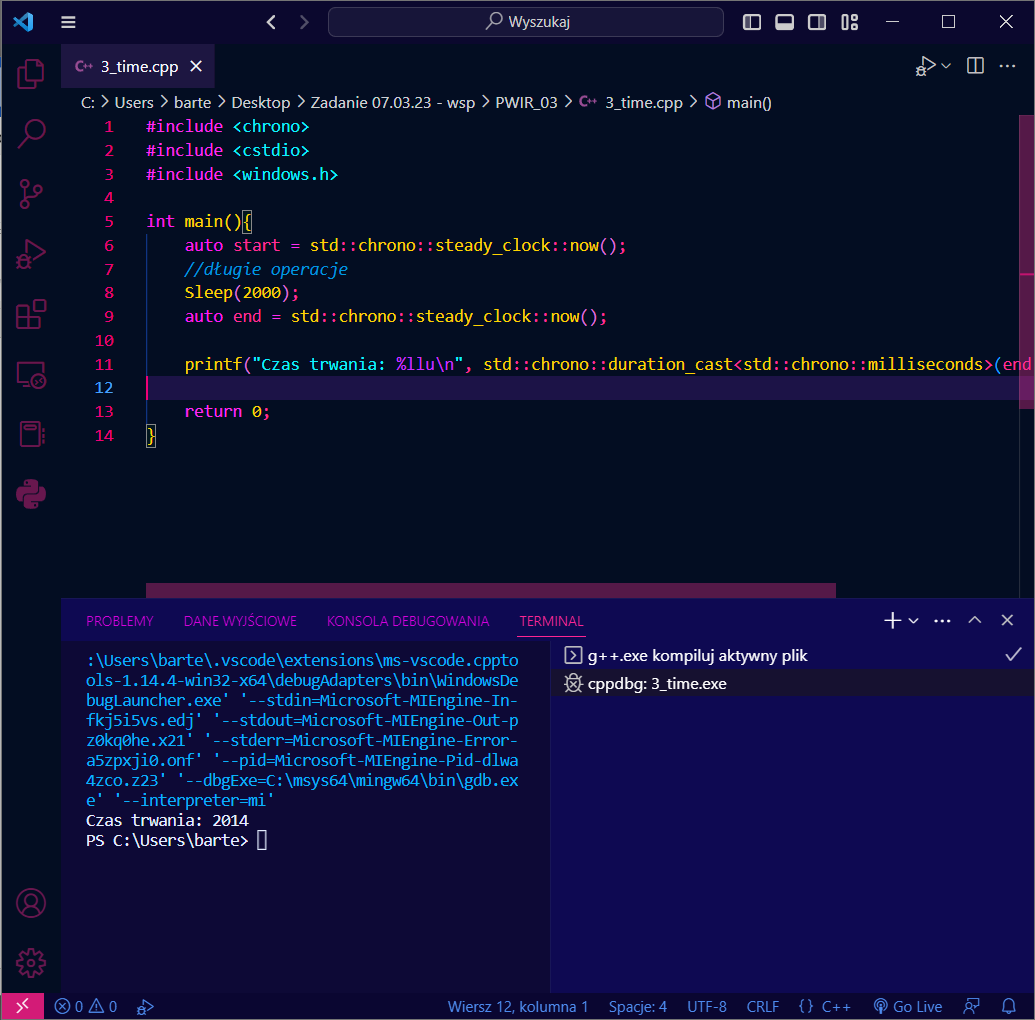
Bartłomiej Cetera – 07,03,2023

**ZAD 3 – podpunkt 1**

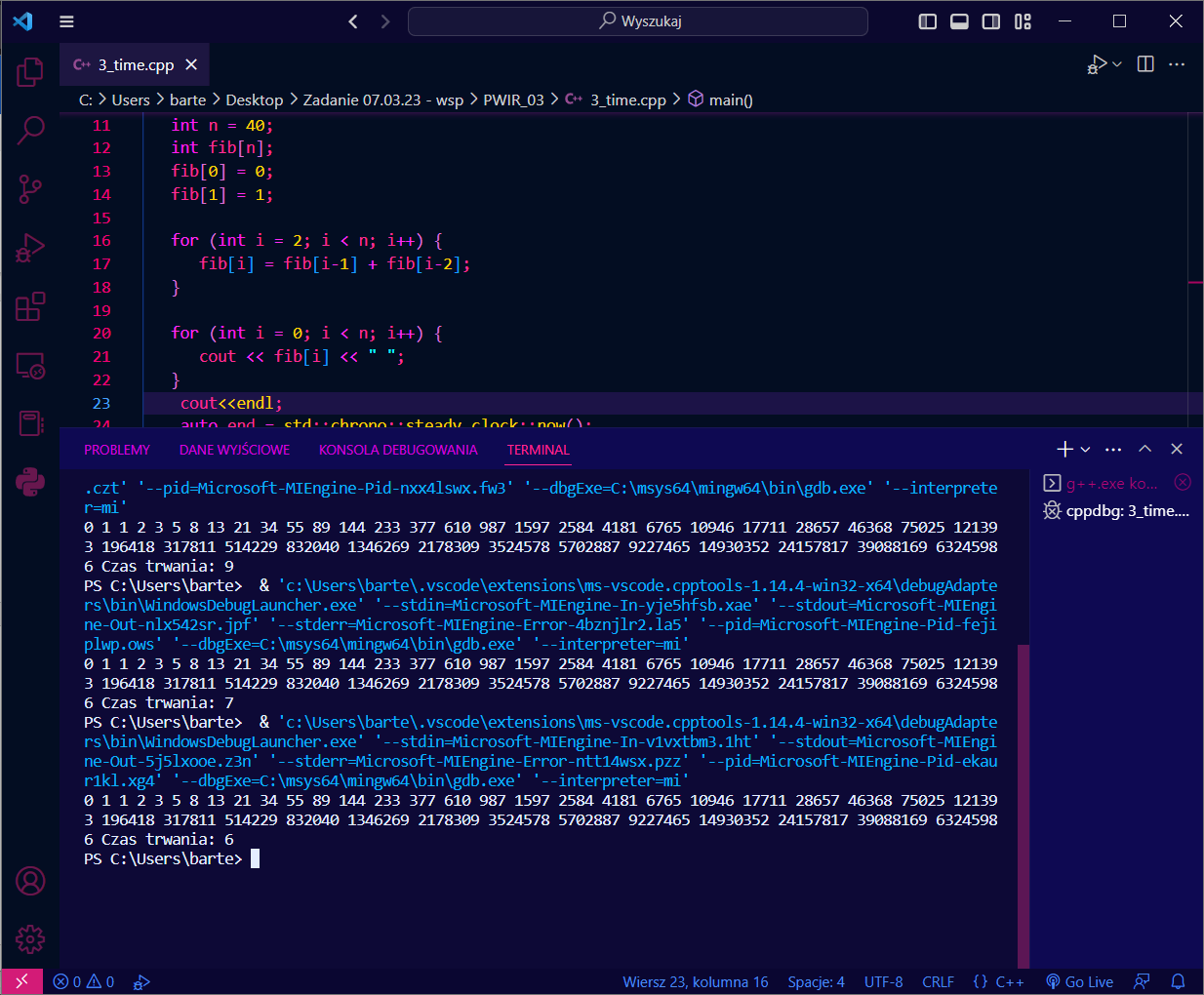


**Czas trwania po pierwszym uruchomieniu – 2014 - bez Sleep(2000) -14**

**Czas trwania po drugim uruchomieniu – 2007 - bez Sleep(2000) - 7**

**Czas trwania po trzecim uruchomieniu – 2003 -bez Sleep(2000) – 3**

**ZAD 3 – podpunkt 2**

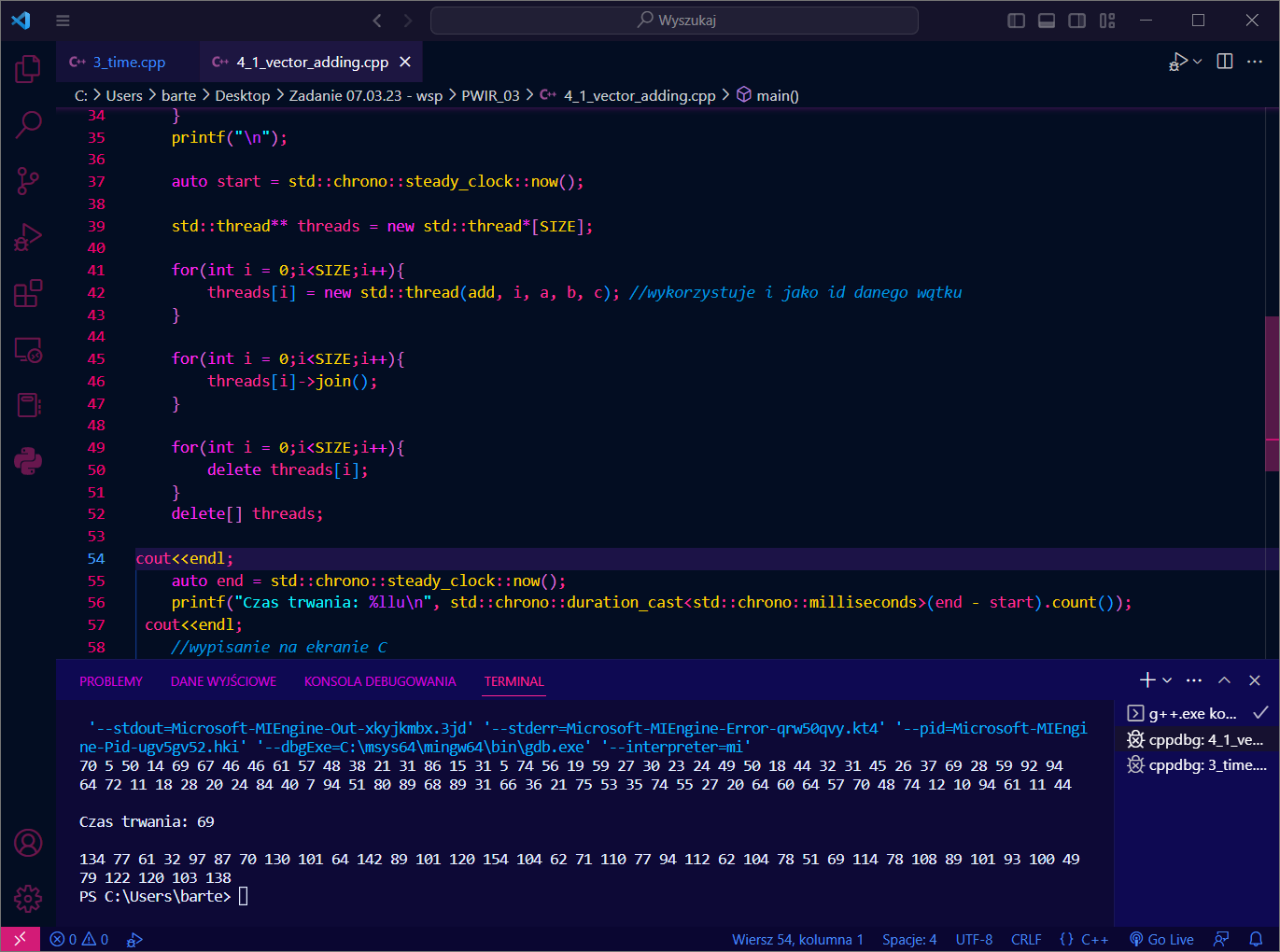


**Czas trwania po pierwszym uruchomieniu – 9**

**Czas trwania po drugim uruchomieniu – 7**

**Czas trwania po trzecim