

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ОТЧЁТ
Лабораторная работа №3
“Алгоритмы синхронизации процессов”

Выполнил: Заяц Д. А., Готин И.

Проверил: Цирук В. А.

Минск 2022

Цель:

Изучить алгоритмы синхронизации процессов и средства операционной системы, позволяющие осуществлять синхронизацию.

Вариант задания: 5.

Условие задания:

Создать консольное приложение, порождающее несколько потоков. Первый поток высчитывает таблицу значений функции (можно взять любую математическую функцию), второй поток осуществляет вывод каждого значения в файл по мере его высчитывания. Третий поток ведет лог процедуры обмена, записывая туда время получения значения функции и время записи данного значения в файл. Каждая пара значений, полученная в процессе вычисления, должна быть занесена в объект класса Point, который может быть уничтожен только тогда, когда информация о нем будет занесена в лог-файл. Обращение к объекту Point должно происходить через потокобезопасный умный указатель.

Выбранный объект синхронизации: семафор.

Семафор (semaphore) – примитив синхронизации работы процессов и потоков, в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующей. Служит для построения более сложных механизмов синхронизации и используется для синхронизации параллельно работающих задач, для защиты передачи данных через разделяемую память, для защиты критических секций.

Почему семафор:

В нашей задаче уместно использование семафора, т.к. в разные моменты работы программы, должно работать разное количество потоков (сначала один поток вычисляет значение u для сгенерированного x , затем 2 потока выполняют запись в файлы.)

Листинг:

Файл laba3.cpp:

```

1  #include <iostream>
2  #include <fstream>
3  #include <thread>
4  #include <semaphore>
5  #include <memory>
6  #include <chrono>
7  #include <ctime>
8  #include "Point.h"
9
10 void WriteResult(std::shared_ptr<Point>, int);
11 void WriteLog(double, std::shared_ptr<Point>, int);
12 void ClearFiles();
13
14 std::counting_semaphore<2> signalToFileWriters(0);
15
16 int main()
17 {
18     srand(time(0));
19     unsigned int numberOfOperations;
20     double startTime;
21     ClearFiles();
22     std::cout << "Enter the number of operations: "; std::cin >> numberOfOperations;
23     auto start = clock() / 1000.0;
24     for (int i = 0; i < numberOfOperations; i++)
25     {
26         auto point = new Point();
27         std::shared_ptr<Point> ptrPoint(point);
28         ptrPoint->SetX(rand() % 500);
29         startTime = clock() / 1000.0;
30         ptrPoint->SetY(ptrPoint->Func());
31         /*WriteLog(startTime, ptrPoint, i);
32         WriteResult(ptrPoint, i);*/
33         signalToFileWriters.release(2);
34         std::thread log(WriteLog, startTime, ptrPoint, i);
35         std::thread write(WriteResult, ptrPoint, i);
36         log.join();
37         write.join();
38     }
39     std::cout << clock() / 1000.0 - start;
40     return 0;

```

```

41 }
42
43 void WriteResult(std::shared_ptr<Point> point, int numberOfOperation)
44 {
45     signalToFileWriters.acquire();
46     std::ofstream fileWriter;
47     fileWriter.open("D:\\121702\\Zayats\\laba3\\files\\results.txt", std::ios::app);
48     fileWriter << numberOfOperation + 1 << ". \tX = " << point->GetX() << "\tY = " << point->GetY() << std::endl;
49     fileWriter.close();
50 }
51
52 void WriteLog(double startTime, std::shared_ptr<Point> point, int numberOfOperation)
53 {
54     signalToFileWriters.acquire();
55     std::ofstream fileLogger;
56     fileLogger.open("D:\\121702\\Zayats\\laba3\\files\\log.txt", std::ios::app);
57     fileLogger << numberOfOperation + 1 << ". \t(" << point->GetX() << ", "
58     << point->GetY() << ")" << " Operation start time: "
59     << startTime << ", operation end time: " << clock()/1000.0 << std::endl;
60     fileLogger.close();
61 }
62
63 void ClearFiles()
64 {
65     std::ofstream file;
66     file.open("D:\\121702\\Zayats\\laba3\\files\\results.txt");
67     file.close();
68     file.open("D:\\121702\\Zayats\\laba3\\files\\log.txt");
69     file.close();
70 }
71

```

Файл Point.h:

```

1  #pragma once
2
3  class Point
4  {
5  private:
6      double x;
7      double y;
8  public:
9      double Func();
10     void SetX(double);
11     double GetX();
12     void SetY(double);
13     double GetY();
14 };

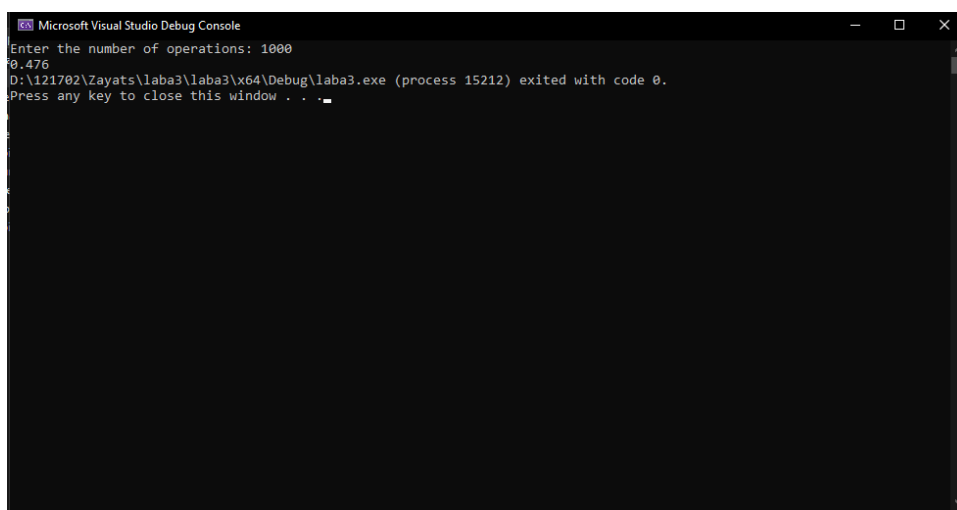
```

Файл Point.cpp:

```
1  #include "Point.h"
2  #include <cmath>
3
4  double Point::Func()
5  {
6      return (pow(x, 3)+pow(x,2) ) / 42;
7  }
8  void Point::SetX(double x)
9  {
10     this->x = x;
11 }
12 double Point::GetX()
13 {
14     return x;
15 }
16 void Point::SetY(double y)
17 {
18     this->y = y;
19 }
20 double Point::GetY()
21 {
22     return y;
23 }
```

Пусть программа выполняет 1000 операций :

Время последовательного выполнения программы:



The screenshot shows the Microsoft Visual Studio Debug Console window. The text inside the console is as follows:

```
Microsoft Visual Studio Debug Console
Enter the number of operations: 1000
0.476
D:\121702\Zayats\laba3\laba3\x64\Debug\laba3.exe (process 15212) exited with code 0.
Press any key to close this window . . .
```

Время выполнения программы с использованием многопоточности:

```
Microsoft Visual Studio Debug Console
Enter the number of operations: 1000
0.249
D:\121702\Zayats\laba3\laba3\x64\Debug\laba3.exe (process 14028) exited with code 0.
Press any key to close this window . . .
```