ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”»

Факультет компьютерных наук

Программная инженерия

Исполнитель: Хан Се Вон

МИКРОПРОЕКТ №2

Работа студента 2 курса бакалавариата группы БПИ-195

по предмету «Архитектура вычислительных систем»

Преподаватель:

Доктор технических наук,

Профессор

Легалов А. И

Москва 2020

Задание:

Задача о курильщиках. Есть три процесса-курильщика и один процесс-посредник. Курильщик непрерывно скручивает сигареты и курит их. Чтобы скрутить сигарету, нужны табак, бумага и спички. У одного процесса курильщика есть табак, у второго – бумага, а у третьего – спички. Посредник кладет на стол по два разных случайных компонента. Тот процесс курильщик, у которого есть третий компонент, забирает компоненты со стола, скручивает сигарету и курит. Посредник дожидается, пока курильщик закончит, затем процесс повторяется. Создать многопоточное приложение, моделирующее поведение курильщиков и посредника. При решении задачи использовать семафоры.

Решение.

```
// Seva_Mp2.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается
выполнение программы.
//
#include <fcntl.h>
#include <sys/types.h>
#include <iostream>
#include <thread>
#include <stdlib.h>
#include <stdio.h>
#include <mutex>
#include<condition_variable>
#include <string>
#include <vector>
using namespace std;
mutex give, ready, printer; //Переменные для просмотра захода в код
bool is_notified = false, is_ready;
int counter = 0; //Счетчик
int choice; //Переменная для выбора
condition_variable on_give, on_ready; //Условные переменные
/// <summary>
/// Вывод в консоль
/// </summary>
/// <param name="str">Строка которую необходимо вывести</param>
void print(string str) {
	unique_lock<mutex> lock(printer);
	cout << str << endl;
}
/// <summary>
/// Метод действия курильщика
/// </summary>
/// <param name="id"></param>
void Smoker(int id) {
	{
		//Смотрим проходим ли в данный участок кода
		unique_lock<mutex> lock(ready);
		//unique_lock<mutex> locker(give);

		//Подготавливаем курильщика
		print("Smoker #" + to_string(id) + " is ready...");
```

```
                this_thread::sleep_for(chrono::milliseconds(500));
                counter++;

                //Проверяем количество
                if (counter == 3)
                {
                        is_ready = true;
                        on_ready.notify_one();
                }
        }
        {
                //Смотрим проходим ли в данный участок кода
                unique_lock<mutex> lock(give);
                while (!is_notified)
                        on_give.wait(lock);

                //Если равно то выводим результат
                if (id == choice)
                {
                        print("Smoker #" + to_string(id) + " smokes. Others are waiting...");
                }
                counter--;
        }

}
/// <summary>
/// Метод действия посредника
/// </summary>
void Mediator() {
        srand(time(nullptr)); //Обнуление для рандома
        while (true) {
                unique_lock<mutex> lock(ready);
                //unique_lock<mutex> locker(give);
                while (!is_ready)
                        on_ready.wait(lock);

                //Теперь посредник выбирает объекты за 2 секунды
                print("Mediator is choosing two things...");
                this_thread::sleep_for(chrono::milliseconds(2000));

                choice = (rand() % 3) + 1;
                is_ready = false;

                //Теперь уведомим курильщика с нужным объектом
                print("Smoker with number " + to_string(choice) + " has other
components!");
                is_notified = true;
                on_give.notify_all();
        }
}
/// <summary>
/// Метод окончания цикла
/// </summary>
void EndingOfCycle() {
        cout << endl;
        choice = 0;
        is_notified = false;
        is_ready = false;
}
/// <summary>
/// Метод с которого начинается компиляция программы
/// </summary>
/// <returns></returns>
int main()
{
```

```
	vector<thread> vec_allsmokers;
	int cycles, i = 0;
	cout << "Input number of cycles:";
	cin >> cycles;
	if (cycles <= 0)
	{
		cout << "Ending of work!" << endl;
		return -1;
	}
	cout << endl;
	thread mediator(Mediator);

	for (size_t i = 0; i < cycles; i++)
	{
		vec_allsmokers.clear();
		for (size_t i = 0; i < 3; i++)
		{
			vec_allsmokers.push_back(thread(Smoker, i + 1));
		}
		for (size_t i = 0; i < 3; i++)
		{
			vec_allsmokers[i].join();
		}
		EndingOfCycle();
	}

	mediator.detach();
}
```

Результат выполнения программы приведен на рисунке 1.

