|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ИУ8 “КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ”

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

Студент Лустин Павел Евгеньевич

*(*Фамилия И.О)

Группа\_\_ИУ8-24\_\_\_\_\_\_\_\_

Студент  **\_\_\_\_\_\_15.05\_\_\_\_\_\_\_\_\_**  Лустин \_\_ \_

*дата (*Фамилия И.О)

Преподаватель **\_\_\_\_\_\_15.05\_\_\_\_\_\_\_**  Барыкин Дмитрий

*дата (*Фамилия И.О)*.*

*2024 г.*

Цель работы

Цель работы состоит в изучении использования многопоточности при работе с функциями

Условие задачи(вариант 4)

## Реализовать программу, в которой кроме главного создается три отдельных потока: первый поток сортирует первую половину вещественного массива, второй поток сортирует вторую половину вещественного массива, третий поток запускается после завершение первых двух, он сортирует массив полностью (уже частично отсортированный). Каждый поток имеет свое имя (например, thread1, thread2, thread3), и печатает отсортированный массив, перед печатью каждого значения элемента массива поток должен напечатать с новой строки свое имя. После завершения дочерних потоков главный поток выдает сообщение об окончании работы. Имена потоков и массив передаются в потоковую функцию через ее параметры, использовать одну потоковую функцию для всех трех потоков. Массив предварительно до запуска потоков заполняются числами с использованием ГПСЧ. Для сортировки использовать метод прямого обмена (метод «пузырька»).

## Запустить программу несколько раз при одних и тех же исходных данных, посмотреть, как меняются результаты вывода. Сделать выводы.

## Между печатью имени потока и значением установить небольшую задержку, например, 10 мс. Посмотреть, как меняются результаты вывода. Сделать выводы.

Текст программы с комментариями

#include <iostream>

#include <thread>

#include <string>

#include <vector>

using namespace std;

void sort(string name, vector <double>& vec) //Функция для сортировки массива вещественных чисел методом пузырька при помощи 3-х потоков

{

double temp = 0;

int a = 0;

if (name == "tread1") a = 1; //Определяем, какую часть кода функции будет выполнять поток

if (name == "tread2") a = 2;

if (name == "tread3") a = 3;

switch(a) //Используем для разделения кода функции на 3 блока, для каждого потока соответственно

{

case 1:

{

for (int j = 0; j < size(vec) / 2; ++j) //Цикл сортировки первой половины массива методом пузырька

{

for (int i = 0; i < (size(vec) / 2) - 1; ++i)

{

if (vec[i] > vec[i + 1])

{

temp = vec[i];

vec[i] = vec[i + 1];

vec[i + 1] = temp;

}

}

}

for (int i = 0; i < size(vec); ++i) //Функция печати отсортированного массива

{

cout << "tread1" << endl;

this\_thread::sleep\_for(chrono::milliseconds(10)); //Устанавливаем задержку на 10мс

cout << vec[i] << endl;

}

break;

}

case 2:

{

for (int j = 0; j < size(vec); ++j) //Цикл сортировки второй половины массива методом пузырька

{

for (int i = 0; i < (size(vec)) - 1; ++i)

{

if (i < size(vec)/2) continue; //Пропускаем половину массива для работы с его второй половиной

if (vec[i] > vec[i + 1])

{

temp = vec[i];

vec[i] = vec[i + 1];

vec[i + 1] = temp;

}

}

}

for (int i = 0; i < size(vec); ++i)

{

cout << "tread2" << endl;

this\_thread::sleep\_for(chrono::milliseconds(10));

cout << vec[i] << endl;

}

break;

}

case 3:

{

for (int j = 0; j < size(vec); ++j) //Цикл сортировки всего массива методом пузырька

{

for (int i = 0; i < (size(vec)) - 1; ++i)

{

if (vec[i] > vec[i + 1])

{

temp = vec[i];

vec[i] = vec[i + 1];

vec[i + 1] = temp;

}

}

}

for (int i = 0; i < size(vec); ++i)

{

cout << "tread3" << endl;

this\_thread::sleep\_for(chrono::milliseconds(10));

cout << vec[i] << endl;

}

break;

}

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int n = 10;

double temp = 0;

vector <double> vec;

for(int i = 0; i < n; ++i) //Заполняем начальный вектор vec псевдослучайными вещественными числами

{

temp =(double) rand() \* 100 / RAND\_MAX;

vec.push\_back(temp);

}

cout << "Начальный массив: " << endl;

for (int i = 0; i < size(vec); ++i)

{

cout << vec[i] << " ";

