

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА  
Факультет прикладної математики та інформатики

**Комп'ютерні інформаційні мережі  
ЛАБОРАТОРНА РОБОТА №10**

**Тема: «Програмування сокетів».**

Виконала:  
*Ст. Пелещак Вероніка*  
ПМІ-35с

**Тема:** Програмування сокетів.

**Мета:** Отримання знань і практичних навичок, необхідних для програмування простих мережевих клієнт-серверних додатків з використанням сокетів.

**Завдання:** Використовуючи бібліотеку для роботи з сокетами, написати клієнт-серверний застосунок, де клієнт та сервер обмінюються певною інформацією. Реалізувати вибір режиму роботи сервера під час його запуску — послідовна/паралельна обробка запитів.

### **Короткий опис серверної частини (server.py):**

У серверній частині створено сокет, який прив'язується до IP-адреси та порту та переходить у режим прослуховування вхідних підключень. Під час запуску користувач обирає режим роботи сервера — *послідовний* або *паралельний*.

У послідовному режимі всі клієнти обробляються один за одним у головному потоці.

У паралельному режимі для кожного нового клієнта створюється окремий потік, що дозволяє серверу одночасно працювати з кількома підключеннями.

Після прийняття з'єднання сервер отримує повідомлення від клієнта, розбирає введену команду та виконує відповідні дії.

До програми додано функціонал міні-калькулятора: сервер підтримує операції додавання (**ADD**), віднімання (**SUB**), множення (**MUL**), ділення (**DIV**), ступінь (**POW**) і квадратний корінь (**SQRT**). Для кожної операції виконується обробка аргументів та повертається результат обчислення.

У разі введення будь-якого іншого тексту сервер повертає клієнту повідомлення у форматі “Сервер отримав: ...”.

Команда **quit** завершує роботу з конкретним клієнтом.

```

import socket
import threading
import sys

HOST = "0.0.0.0"
PORT = 5050

def handle_client(conn, addr): 2 usages
    print(f"[INFO] Підключено клієнта: {addr}")

    with conn:
        while True:
            data = conn.recv(1024)

            if not data:
                print(f"[INFO] Клієнт {addr} відключився")
                break

            msg = data.decode("utf-8").strip()
            print(f"[RECV] {addr}: {msg}")

            parts = msg.split()

            if len(parts) == 3:
                cmd = parts[0].lower()
                try:
                    a = float(parts[1])
                    b = float(parts[2])
                except ValueError:
                    conn.sendall("Помилка: аргументи повинні бути числами.\n".encode("utf-8"))
                    continue

                if cmd == "add":
                    conn.sendall(f"Результат: {a + b}\n".encode("utf-8"))
                    continue

```

```

                if cmd == "sub":
                    conn.sendall(f"Результат: {a - b}\n".encode("utf-8"))
                    continue

                if cmd == "mul":
                    conn.sendall(f"Результат: {a * b}\n".encode("utf-8"))
                    continue

                if cmd == "div":
                    if b == 0:
                        conn.sendall("Помилка: ділення на нуль.\n".encode("utf-8"))
                    else:
                        conn.sendall(f"Результат: {a / b}\n".encode("utf-8"))
                    continue

                if cmd == "pow":
                    conn.sendall(f"Результат: {a ** b}\n".encode("utf-8"))
                    continue

            if len(parts) == 2:
                cmd = parts[0].lower()
                try:
                    x = float(parts[1])
                except ValueError:
                    conn.sendall("Помилка: аргумент повинен бути числом.\n".encode("utf-8"))
                    continue

                if cmd == "sqrt":
                    import math
                    if x < 0:
                        conn.sendall("Помилка: не можна брати корінь з від'ємного числа.\n".encode("utf-8"))
                    else:
                        conn.sendall(f"Результат: {math.sqrt(x)}\n".encode("utf-8"))
                    continue

```

```

        if msg.lower() == "quit":
            conn.sendall("З'єднання завершено.\n".encode("utf-8"))
            break

        response = f"Сервер отримав: {msg}\n"
        conn.sendall(response.encode("utf-8"))

    print(f"[INFO] Сеанс завершено: {addr}")

def run_sequential(listener): 1 usage
    print("[MODE] Послідовний режим")

    while True:
        conn, addr = listener.accept()
        handle_client(conn, addr)

def run_parallel(listener): 1 usage
    print("[MODE] Паралельний режим")

    while True:
        conn, addr = listener.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr), daemon=True)
        thread.start()
        print(f"[THREAD] Створено потік для клієнта {addr}")

```

```

> if __name__ == "__main__":
    if len(sys.argv) != 2:
        sys.exit(1)

    mode = sys.argv[1].lower()

    listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    listener.bind((HOST, PORT))
    listener.listen(5)

    print(f"[INFO] Сервер запущено на {HOST}:{PORT}")

    if mode == "sequential":
        run_sequential(listener)
    elif mode == "parallel":
        run_parallel(listener)
    else:
        print("Використовуйте: sequential / parallel")

```

## Короткий опис клієнтської частини (client.py):

Клієнт створює сокет та встановлює з'єднання з сервером за вказаними IP-адресою та портом. Після підключення користувач може вводити команди або текстові повідомлення.

