|  |  |
| --- | --- |
|  | **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» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 7

ПО КУРСУ

«Алгоритмические языки»

на тему «**Изучение потоковой многозадачности**»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-23 |  |  |  | Д.С. Афанасьев |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | М. В. Малахов |
|  |  |  |  |  | (И.О. Фамилия) |

2024

Введение

Условия для 4 варианта

***Представить 2 варианта программы: использование шаблона future и функции async и использование класса thread.***

## 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.

Основная часть

Исходный текст программы:

***использование шаблона future и функции async***

#include <condition\_variable>

#include <iostream>

#include <chrono>

#include <mutex>

#include <random>

#include <future>

*//  g++ source/lab7/Source2.cpp -o build/lab7\_future -w*

std::mutex cout\_guard;

*// std::condition\_variable condition\_variable;*

void sort\_from\_l\_to\_r(double\* arr, int l, int r, char\* thread\_name) {

  for (int i = l - 1; i < r; ++i) {

    for (int j = i + 1; j < r; ++j) {

      if (arr[j] < arr[j - 1]) {

        std::swap(arr[j], arr[j - 1]);

      }

    }

  }

*// std::sort(arr + l - 1, arr + r);*

  std::cout << std::endl;

  for (int i = l - 1; i < r; ++i) {

*// const std::lock\_guard<std::mutex> lock(cout\_guard);*

    cout\_guard.lock();

    std::cout << std::endl

              << thread\_name << ' ';

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

    std::cout << arr[i];

    cout\_guard.unlock();

  }

*// condition\_variable.notify\_one();*

}

const int ARR\_SIZE = 20;

void Working() {

  char\* thread\_name1 = "thread1";

  char\* thread\_name2 = "thread2";

  char\* thread\_name3 = "thread3";

  srand(time(NULL));

  double array[ARR\_SIZE];

  for (int i = 0; i < ARR\_SIZE; i++) {

    array[i] = static\_cast<double>(1 + rand() % 2002) / 100.0;

  }

  std::pair<int, int> first\_half = std::make\_pair(1, ARR\_SIZE / 2);

  std::pair<int, int> second\_half = std::make\_pair(ARR\_SIZE / 2 + 1, ARR\_SIZE);

*// Sorting the first half of array*

  std::future<void> thread1 = std::async(sort\_from\_l\_to\_r,

                      array, first\_half.first, first\_half.second, thread\_name1);

*// Sorting the second half of array*

  std::future<void> thread2 = std::async(sort\_from\_l\_to\_r,

                      array, second\_half.first, second\_half.second, thread\_name2);

  thread1.wait(); thread2.wait();

*// Sorting the full array*

  std::future<void> thread3 = std::async(sort\_from\_l\_to\_r,

                      array, first\_half.first, second\_half.second, thread\_name3);

  thread3.wait();

  std::cout << std::endl

            << "All threads have completed their tasks";

}

int main()

{

  const auto start\_time = std::chrono::steady\_clock::now();

  Working();

  const auto end\_time = std::chrono::steady\_clock::now();

  const auto elapsed\_time = std::chrono::duration\_cast<std::chrono::milliseconds>(end\_time - start\_time);

*// std::cout << std::endl*

*//          << "The array in main thread is "*

*//          << std::endl;*

*// for (int i = 0; i < ARR\_SIZE; ++i) {*

*//  std::cout << array[i] << ' ';*

*// }*

  std::cout << std::endl

            << "Main thread has stopped working"

            << std::endl;

  std::cout << "It took " << elapsed\_time.count() << " milliseconds"

            << std::endl;

  return 0;

}

***использование класса thread***

#include <algorithm>

#include <atomic>

#include <chrono>

#include <condition\_variable>

#include <iostream>

#include <mutex>

#include <thread>

#include <random>

*//  g++ source/lab7/Source1.cpp -o build/lab7\_thread -w*

std::condition\_variable condition\_variable;

std::mutex cout\_guard;

void sort\_from\_l\_to\_r(double\* arr, int l, int r, char\* thread\_name) {

  for (int i = l - 1; i < r; ++i) {

    for (int j = i + 1; j < r; ++j) {

      if (arr[j] < arr[j - 1]) {

        std::swap(arr[j], arr[j - 1]);

      }

    }

  }

*// std::sort(arr + l - 1, arr + r);*

  std::cout << std::endl;

  for (int i = l - 1; i < r; ++i) {

    const std::lock\_guard<std::mutex> lock(cout\_guard);

*// cout\_guard.lock();*

    std::cout << std::endl

              << thread\_name << ' ';

    std::this\_thread::sleep\_for(std::chrono::milliseconds(20));

    std::cout << arr[i];

*// cout\_guard.unlock();*

  }

  condition\_variable.notify\_one();

}

const int ARR\_SIZE = 20;

int main() {

  char\* thread\_name1 = "thread1";

  char\* thread\_name2 = "thread2";

  char\* thread\_name3 = "thread3";

  srand(time(NULL));

  double array[ARR\_SIZE];

  for (int i = 0; i < ARR\_SIZE; i++) {

    array[i] = static\_cast<double>(1 + rand() % 2002) / 100.0;

  }

  std::pair<int, int> first\_half = std::make\_pair(1, ARR\_SIZE / 2);

  std::pair<int, int> second\_half = std::make\_pair(ARR\_SIZE / 2 + 1, ARR\_SIZE);

*// Sorting the first half of array*

  std::thread thread1(sort\_from\_l\_to\_r,

                      array, first\_half.first, first\_half.second, thread\_name1);