uruchomieniu – 6**

**ZAD 4 – podpunkt 1**



**Czas trwania po pierwszym uruchomieniu – 69**

**Czas trwania po drugim uruchomieniu – 59**

**Czas trwania po trzecim uruchomieniu – 63**

#include <chrono>

#include <cstdio>

#include <windows.h>

#include<thread>

#include <iostream>

using namespace std;

#define SIZE 40

void add(int id, int\* a, int\* b, int\* c){

    c[id] = a[id] + b[id];

}

int main(){

    srand(time(NULL));

    int a[SIZE];

    int b[SIZE];

    int c[SIZE];

    for(int i = 0;i<SIZE;i++){

        a[i] = rand() % 100 + 1; *//1 do 100*

        b[i] = rand() % 100 + 1;

    }

*//wypisanie na ekranie A*

    for(int i = 0;i<SIZE;i++){

        printf("%u ", a[i]);

    }

    printf("\n");

*//wypisanie na ekranie B*

    for(int i = 0;i<SIZE;i++){

        printf("%u ", b[i]);

    }

    printf("\n");

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

    std::thread\*\* threads = new std::thread\*[SIZE];

    for(int i = 0;i<SIZE;i++){

        threads[i] = new std::thread(add, i, a, b, c); *//wykorzystuje i jako id danego wątku*

