**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Лабораторная работа №3

по курсу

«Операционные системы и сети»

Выполнили:

студенты группы ИКПИ-14

Хохлов Т.В.

Принял:

доцент кафедры ПИиВТ

Дагаев А.В

Санкт-Петербург

2023 г.

**Цель работы**

Целью работы является разработка программы на языке C++ с использованием библиотеки потоков <thread>.

# Описание программы

Программа представляет собой многопоточное приложение на C++, в котором два потока работают параллельно: один поток генерирует случайное число, а другой поток пытается угадать это число. Есть также третий поток, отвечающий за ввод пользователя. Для синхронизации потоков и для эксклюзивного доступа к ресурсам используются мьютексы.

# Описание кода

**Основные компоненты:**

Критические секции:

**generateSecretNumber():**

* Поток, который генерирует случайное число в диапазоне от 1 до 100.
* Число обновляется каждые 40 миллисекунд.

**guessNumber():**

* Поток, который пытается угадать сгенерированное число.
* Попытка происходит каждые 10 миллисекунд.

**userInput():**

* Поток, ожидающий ввода пользователя.

Мьютекс:

**std::mutex mtx;**

* В функции **generateSecretNumber**, перед генерацией нового секретного числа, поток "захватывает" мьютекс, чтобы обновить **secretNumber** и вывести информацию о загаданном числе.
* В функции **guessNumber**, перед угадыванием числа, поток "захватывает" мьютекс, чтобы обновить **numberGuessAttempts** и вывести информацию о задуманном числе.
* В функции **userInput**, после пользовательского ввода, поток "захватывает" мьютекс, чтобы обновить **programStop** и вывести информацию о завершении программы.

Основная функция:

**main():**

* Запускает три потока: генерации числа, попыток угадывания и ввода пользователя.
* После завершения потоков выводится статистика:
  + Общее количество сгенерированных чисел.
  + Количество угаданных чисел.
  + Общее количество попыток угадывания.
  + Процент успешных и неудачных попыток.

**Завершение программы:**

Программа завершается, когда пользователь вводит от клавиатуры **“s”** в терминал. По завершении выводится статистика и сообщение "Program finished!".

**Статистика:**

Статистика включает в себя общее количество сгенерированных чисел, количество угаданных чисел, общее количество попыток угадывания, а также процент успешных и неудачных попыток.

**Графики:**





**Выводы**

В ходе проведённой лабораторной работы были получены следующие результаты:

1. Освоена библиотека <**threads>** языка C++

2. Создана программа с использованием параллельных потоков.

Исходный код программы

#include <iostream>

#include <thread>

#include <mutex>

#include <random>

#include <chrono>

#include <string>

std::mutex mtx;

bool programStop = false;

int secretNumber = 0;

float numbersGenerated = 0;

float numbersGuessed = 0;

float numberGuessAttempts = 0;

void generateSecretNumber() {

    std::random\_device rd;

    std::mt19937 gen(rd());

    std::uniform\_int\_distribution<> distr(1, 100);

    while (!programStop) {

        std::unique\_lock<std::mutex> lock(mtx);

        secretNumber = distr(gen);

        std::cout << "Thread 1: New secret number is generated: " << secretNumber << std::endl;

        numbersGenerated++;

        lock.unlock();

        std::this\_thread::sleep\_for(std::chrono::seconds(2));

    }

}

void guessNumber() {

    std::random\_device rd;

    std::mt19937 gen(rd());

    std::uniform\_int\_distribution<> distr(1, 100);

    while (!programStop) {

        std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

        std::unique\_lock<std::mutex> lock(mtx);

        int guess = distr(gen);

        std::cout << "Thread 2: Guessing " << guess << std::endl;

        numberGuessAttempts++;

        if (guess == secretNumber) {

            std::cout << "Thread 2: Guessed the number " << secretNumber << std::endl;

            numbersGuessed++;

        }

    }

}

void userInput() {

    std::string input;

    while (true) {

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

        if (input == "s") {

            std::unique\_lock<std::mutex> lock(mtx);

            programStop = true;

            break;

        }

    }

}

int main() {

    std::thread generatorThread(generateSecretNumber);

    std::thread playerThread(guessNumber);

    std::thread inputThread(userInput);

    generatorThread.join();

    playerThread.join();

    inputThread.join();

    std::cout << "Program finished!" << std::endl;

    std::cout << "Number of times the number was generated: " << numbersGenerated << std::endl;

    std::cout << "Number of times the number was guessed: " << numbersGuessed << std::endl;

    std::cout << "Number of times the number was attempted to guess: " << numberGuessAttempts << std::endl;

    std::cout << std::endl << "Statistics: " << std::endl;

    std::cout << "Success attempts: " << numbersGuessed/numberGuessAttempts << "freq" << " or " << (numbersGuessed/numberGuessAttempts)\*100 << "%" << std::endl;

    std::cout << "Failure attempts: " << (numberGuessAttempts-numbersGuessed)/numberGuessAttempts << "freq" << " or " << ((numberGuessAttempts-numbersGuessed)/numberGuessAttempts)\*100 << "%" << std::endl;

    return 0;

}