Клієнт надсилає введені дані на сервер, очікує відповідь за допомогою `recv()` та виводить її на екран.

Підтримуються звичайні повідомлення і команди калькулятора (ADD, SUB, MUL, DIV, POW, SQRT).

Робота клієнта триває, доки користувач не введе команду `quit`, після чого з'єднання коректно закривається.

```
import socket
import sys

if len(sys.argv) != 3:
    sys.exit(1)

HOST = sys.argv[1]
PORT = int(sys.argv[2])

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    print(f"[INFO] Підключення до {HOST}:{PORT} ...")
    s.connect((HOST, PORT))
    print("[INFO] Підключено.")

    while True:
        msg = input("Введіть текст (або 'quit' щоб вийти): ")

        s.sendall((msg + "\n").encode("utf-8"))

        if msg.lower() == "quit":
            data = s.recv(1024).decode("utf-8")
            print(data)
            break

        data = s.recv(1024).decode("utf-8")
        print("Відповідь сервера:", data.strip())
```

```
lab10 — Python server.py parallel — 70x28
[veronikapelesak@Noutbuk-Veronika lab10 % python3 server.py parallel
[INFO] Сервер запущено на 0.0.0.0:5050
[MODE] Паралельний режим
[INFO] Підключено клієнта: ('127.0.0.1', 54052)
[THREAD] Створено потік для клієнта ('127.0.0.1', 54052)
[RECV] ('127.0.0.1', 54052): add 66 33
[RECV] ('127.0.0.1', 54052): sub 099 123
[RECV] ('127.0.0.1', 54052): mul 12 4
[RECV] ('127.0.0.1', 54052): div 4 0
[RECV] ('127.0.0.1', 54052): div 123 5
[RECV] ('127.0.0.1', 54052): div 4 3
[RECV] ('127.0.0.1', 54052): d
[RECV] ('127.0.0.1', 54052): pow 4 2
[RECV] ('127.0.0.1', 54052): pow 8 12
[RECV] ('127.0.0.1', 54052): sqrt 125
[RECV] ('127.0.0.1', 54052): sqrt 225
[RECV] ('127.0.0.1', 54052): quit
[INFO] Сеанс завершено: ('127.0.0.1', 54052)

lab10 — -zsh — 80x29
[veronikapelesak@Noutbuk-Veronika lab10 % python3 client.py 127.0.0.1 5050
[INFO] Підключення до 127.0.0.1:5050 ...
[INFO] Підключено.
Введіть текст (або 'quit' щоб вийти): add 66 33
Відповідь сервера: Результат: 99.0
Введіть текст (або 'quit' щоб вийти): sub 099 123
Відповідь сервера: Результат: -24.0
Введіть текст (або 'quit' щоб вийти): mul 12 4
Відповідь сервера: Результат: 48.0
Введіть текст (або 'quit' щоб вийти): div 4 0
Відповідь сервера: Помилка: ділення на нуль.
Введіть текст (або 'quit' щоб вийти): div 123 5
Відповідь сервера: Результат: 24.6
Введіть текст (або 'quit' щоб вийти): div 4 3
Відповідь сервера: Результат: 1.3333333333333333
Введіть текст (або 'quit' щоб вийти): d
Відповідь сервера: Сервер отримав: d
Введіть текст (або 'quit' щоб вийти): pow 4 2
Відповідь сервера: Результат: 16.0
Введіть текст (або 'quit' щоб вийти): pow 8 12
Відповідь сервера: Результат: 68719476736.0
Введіть текст (або 'quit' щоб вийти): sqrt 125
Відповідь сервера: Результат: 11.180339887498949
Введіть текст (або 'quit' щоб вийти): sqrt 225
Відповідь сервера: Результат: 15.0
Введіть текст (або 'quit' щоб вийти): quit
З'єднання завершено.
```

**Висновок:** У ході лабораторної роботи було розроблено клієнт-серверний застосунок з використанням сокетів. Реалізовано два режими роботи сервера — послідовний та паралельний, що дозволяє дослідити різницю між однопоточною та багатопоточною обробкою підключень.

Клієнт і сервер успішно встановлюють з'єднання, обмінюються повідомленнями та виконують додаткову функціональність — обчислення математичних операцій у форматі простого мережевого калькулятора.

У результаті виконання роботи отримано практичні навички створення сокетів, передачі даних між процесами, організації багатопоточної обробки підключень та розробки простих мережових протоколів взаємодії. Застосунок коректно працює та відповідає поставленим у лабораторній роботі вимогам.