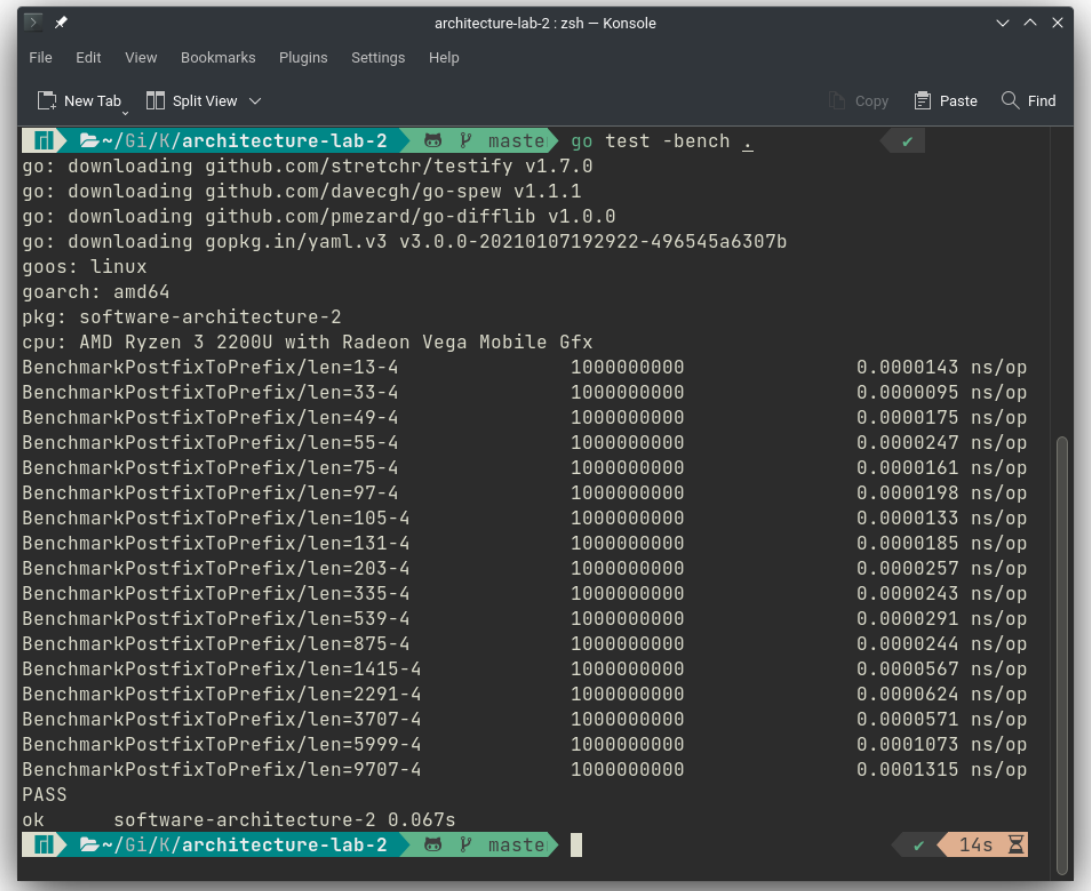
Частина перша

Функція для підтвердження лінійного часу виконання функції:

Скріншот коду:

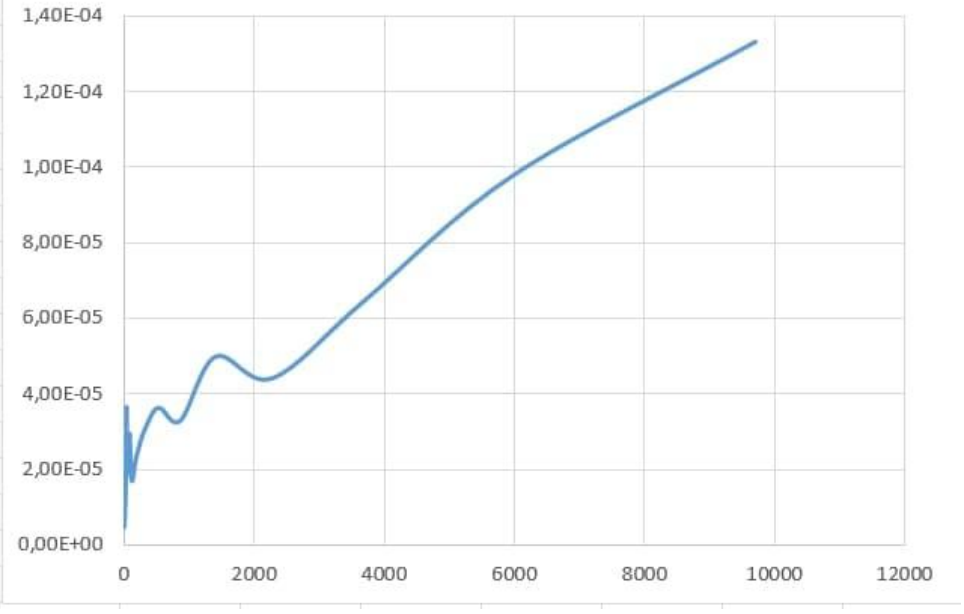
```go
package lab2

import (
	"fmt"
	"testing"
)

func BenchmarkPostfixToPrefix(b *testing.B) {
	postfixes := []string{
		"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 ^ / +",
		"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 2 - 3 * 5 +",
		"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 * 5 + 3 4 2 * 1 3 - 2 ^ / + 3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
		"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
		"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
		"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
		"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
		"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
		"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
		"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
		"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
		"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
		"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
	}

	count := len(postfixes)
	for i := 0; i < count; i++ {
		b.Run(fmt.Sprintf("len=%d", len(postfixes[i])), func(b *testing.B) {
			PostfixToPrefix(postfixes[i])
		})
	}
}
```