}

cout << endl;

thread tread1(sort, "tread1", ref(vec)); //Включаем первый дочерний поток

thread tread2(sort, "tread2", ref(vec)); //Включаем второй дочерний поток

tread1.join(); //Дожидаемся конца работы первого потока

tread2.join(); //Дожидаемся конца работы второго потока

thread tread3(sort, "tread3", ref(vec)); //Включаем третий дочерний поток после выполнения первых двух

tread3.join(); //Дожидаемся конца работы третьего потока

cout << "Работа окончена";

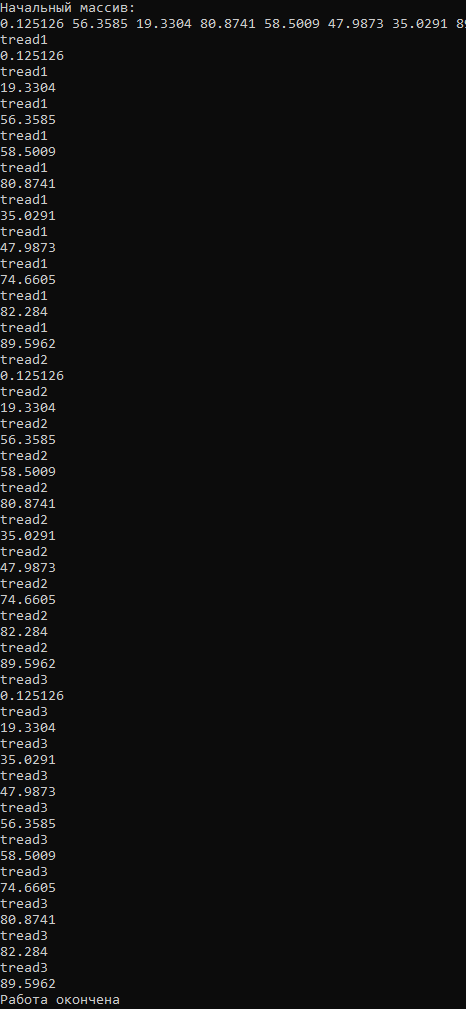
return 0;

}

Вывод 1:

При запуске программы несколько раз при одних и тех же исходных данных результаты вывода никак не меняются.

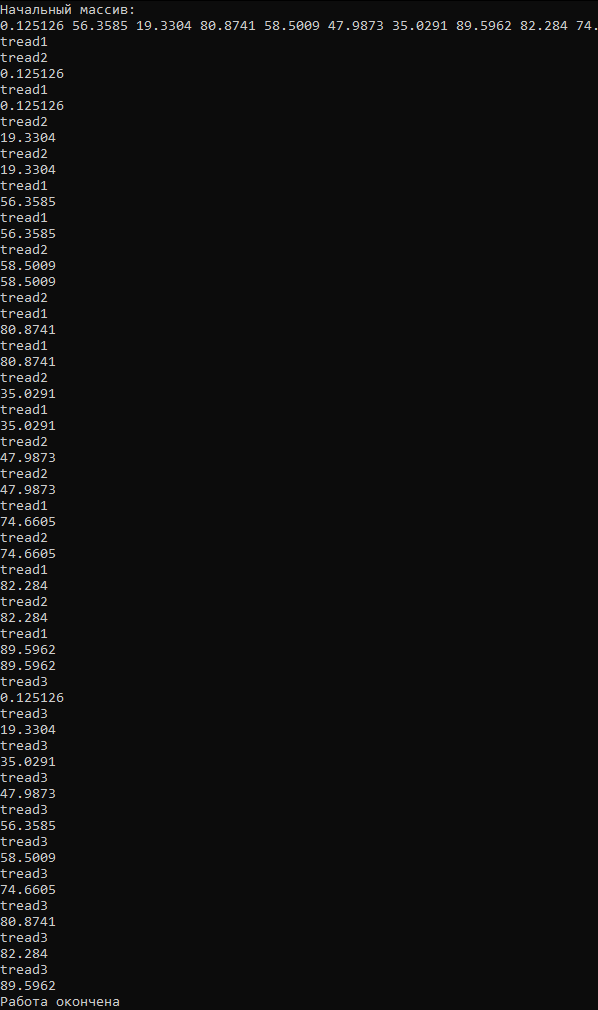
Значит, несмотря на то, что мы работаем над одним и тем же массивом при помощи нескольких разных потоков одновременно, результат не меняется, так как каждый раз программа выполняется за одинаковое время и каждая команда так же выполняется за одно и то же время.



Вывод 2:

Если поставить между печатью имени потока и значением небольшую задержку, равную 10 мс, то при выводе значений в консоли результаты перемешиваются.

То есть получается так, что массив выводится в консоль при помощи двух потоков одновременно и из-за задержки вывод в консоль становится неправильным. Перемешиваются значения и названия потоков, хоть числа и сохраняют свою отсортированную последовательность.



Вывод:

Я научился использовать многопоточность при работе с функциями