Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра системного аналізу та теорії прийняття рішень

Звіт

з лабораторної роботи № 2

на тему:

**«Взаємодія розподілених процесів через механізм сокетів»**

Варіант 27

Студента другого курсу

групи К-23

Старостюк Дарії Олегівни

Факультету комп’ютерних наук

та кібернетики

Київ – 2023

## Мета

В даній лабораторній роботі необхідно освоїти механізм (технологію) сокетів стеку протоколів TCP/IP, зокрема його реалізацію в MS Windows. Індивідуальний варіант роботи полягає в розробці двох програм (клієнта та сервера, які запускаються на різних станціях мережі), розробці протоколу обміну даними між ними та демонстрації роботи програм.

## Зміст індивідуального завдання

**(Відновлення об'єкта)** На сервері у прямокутній області відомого для клієнта розміру зберігається певний об'єкт (може бути просто множина точок). Клієнт має N спроб для розпізнавання об'єкта (фігури), це буде правильно вгадана множина точок об'єкта в аналогічній прямокутній області. Вибір точки в області проводити за методом Монте-Карло (вибравши початок координат, наприклад, у нижній лівій вершині, і випадково вибираємо для поточної точки значення ширини та висоти). Точка буде вгадана, якщо вона попадає в заданий окіл правильної точки об'єкта. На клієнті потрібно відобразити отриманий об'єкт. На сервері має бути відображений оригінальний об'єкт. Можна область з зображенням будувати у матриці символів у консольному вікні без графічних об'єктів

## Огляд протоколу

* **Протокол обміну**: Текстовий протокол.
* **Заголовок повідомлення (1 байт)**: Визначає тип команди.
* **Тіло повідомлення (до 255 байт)**: Містить додаткові дані, включаючи координати і стан ігрового поля.

## Типи команд:

* socket: Використовується для створення сокету (server\_socket та client\_socket) для слухання та з'єднання клієнтів.
* bind: Призначено для прив'язки сокету до певного порту та адреси.
* listen: Використовується для переходу сокету до режиму прослуховування, при якому він очікує на вхідні підключення.
* accept: Ця команда призначена для приймання вхідного підключення від клієнта та створення нового сокету для обміну даними з цим клієнтом.
* connect: Використовується клієнтом для з'єднання з сервером за допомогою IP-адреси та порту сервера.
* send: Використовується для надсилання даних через сокет.
* recv: Використовується для отримання даних з сокету.

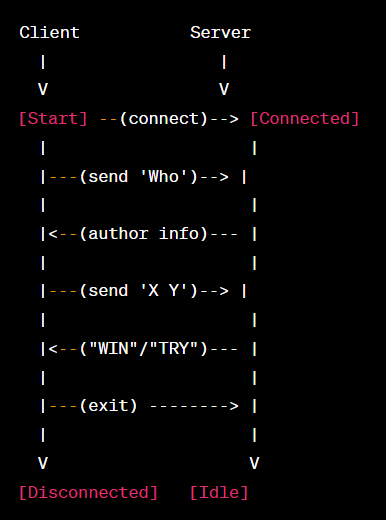
## Програми

1. **Сервер (server.cpp)**: Програма сервера створює об'єкт з випадковими координатами, який представляє об'єкт у прямокутній області. Сервер приймає команди від клієнта та надсилає відповіді, засновані на порівнянні координат клієнта і об'єкта на сервері.
2. **Клієнт (client.cpp)**: Програма клієнта взаємодіє з сервером, намагаючись вгадати об'єкт, розташований на сервері, шляхом випадкового вибору точок в прямокутній області.

## Інструкція користувача для клієнта:

1. Підключіться до сервера за допомогою команди "CONNECT\_CLIENT".
2. Клієнт має обмежену кількість спроб (N) для вгадування об'єкта.
3. Клієнт вибирає випадкову точку в прямокутній області і надсилає її серверу.
4. Сервер перевіряє, чи точка потрапляє в окіл правильної точки об'єкта.
5. Якщо клієнт вгадав об'єкт, сервер відправляє "WIN" і гра завершується.
6. Якщо клієнт не вгадав об'єкт, сервер відправляє "TRY AGAIN" і спроба клієнта зменшується.
7. Гра завершується, коли клієнт використав всі свої спроби.

