

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Навчально-науковий комплекс «Інститут прикладного
системного аналізу»
Кафедра системного проектування

Лабораторна робота № 1
з курсу «Операційні системи»

Виконала студентка
Групи ДА-02
Лесечко О. Р

Київ 2022

Завдання:

Розробити вирішення класичної задачі синхронізації про «Виробника - споживача» за допомогою взаємно виключення (UNIX).

Опис завдання:

Задача Виробник-споживач (також відома як задача обмеженого буфера): 2 процеси — виробник і споживач — працюють із загальним ресурсом (буфером), що має максимальний розмір N. Виробник записує в буфер дані послідовно в чарунки 0,1,2, ..., поки він не заповниться, а споживач читає дані з буфера у зворотному порядку, поки він не спорожніє. Запис і зчитування не можуть відбуватися одночасно

Код програми:

```
#include <iostream>
#include <chrono>
#include <thread>
#include <mutex>
#include <random>

/* Класс BoundedBuffer, это ограниченный буфер */
class BoundedBuffer {
public:
    /* положить элемент в буфер */
    void push() {
        if (size_ < capacity_) {
            size_++;
            std::cout << "> Push\t\tSize = " << size_ << "\n\n";
        }
        else {
            std::cout << "> Buffer full!!!\n\n";
        }
    }
    /* удалить элемент из буфера */
    void remove() {
        if (size_ > 0) {
            size_--;
            std::cout << "> Remove\t\tSize = " << size_ << "\n\n";
        }
        else {
            std::cout << "> Buffer is empty!!!\n\n";
        }
    }
private:
    /* текущий размер буфера */
    int size_ = 0;
    /* ограниченный размер буфера */
    int capacity_ = 10;
};

/* Класс Runnable, служит общим представлением производителя и потребителя */
```

```

class Runnable {
public:
    /* это циклический процесс, и каждый раз,
       проходя свой цикл, он производит определенную порцию информации,
       которую должен обработать потребитель */
    void producer() {
        while (true) {
            int duration = generationRandomSleep(4000, 8000);
            std::this_thread::sleep_for(std::chrono::milliseconds(duration));
            {
                std::lock_guard<std::mutex> g(buffer_manipulation_);
                std::cout << "id thread = " << std::this_thread::get_id() << "\n";
                buffer_.push();
            }
        }
    }

    /* является циклическим процессом, и каждый раз,
       когда он проходит свой цикл, он может обрабатывать очередную порцию информации,
       как это было произведено производителем */
    void consumer() {
        while (true) {
            int duration = generationRandomSleep(3000, 6000);
            std::this_thread::sleep_for(std::chrono::milliseconds(duration));
            {
                std::lock_guard<std::mutex> g(buffer_manipulation_);
                std::cout << "id thread = " << std::this_thread::get_id() << "\n";
                buffer_.remove();
            }
        }
    }

private:
    /* ограниченный буфер */
    BoundedBuffer buffer_;
    std::mutex buffer_manipulation_;

    /* генерация случайного числа задержки

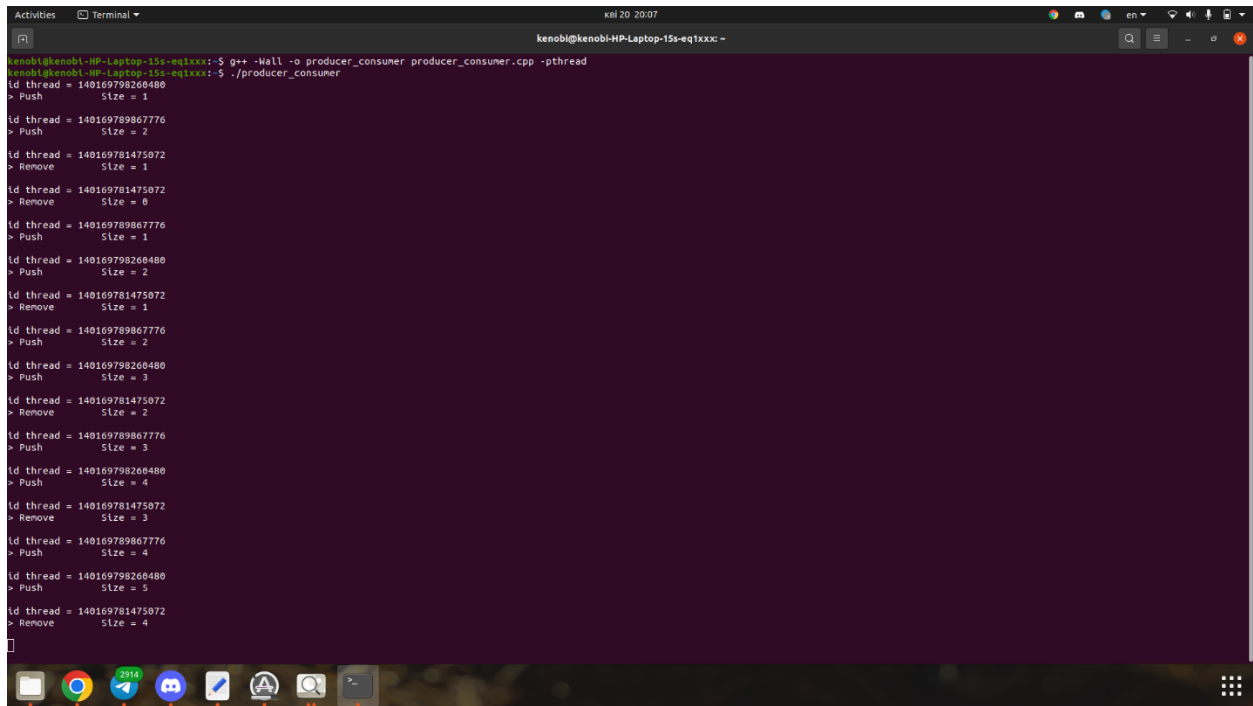
        @param min минимальное число с какого рандомится число
        @param max максимальное число до какого рандомится число

        @returns случайное число задержки */
    int generationRandomSleep(int min, int max) {
        static std::mt19937 rnd(std::time(nullptr));
        return std::uniform_int_distribution<>(min, max)(rnd);
    }
};

int main() {
    Runnable runnable;
    std::thread t1(&Runnable::producer, &runnable);
    std::thread t2(&Runnable::producer, &runnable);
    std::thread t3(&Runnable::consumer, &runnable);
    t1.join();
    t2.join();
    t3.join();
    return 0;
}