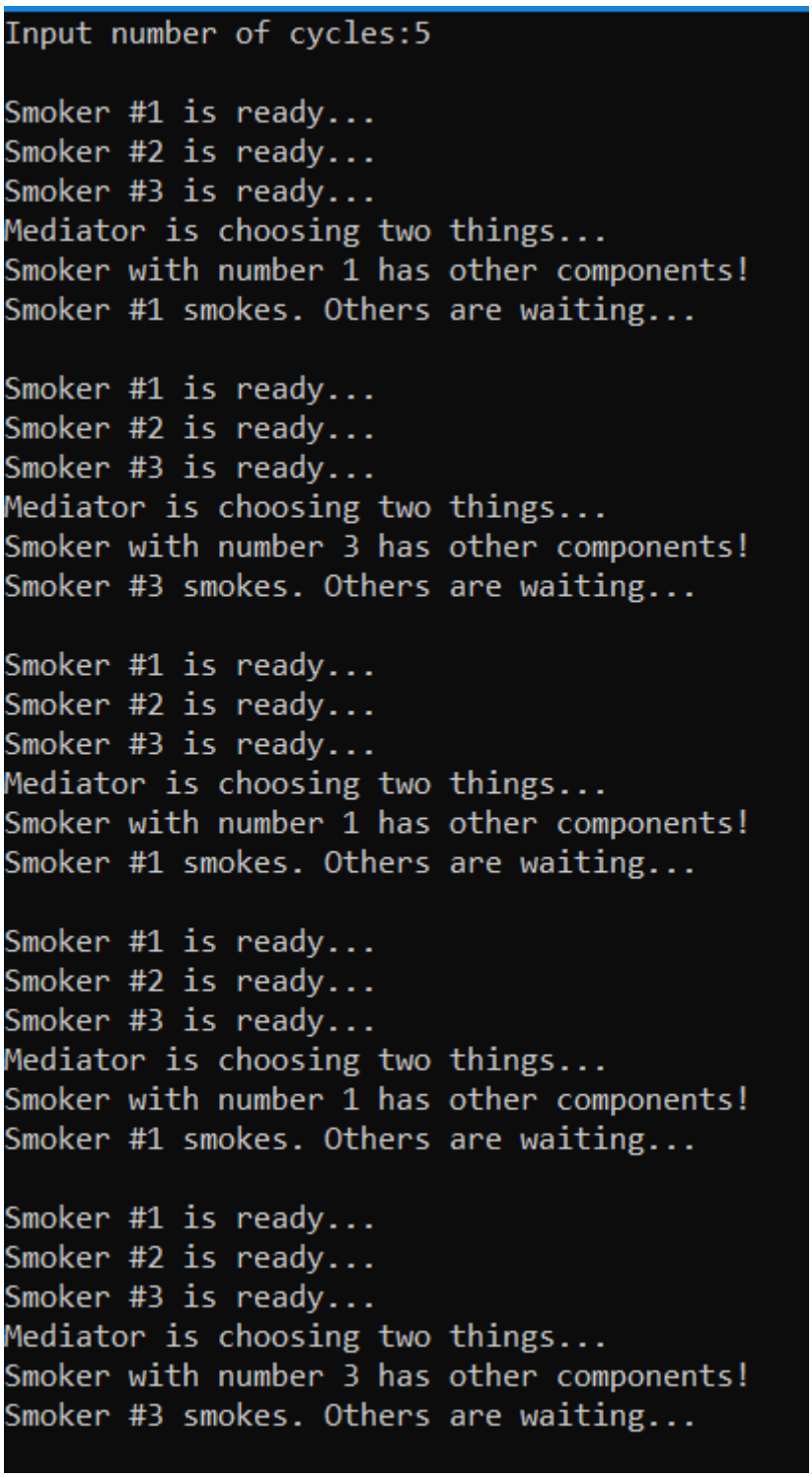

Рисунок 1 – Результат выполнения программы

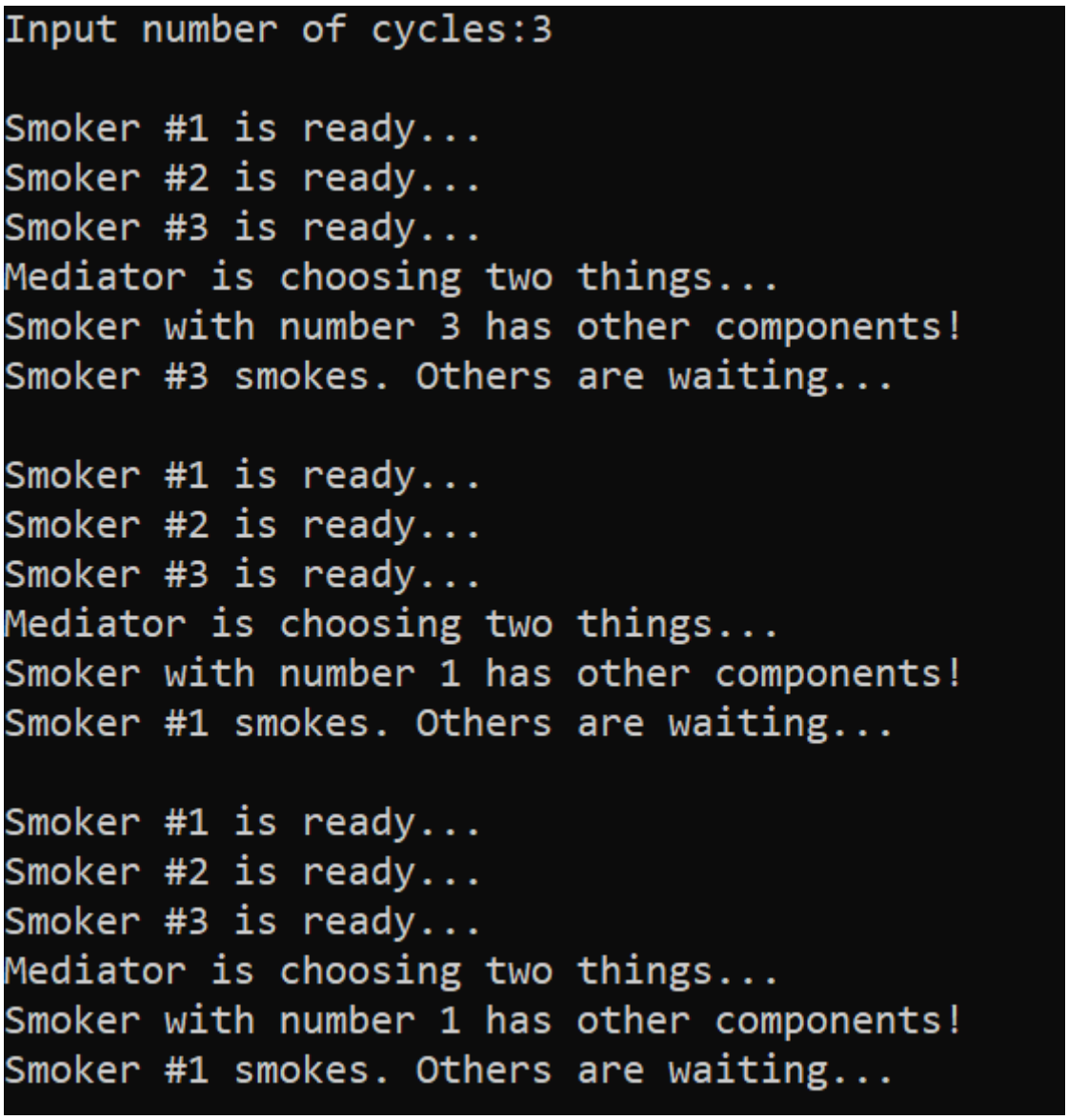

Рисунок 2 – Тест №2

Список использованных источников

- https://habr.com/ru/post/182610/
- https://www.geeksforgeeks.org/multithreading-in-cpp/
- http://www.softcraft.ru/edu/comparch/tasks/mp02/mp02.pdf