*// Sorting the second half of array*

  std::thread thread2(sort\_from\_l\_to\_r,

                      array, second\_half.first, second\_half.second, thread\_name2);

  {

    std::unique\_lock<std::mutex> lock(cout\_guard);

    condition\_variable.wait\_for(lock,

                      std::chrono::duration<double, std::milli>(1e9));

    condition\_variable.wait\_for(lock,

                      std::chrono::duration<double, std::milli>(1e9));

  }

*// Sorting the full array*

  std::thread thread3(sort\_from\_l\_to\_r,

                      array, first\_half.first, second\_half.second, thread\_name3);

  {

    std::unique\_lock<std::mutex> lock(cout\_guard);

    condition\_variable.wait\_for(lock,

                      std::chrono::duration<double, std::milli>(1e9));

    std::cout << std::endl

              << "The threads have finished working";

  }

  thread1.join();

  thread2.join();

  thread3.join();

*// std::cout << std::endl*

*//          << "The array in main thread is "*

*//          << std::endl;*

*// for (int i = 0; i < ARR\_SIZE; ++i) {*

*//  std::cout << array[i] << ' ';*

*// }*

  std::cout << std::endl

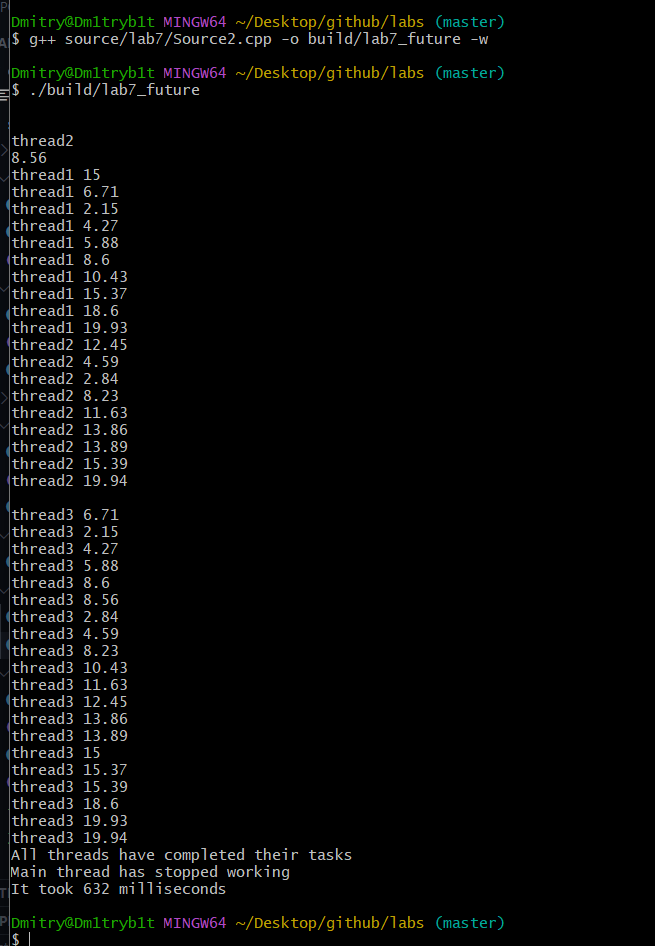
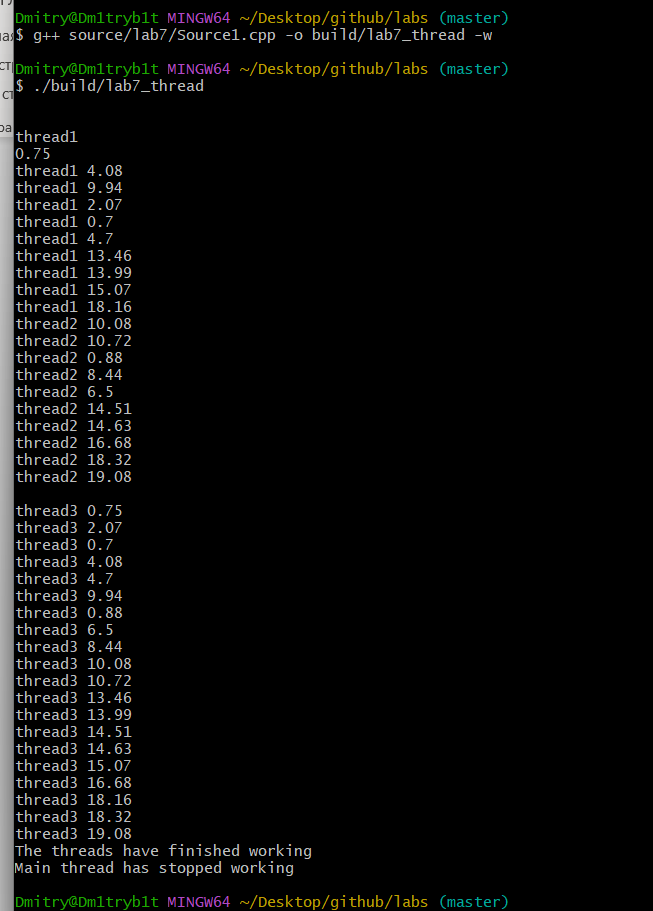
            << "Main thread has stopped working"

            << std::endl;

  return 0;

}

Снимки выполнения работы программы

Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучены на практике методы многопоточного программирования на языке C++.