```

Результат роботи:



```
kenobi@kenobi-HP-Laptop-15s-eq1xxx:~$ g++ -Wall -o producer_consumer producer_consumer.cpp -pthread
kenobi@kenobi-HP-Laptop-15s-eq1xxx:~$ ./producer_consumer
id thread = 140169798260480
> Push      Size = 1

id thread = 140169789867776
> Push      Size = 2

id thread = 140169781475072
> Remove    Size = 1

id thread = 140169781475072
> Remove    Size = 0

id thread = 140169789867776
> Push      Size = 1

id thread = 140169798260480
> Push      Size = 2

id thread = 140169781475072
> Remove    Size = 1

id thread = 140169789867776
> Push      Size = 2

id thread = 140169798260480
> Push      Size = 3

id thread = 140169781475072
> Remove    Size = 2

id thread = 140169789867776
> Push      Size = 3

id thread = 140169798260480
> Push      Size = 4

id thread = 140169781475072
> Remove    Size = 3

id thread = 140169789867776
> Push      Size = 4

id thread = 140169798260480
> Push      Size = 5

id thread = 140169781475072
> Remove    Size = 4
```

Рис. 1 Результат роботи

Висновки:

У ході виконання даної лабораторної роботи я реалізовувала вирішення проблемної задачі про продавців та покупців, що ставила перед нами завдання про обмежений буфер, до якого намагаються отримати доступ потоки.