Отчёт по ДЗ №8

**Отчёт о выполнении:**

Для выполнения задания были реализованы функции FillArrayA, FillArrayB, ArrayProduct, ArrayProductMultiThread и MeasureArrayProduct с сигнатурами:

constexpr void FillArrayA(double\*, size\_t)

constexpr void FillArrayB(double\*, size\_t)

constexpr void\* ArrayProduct(void\*)

template<size\_t> void ArrayProductMultiThread(const double\*, const double\*, double\*, size\_t)

template <size\_t> void MeasureArrayProduct(const double\*, const double\*, double\*, size\_t)

Функции FillArrayA и FillArrayA заполняют исходные массивы согласно условию (по формуле A[i] = i + 1, B[i] = n – i, где n – размер массива)

Функция MeasureArrayProduct вывод в консоль информацию о том, с какими параметрами будет происходить вызов функции ArrayProductMultiThread, а также измеряет время работы функции.

Функция ArrayProductMultiThread имеет шаблонный параметр kThreadsCount (см. Google C++ codestyle), который указывает на количество параллельных запусков функции ArrayProduct. Функция создаёт kThreadsCount – 1 потоков (threads), в которых запускается функция ArrayProduct с необходимыми параметрами. Последний (kThreadsCount-ый) вызов функции ArrayProduct делается в основном потоке, в котором изначально была вызвана функция main.

Для каждого размера входных данных запускается 5 функций MeasureArrayProduct с количествами потоков: 1, 2, 4, 8 и 1000.

На системе с Windows 10 (без использования WSL2) были произведены измерения для всех размеров входных данных: 100000000, 500000000 и 1000000000.  
При использовании WSL2 ОС не смогла выделить память для 3 массивов из 1000000000 элементов.  
При использовании Ubuntu 22.04 (4 gb ram, 4 threads) ОС не смогла выделить память для 3 массивов из 500000000 и 1000000000 элементов.

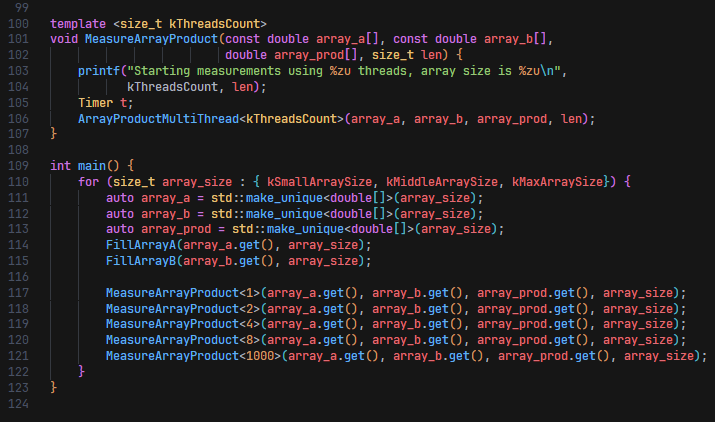
В курсе не было указано, что при написании кода должен использоваться какой-то конкретный кодстайл. В данной реализации используется Google C++ code style (<https://google.github.io/styleguide/cppguide.html>)

**Исходный код программы:**

”main.cpp”:



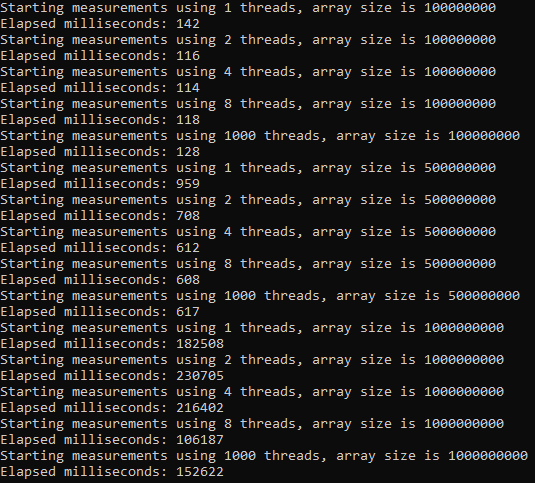




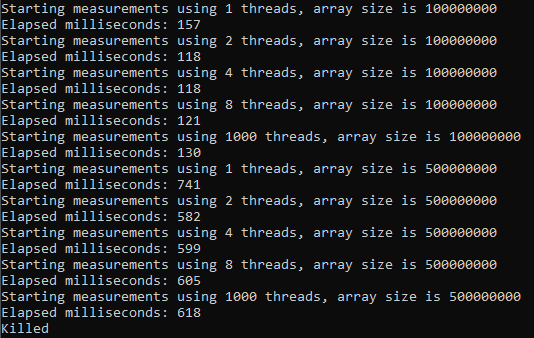
Также исходный код можно найти в файле main.cpp, находящимся в одном архиве с этим отчётом.

