Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №7

по курсу «Технология программирования»

# «Разработка многопоточного приложения»

Вариант 6

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Долженкова М.Л/

Киров 2022

**1. Задание**

Разработать алгоритм решения задания про курильщиков, с учетом разделения вычислений между несколькими потоками. Ввести семафоры для условной синхронизации. Составить схему взаимодействие потоков. Реализовать алгоритм и протестировать его.

**2. Разработка алгоритма**

1. Основной поток забирает себе все семафоры, перейти к п.2

2. Запускаются все потоки (функции курильщиков), перейти к п.3

3. Происходит выбор курильщика при помощи случайной генерации чисел, перейти к п.4

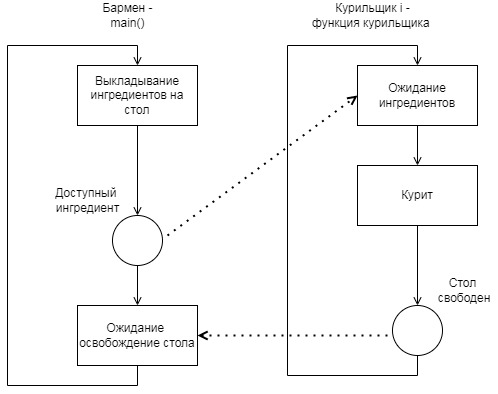
4. Разблокировка семафора для функции соответствующего курильщика, перейти к п.5

5. Выполнение потока и ожидание его завершения, перейти к п.6

6. Блокирование используемого семафора, перейти к п.7

7. Перезапуск потока, перейти к п.3

**3. Схема взаимодействия потоков**



**4. Листинг программы**

#include <iostream>

#include <Windows.h>

#include <thread>

//#include <semaphore.h>

#include <chrono>

HANDLE h\_Sem1;

HANDLE h\_Sem2;

HANDLE h\_Sem3;

int l;

using namespace std;

void first()

{

if (l == 1) {

Sleep(1000);

WaitForSingleObject(h\_Sem1, INFINITY);

cout << "1 -ый курит " << endl;

this\_thread::sleep\_for(chrono::milliseconds(100));

ReleaseSemaphore(h\_Sem1, 1, NULL);

}

}

void second()

{

if (l == 2) {

Sleep(1000);

WaitForSingleObject(h\_Sem2, INFINITY);

cout << "2 -ой курит" << endl;

this\_thread::sleep\_for(chrono::milliseconds(100));

ReleaseSemaphore(h\_Sem2, 1, NULL);

}

}

void three()

{

if (l == 3) {

Sleep(1000);

WaitForSingleObject(h\_Sem3, INFINITY);

cout << "3 -ий курит" << endl;

this\_thread::sleep\_for(chrono::milliseconds(100));

ReleaseSemaphore(h\_Sem3, 1, NULL);

}

}

int main()

{

setlocale(LC\_ALL, "ru");

cin.get();

h\_Sem1 = CreateSemaphore(NULL, 0, 1, NULL);

h\_Sem2 = CreateSemaphore(NULL, 0, 1, NULL);

h\_Sem3 = CreateSemaphore(NULL, 0, 1, NULL);

std::thread t1(first);

std::thread t2(second);

std::thread t3(three);

Sleep(300);

while (1)

{

int d = 1 + rand() % 3;

//cout « d « endl;

switch (d)

{

case 1:

if (l != 1) {

l = 1;

Sleep(300);

t1.detach();

cout << "На столе: бумага, спички" << endl;

ReleaseSemaphore(h\_Sem1, 1, NULL);

this\_thread::sleep\_for(chrono::milliseconds(100));

WaitForSingleObject(h\_Sem1, INFINITY);

t1 = thread(first);

}

break;

case 2:

if (l != 2) {

l = 2;

Sleep(300);

t2.detach();

cout << "На столе: табак, спички" << endl;

ReleaseSemaphore(h\_Sem2, 1, NULL);

this\_thread::sleep\_for(chrono::milliseconds(100));

WaitForSingleObject(h\_Sem2, INFINITY);

t2 = thread(second);

}

break;

case 3:

if (l != 3) {

l = 3;

Sleep(300);

t3.detach();

cout << "На столе: табак, бумага" << endl;

ReleaseSemaphore(h\_Sem3, 1, NULL);

this\_thread::sleep\_for(chrono::milliseconds(100));

WaitForSingleObject(h\_Sem3, INFINITY);

t3 = thread(three);

}

break;

default:

break;

}

Sleep(1000);

}

return 0;

}

**5. Вывод**

В ходе выполнения лабораторной работы были получены навыки работы с семафорами для условной синхронизации. В контексте данной задачи их использование является не целесообразным из-за ряда своих особенностей:

1) В задаче курит всегда один, поэтому максимальное кол-во потоков, которое мы можем пропустить за семафор – 1.

2) Семафоры в отличие от мьютексов не забираются тем потоком, который их ожидает. Именно поэтому еще одним критическим ресурсом, помимо функций курильщиков, является консоль.

Подводя итог всему вышесказанному, можно сделать предположение о том, что для условной синхронизации лучше всего подошёл мьютекс.