    }

    for(int i = 0;i<SIZE;i++){

        threads[i]->join();

    }

    for(int i = 0;i<SIZE;i++){

        delete threads[i];

    }

    delete[] threads;

cout<<endl;

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

    printf("Czas trwania: %llu\n", std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count());

 cout<<endl;

*//wypisanie na ekranie C*

    for(int i = 0;i<SIZE;i++){

        printf("%u ", c[i]);

    }

    return 0;

}

**ZAD 4 – podpunkt 2**

#include <cstdio>

#include <cstdlib>

#include <time.h>

#include <thread>

#include <chrono>

#define SIZE 100

#define THREAD\_COUNT 10

#define CHUNK\_SIZE (SIZE / THREAD\_COUNT)

void add\_chunk(int id, int\* a, int\* b, int\* c){

int start = id \* CHUNK\_SIZE;

int end = start + CHUNK\_SIZE;

for(int i = start; i < end; i++){

c[i] = a[i] + b[i];

}

}

int main(){

srand(time(NULL));

int a[SIZE];

int b[SIZE];

int c[SIZE];

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

a[i] = rand() % 100 + 1; //1 do 100

b[i] = rand() % 100 + 1;

}

//wypisanie na ekranie A

printf("A: ");

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

printf("%u ", a[i]);

}

printf("\n");

//wypisanie na ekranie B

printf("B: ");

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

printf("%u ", b[i]);

}

printf("\n");

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

std::thread\*\* threads = new std::thread\*[THREAD\_COUNT];

for(int i = 0;i<THREAD\_COUNT;i++){

threads[i] = new std::thread(add\_chunk, i, a, b, c); //wykorzystuje i jako id danego wątku

}

for(int i = 0;i<THREAD\_COUNT;i++){

threads[i]->join();

}

for(int i = 0;i<THREAD\_COUNT;i++){

delete threads[i];

}

delete[] threads;

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

//wypisanie na ekranie C

printf("C: ");

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

printf("%u ", c[i]);

}

printf("\nCzas trwania: %llu\n", std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count());

return 0;

}

**ZAD 4 – podpunkt 3**

Aby umożliwić pobieranie rozmiaru tablic a, b i c od użytkownika, należy dodać kod, który zapyta użytkownika o wielkość tablic i przypisze ją do odpowiednich zmiennych. Można to zrobić, np. poprzez wykorzystanie funkcji scanf w celu wczytania liczby całkowitej podanej przez użytkownika i przypisanie jej do zmiennych reprezentujących rozmiary tablic.

Wprowadzenie takiej modyfikacji umożliwi użytkownikowi dostosowanie rozmiaru tablic do swoich potrzeb, co może być szczególnie istotne w przypadku bardziej złożonych programów, gdzie użytkownik może chcieć pracować na większych lub mniejszych danych.

#include <cstdio>

#include <cstdlib>

#include <time.h>

#include <thread>

#include <chrono>

#define THREAD\_COUNT 10

void add\_chunk(int id, int\* a, int\* b, int\* c, int size){

int start = id \* size / THREAD\_COUNT;

int end = start + size / THREAD\_COUNT;

for(int i = start; i < end; i++){

c[i] = a[i] + b[i];

}

}

int main(){

srand(time(NULL));

int size;

printf("Podaj rozmiar tablic: ");

scanf("%d", &size);

int\* a = new int[size];

int\* b = new int[size];

int\* c = new int[size];

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

a[i] = rand() % 100 + 1; //1 do 100

b[i] = rand() % 100 + 1;

}

//wypisanie na ekranie A

printf("A: ");

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

printf("%u ", a[i]);

}

printf("\n");

//wypisanie na ekranie B

printf("B: ");

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

printf("%u ", b[i]);

}

printf("\n");

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

std::thread\*\* threads = new std::thread\*[THREAD\_COUNT];

for(int i = 0;i<THREAD\_COUNT;i++){

threads[i] = new std::thread(add\_chunk, i, a, b, c, size); //wykorzystuje i jako id danego wątku

}

for(int i = 0;i<THREAD\_COUNT;i++){

threads[i]->join();

}

for(int i = 0;i<THREAD\_COUNT;i++){

delete threads[i];

}

delete[] threads;

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

//wypisanie na ekranie C

printf("C: ");

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

printf("%u ", c[i]);

}

printf("\nCzas trwania: %llu\n", std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count());

delete[] a;

delete[] b;

delete[] c;

return 0;

}