**Скриншоты работы программы:**

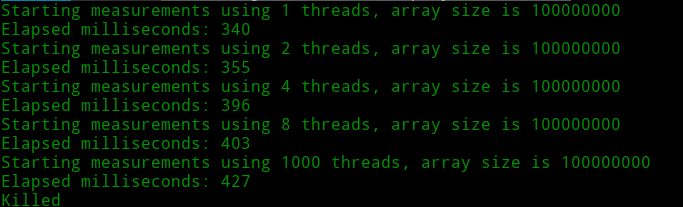
Windows 10 (16 gb ram, 16 threads):



Windows 10 with WSL2 (16 gb ram, 16 threads):



Ubuntu 22.04 (4 gb ram, 4 threads):



Таким образов, можно сформировать таблицу (для Windows 10 без WSL2):

|  |  |  |
| --- | --- | --- |
| Число элементов в векторе | Число потоков | Время выполнения |
| 100000000 | 1 | 142ms |
| 100000000 | 2 | 116 ms |
| 100000000 | 4 | 114 ms |
| 100000000 | 8 | 118 ms |
| 100000000 | 1000 | 128 ms |
| 500000000 | 1 | 959 ms |
| 500000000 | 2 | 708 ms |
| 500000000 | 4 | 612 ms |
| 500000000 | 8 | 608 ms |
| 500000000 | 1000 | 617 ms |
| 1000000000 | 1 | 182508 ms |
| 1000000000 | 2 | 230705 ms |
| 1000000000 | 4 | 216402 ms |
| 1000000000 | 8 | 106187 ms |
| 1000000000 | 1000 | 152622 ms |

Как видно из результатов тестирования, при использовании 1000 потоков общее время работы программы увеличивается по сравнению с 8 потоками. Возможно, это можно объяснить тем, что потоки создаются достаточно длительное время, и создание 1000 потоков избыточно для решения данной задачи.

Т.к. при увеличении входных данных с 100000000 до 500000000 время работы увеличилось примерно в 5-7 раз, то можно предположить, что своппинга не происходит.

Однако, при увеличении размера входных данных с 100000000 до 1000000000 время работы увеличилось в среднем в 1453 раза, также при работе программы ОС задействовала часть системного диска (программа использует порядка 24 gb ram, но в системе только 16 gb ram, часть которых уже занята ядром ОС и другими системными службами), что указывает на то, что происходит своппинг.

Программа компилировалась с параметрами:

1. На Windows 10:
   1. Компилятор:  
      g++ version 13.2.0, built by msys2 project, environment mingw w64
   2. Версия языка: C++ 20
   3. Флаги компиляции:  
      -std=c++2a -O2 -march=native -Wall -Wextra -Werror -Wunused -Wconversion -Wshadow -Wnull-dereference -Wsign-conversion -Wcast-align=strict -Warray-bounds=2 -lpthread
2. На Windows 10 в WSL2 с подсистемой Ubuntu 20.04.2:
   1. Компилятор:  
      g++ version 9.4.0
   2. Версия языка: C++ 20
   3. Флаги компиляции:  
      -std=c++2a -O2 -march=native -Wall -Wextra -Werror -Wunused -Wconversion -Wshadow -Wnull-dereference -Wsign-conversion -Wcast-align=strict -Warray-bounds=2 -lpthread
3. На Ubuntu 22.04:
   1. Компилятор:  
      g++ version 11.4.0
   2. Версия языка: C++ 20
   3. Флаги компиляции:  
      -std=c++20 -O3 -DNDEBUG -Wall -Wextra -Wshadow -Wformat=2 -Wconversion -Wnull-dereference -Wsign-conversion -Wcast-align -Warith-conversion -Warray-bounds=2

aвтор: кормилицын владимир, БПИ 226