МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №5

**«Многопоточность»**

Выполнил:

студент 3 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверил:

Козик И. Д.

**Цель работы**: изучить, научиться и разобраться, как работать с многопоточностью в приложениях, выполняющихся на операционной системе Microsoft Windows, версий 10 и выше.

**Вариант №3**

Необходимо написать 2 программы, используя в них несколько потоков. Одну из программ реализовать через атомные переменные, вторую – через mutex.

Создать три потока, выполняющих различные арифметические операции над переменной типа float.

**Код программы**

**atomic.cpp**

#include <iostream>

#include <thread>

#include <atomic>

using namespace std;

int main()

{

for (int i = 0; i < 10; i++) {

cout << i << ":\t";

atomic<float> variable{ 0 };

thread thread\_plus([&variable]()

{

// fetch\_add() только для atomic<int> и др.;

// с Floating типами незя

variable = variable + 1.0;

cout << "+2.0 ";

});

thread thread\_minus([&variable]()

{

variable = variable - 1.0;

cout << "-1.0 ";

});

thread thread\_multiplier([&variable]()

{

variable = variable \* 1.5;

cout << "\*1.5 ";

});

// Ждем завершения потоков

thread\_plus.join();

thread\_minus.join();

thread\_multiplier.join();

cout << "\t" << variable << endl;

}

return 0;

}

**mutex.cpp**

#include <iostream>

#include <thread>

#include <mutex>

using namespace std;

int main()

{

for (int i = 0; i < 10; i++) {

cout << i << ":\t";

float variable = 0;

mutex variable\_mutex;

thread thread\_plus([&variable, &variable\_mutex]()

{

variable\_mutex.lock();

variable = variable + 2.0;

cout << "+2.0 ";

variable\_mutex.unlock();

});

thread thread\_minus([&variable, &variable\_mutex]()

{

variable\_mutex.lock();

variable = variable - 1.0;

cout << "-1.0 ";

variable\_mutex.unlock();

});

thread thread\_multiplier([&variable, &variable\_mutex]()

{

variable\_mutex.lock();

variable = variable \* 1.5;

cout << "\*1.5 ";

variable\_mutex.unlock();

});

thread\_plus.join();

thread\_minus.join();

thread\_multiplier.join();

cout << "\t" << variable << endl;

}

return 0;

}

**Пример работы программ:**

mutex:

0: -1.0 +2.0 \*1.5 1.5

1: -1.0 +2.0 \*1.5 1.5

2: +2.0 -1.0 \*1.5 1.5

3: +2.0 -1.0 \*1.5 1.5

4: -1.0 +2.0 \*1.5 1.5

5: -1.0 +2.0 \*1.5 1.5

6: +2.0 -1.0 \*1.5 1.5

7: +2.0 -1.0 \*1.5 1.5

8: +2.0 -1.0 \*1.5 1.5

9: +2.0 \*1.5 -1.0 2 .

atomic:

0: -1.0 +2.0 \*1.5 0

1: +2.0 -1.0 \*1.5 0

2: -1.0 \*1.5 +2.0 -0.5

3: +2.0 \*1.5 -1.0 0.5

4: -1.0 \*1.5 +2.0 -0.5

5: +2.0 -1.0 \*1.5 0

6: +2.0 -1.0 \*1.5 0

7: +2.0 -1.0 \*1.5 0

8: +2.0 -1.0 \*1.5 0

9: +2.0 -1.0 \*1.5 0 .