## **Діаграма Станів**



## **Код server.cpp**

#include <iostream>

#include <winsock2.h>

#include <windows.h>

#include <ctime>

#include <string>

#include <fstream>

#pragma comment(lib, "ws2\_32.lib")

const int SERVER\_PORT = 1052;

const int RECTANGLE\_AREA\_WIDTH = 50;

const int OBJECT\_RADIUS = 5;

int maxAttempts = 10;

struct Point {

    int x;

    int y;

};

bool isPointInsideObject(const Point& point, const Point& objectCenter, int objectRadius) {

    int dx = point.x - objectCenter.x;

    int dy = point.y - objectCenter.y;

    return (dx \* dx + dy \* dy) <= (objectRadius \* objectRadius);

}

Point generateRandomPointInRectangle(int width, int height) {

    Point point;

    point.x = rand() % width;

    point.y = rand() % height;

    return point;

}

void writeToServerLog(const std::string& message) {

    std::ofstream serverLog("server\_log.txt", std::ios::app);

    if (serverLog.is\_open()) {

        time\_t rawTime;

        struct tm timeInfo;

        char timeBuffer[80];

        time(&rawTime);

        localtime\_s(&timeInfo, &rawTime);

        strftime(timeBuffer, sizeof(timeBuffer), "%Y-%m-%d %H:%M:%S", &timeInfo);

        serverLog << timeBuffer << ": " << message << std::endl;

        serverLog.close();

    }

}

std::string GetAuthorAndVariantInfo() {

    return "Author: [Dariia Starostiuk K-23]\nVariant: [27]\n";

}

void displayObjectAndRadius(const Point& objectCenter, int objectRadius) {

    std::cout << "Оригінальний об'єкт: X=" << objectCenter.x << " Y=" << objectCenter.y << " Радіус=" << objectRadius << std::endl;

    for (int y = 0; y < RECTANGLE\_AREA\_WIDTH; ++y) {

        for (int x = 0; x < RECTANGLE\_AREA\_WIDTH; ++x) {

            if (isPointInsideObject({ x, y }, objectCenter, objectRadius)) {

                std::cout << "O";

            } else {

                std::cout << ".";

            }

        }

        std::cout << std::endl;

    }

}

int main() {

    SetConsoleOutputCP(CP\_UTF8);

    srand(static\_cast<unsigned int>(time(nullptr)));

    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

        std::cerr << "Помилка при ініціалізації Winsock." << std::endl;

        return 1;

    }

    int server\_socket, new\_socket, valread;

    sockaddr\_in server\_addr;

    char buffer[255] = {0};

    int objectRadius = 0;

    Point objectCenter;

    if ((server\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

        perror("socket failed");

        WSACleanup();

        return 1;

    }

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(SERVER\_PORT);

    server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

    if (bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

        perror("bind");

        closesocket(server\_socket);

        WSACleanup();

        return 1;

    }

    if (listen(server\_socket, 3) == -1) {

        perror("listen");

        closesocket(server\_socket);

        WSACleanup();

        return 1;

    }

    std::cout << "Сервер слухає на порту " << SERVER\_PORT << std::endl;

    writeToServerLog("Сервер стартував.");

    while (true) {

        if ((new\_socket = accept(server\_socket, nullptr, nullptr)) == -1) {

            perror("accept");

            WSACleanup();

            return 1;

        }

        objectRadius = rand() % OBJECT\_RADIUS + 1;

        objectCenter = generateRandomPointInRectangle(RECTANGLE\_AREA\_WIDTH, RECTANGLE\_AREA\_WIDTH);

        displayObjectAndRadius(objectCenter, objectRadius);

        int remainingAttempts = maxAttempts;

        while (remainingAttempts > 0) {

            std::cout << "Залишилось спроб: " << remainingAttempts << std::endl;

            valread = recv(new\_socket, buffer, sizeof(buffer), 0);

            if (valread <= 0) {

                std::cerr << "Клієнт відключився." << std::endl;

                break;

            }

            writeToServerLog("Отримано від клієнта: " + std::string(buffer));

            if (strcmp(buffer, "Who") == 0) {

                std::string authorAndVariantInfo = GetAuthorAndVariantInfo();

                send(new\_socket, authorAndVariantInfo.c\_str(), authorAndVariantInfo.length(), 0);

            }

            else {

    int x, y;

    char response[255];

    if (sscanf(buffer, "%d %d", &x, &y) == 2) {

        if (isPointInsideObject({ x, y }, objectCenter, objectRadius)) {

            strcpy(response, "WIN");

        } else {

            strcpy(response, "TRY AGAIN");

        }

        send(new\_socket, response, strlen(response), 0);

        if (strcmp(response, "WIN") == 0) {

            std::cout << "Клієнт виграв!" << std::endl;

            writeToServerLog("Клієнт виграв.");

            break;

        }

    } else {

        strcpy(response, "Невірний формат координат.");

        send(new\_socket, response, strlen(response), 0);

    }

    writeToServerLog("Надіслано клієнту: " + std::string(response));

    remainingAttempts--;

}

        }

        closesocket(new\_socket);

    }

    closesocket(server\_socket);

    WSACleanup();

    return 0;