Скріншот виконання:



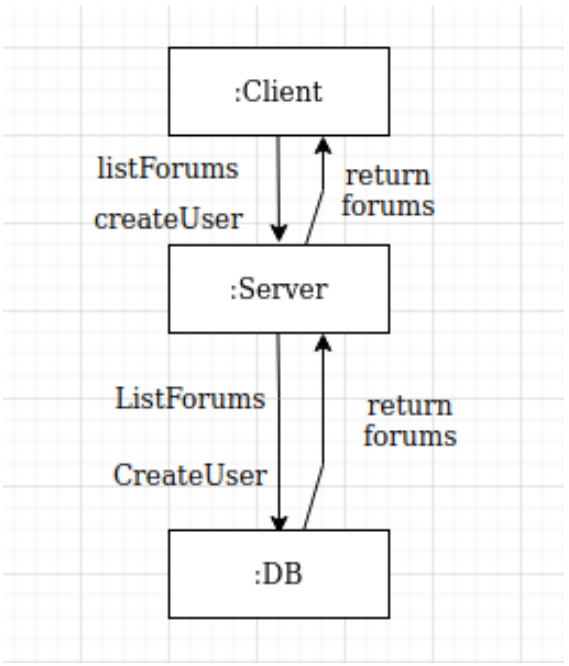
Графік результатів:

13	4,90E-06
33	1,48E-05
49	3,67E-05
55	1,95E-05
75	2,44E-05
97	2,96E-05
105	2,44E-05
131	1,70E-05
203	2,39E-05
335	3,10E-05
539	3,65E-05
875	3,31E-05
1415	5,00E-05
2291	4,43E-05
3707	6,49E-05
5999	9,81E-05
9707	0,000133



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.

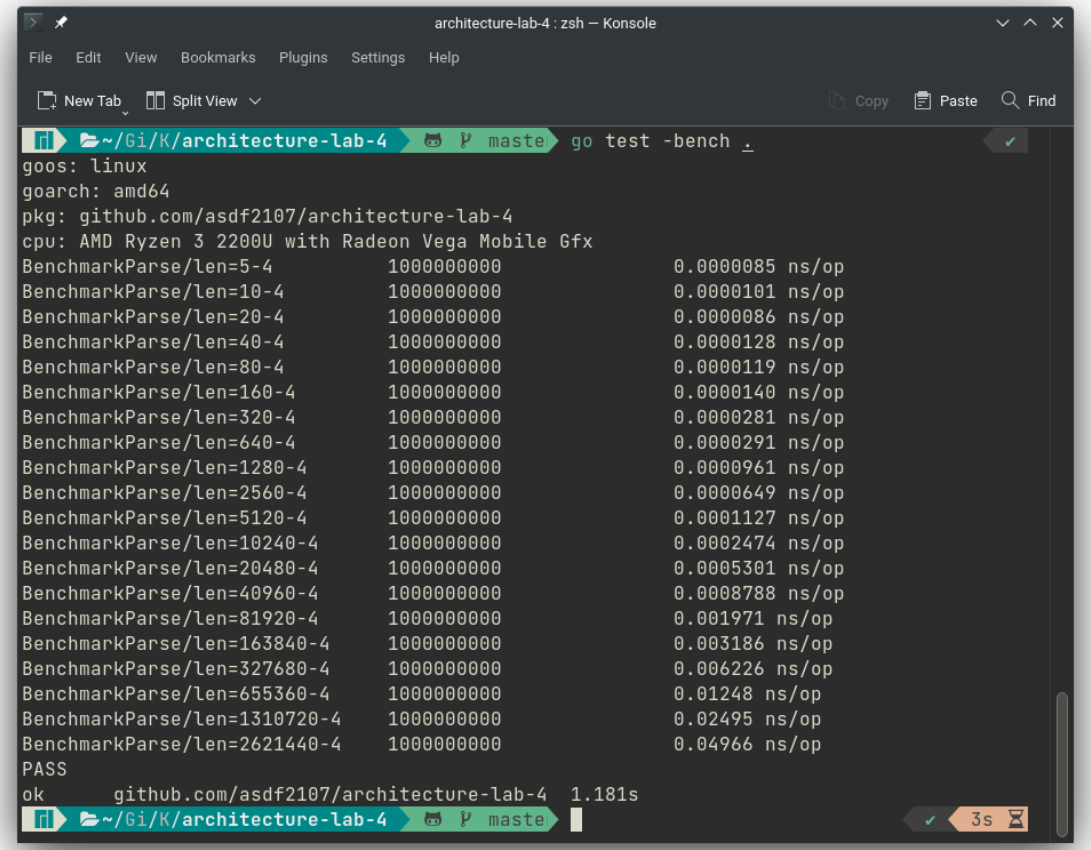
Частина третя

Функція для підтвердження лінійного часу роботи парсера команд:

Скріншот коду:

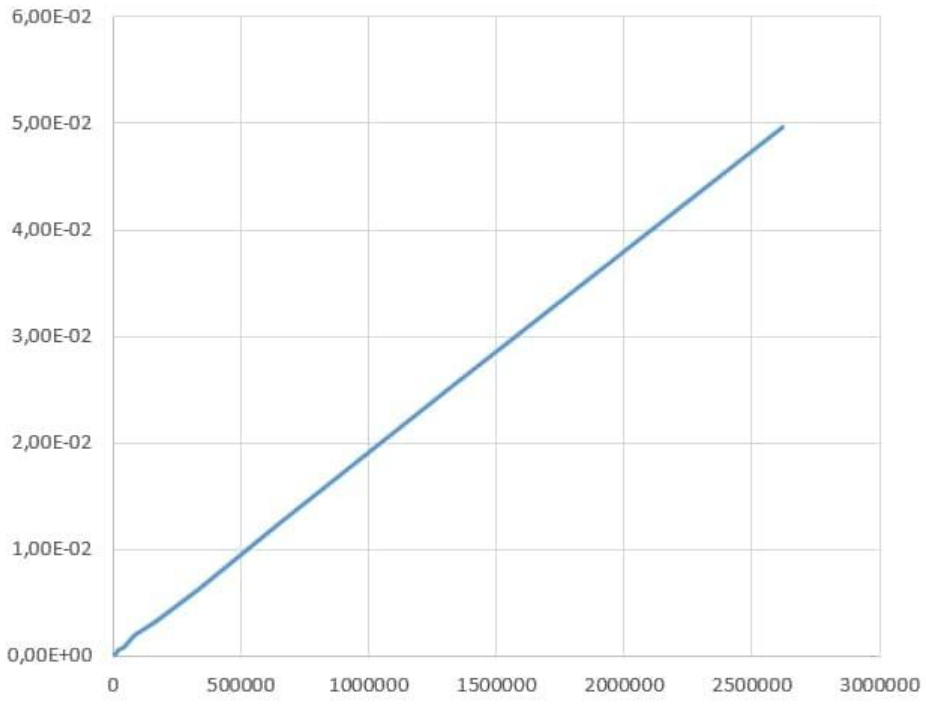
```go
package main

import (
	"fmt"
	"strings"
	"testing"

	. "github.com/asdf2107/architecture-lab-4/src/utils"
)

func BenchmarkParse(b *testing.B) {
	commands := []string{
		"print",
		"palindrom",
	}
	const baseLen = 5

	for i := 0; i < 20; i++ {
		l := baseLen * 1 << i
		in := make([]string, l)
		in[0] = commands[i%2]
		for k := 1; k < l; k++ {
			in[k] = "a"
		}
		b.Run(fmt.Sprintf("len=%d", l), func(b *testing.B) {
			Parse(strings.Join(in, ""))
		})
	}
}
```

Скріншот виконання:

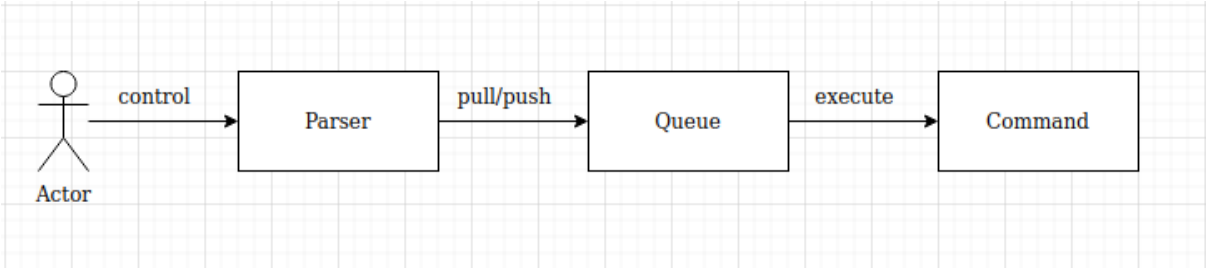


Графік результатів:

5	8,50E-06
10	1,01E-05
20	8,60E-06
40	1,28E-05
80	1,19E-05
160	1,40E-05
320	2,81E-05
640	2,91E-05
1280	9,61E-05
2560	6,49E-05
5120	0,000113
10240	0,000247
20480	0,00053
40960	0,000879
81920	0,001971
163840	0,003186
327680	0,006226
655360	0,01248
1310720	0,02495
2621440	0,04966



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.