}

## **Код client.cpp**

#include <iostream>

#include <winsock2.h>

#include <windows.h>

#include <ctime>

#include <string>

#include <fstream>

#pragma comment(lib, "ws2\_32.lib")

const char\* SERVER\_IP = "127.0.0.1";

const int SERVER\_PORT = 1052;

const int RECTANGLE\_AREA\_WIDTH = 50;

void displayGrid(int objectRadius, int objectX, int objectY) {

    int gridSize = objectRadius \* 2 + 1;

    for (int y = 0; y < gridSize; ++y) {

        for (int x = 0; x < gridSize; ++x) {

            if (x == objectRadius && y == objectRadius) {

                std::cout << "O";

            } else {

                std::cout << ".";

            }

        }

        std::cout << std::endl;

    }

}

std::string GetAuthorAndVariantInfo() {

    return "Author: [Dariia Starostiuk K-23]\nVariant: [27]\n";

}

int main() {

    SetConsoleOutputCP(CP\_UTF8);

    srand(static\_cast<unsigned int>(time(nullptr)));

    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

        std::cerr << "Помилка при ініціалізації Winsock." << std::endl;

        return 1;

    }

    int client\_socket;

    sockaddr\_in server\_addr;

    char buffer[255] = {0};

    int objectRadius = 0;

    int objectX = 0;

    int objectY = 0;

    if ((client\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

        perror("socket failed");

        WSACleanup();

        return 1;

    }

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(SERVER\_PORT);

    server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);

    if (connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

        perror("connect");

        closesocket(client\_socket);

        WSACleanup();

        return 1;

    }

    std::ofstream clientLog("client\_log.txt");

    clientLog << "Client Log:\n";

    std::cout << "Connected to the server." << std::endl;

    std::cout << "Розмір області: " << RECTANGLE\_AREA\_WIDTH << "x" << RECTANGLE\_AREA\_WIDTH << std::endl;

    while (true) {

        std::cout << "Enter coordinates (X Y), 'exit' to quit, or 'Who' to get author info: ";

        std::string input;

        std::getline(std::cin, input);

        if (input == "exit") {

            send(client\_socket, input.c\_str(), input.length(), 0);

            break;

        }

        if (input == "Who") {

            send(client\_socket, input.c\_str(), input.length(), 0);

            int valread = recv(client\_socket, buffer, sizeof(buffer), 0);

            if (valread <= 0) {

                std::cerr << "Server disconnected." << std::endl;

                clientLog << "Server disconnected." << std::endl;

                break;

            }

            std::cout << "Server Response:\n" << buffer << std::endl;

            clientLog << "Server Response:\n" << buffer << std::endl;

        } else if (sscanf(input.c\_str(), "%d %d", &objectX, &objectY) == 2) {

            char message[255];

            sprintf(message, "%d %d", objectX, objectY);

            send(client\_socket, message, strlen(message), 0);

            int valread = recv(client\_socket, buffer, sizeof(buffer), 0);

            if (valread <= 0) {

                std::cerr << "Server disconnected." << std::endl;

                clientLog << "Server disconnected." << std::endl;

                break;

            }

            if (strcmp(buffer, "WIN") == 0) {

                objectRadius = objectX;

                displayGrid(objectRadius, objectX, objectY);

            } else {

                std::cout << "Server Response: " << buffer << std::endl;

                clientLog << "Server Response: " << buffer << std::endl;

            }

        } else {

            std::cout << "Invalid input. Please enter coordinates in the format 'X Y'." << std::endl;

        }

        memset(buffer, 0, sizeof(buffer));

    }

    clientLog.close();

    closesocket(client\_socket);

    WSACleanup();

    return 0;

}

## **Приклад тексту системного журнала server\_log.txt**

2023-10-21 20:16:05: Сервер стартував.

2023-10-21 20:16:33: Отримано від клієнта: Who

2023-10-21 20:16:49: Отримано від клієнта: 29 34

2023-10-21 20:16:49: Надіслано клієнту: TRY AGAIN

2023-10-21 20:16:54: Отримано від клієнта: 54 74

2023-10-21 20:16:54: Надіслано клієнту: TRY AGAIN

2023-10-21 20:17:04: Отримано від клієнта: 29 41

2023-10-21 20:17:04: Надіслано клієнту: TRY AGAIN

2023-10-21 20:17:12: Отримано від клієнта: 28 40

2023-10-21 20:17:12: Клієнт виграв.

2023-10-21 20:18:06: Отримано від клієнта: 32 42

2023-10-21 20:18:06: Клієнт виграв.

## **Приклад тексту системного журнала client\_log.txt**

Client Log:

Server Response: TRY AGAIN

Server Response: TRY AGAIN