МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**Робочий план**

з курсу «Об’єктно-орієнтоване програмування»

Практична робота №10

Солом’яного Ярослава Сергійовича

*ПІБ студента*

студента групи **2П-18**

|  |  |  |
| --- | --- | --- |
| **Вид роботи** | **Дата** | **Оцінка** |
| *Пром. контроль* |  |  |
| *Залік* |  |  |

Викладач Марченко С. В.

Черкаси-2021

Завдання 1. Припустимо, що потрібно написати програмне забезпечення для приготування страви. Наприклад, у замовленні в Mcdonalds працюватиме 3 задачі: з приготування бургеру, картоплі фрі та коли. Проте замовлення буде готове до подачі лише тоді, коли будуть готові всі його компоненти. Реалізуйте різні підходи до реалізації даного завдання та засічіть час виконання окремих задач і коду в цілому:

1. Послідовні задачі:



Рис 1. Послідовність виконання завдання

Лістинг 1. Код для виконання завдання

using System;

using System.Diagnostics;

using System.Threading;

using System.Threading.Tasks;

using System.Text;

using System.Globalization;

namespace Practice\_10\_ex\_1\_McDonalds

{

class Program

{

static void Main(string[] args)

{

OrderAsync();

Console.ReadLine();

}

static async void OrderAsync()

{

Console.OutputEncoding = Encoding.UTF8;

var stopwatch = Stopwatch.StartNew();

await Task.Delay(3000)

.ContinueWith(task => ShowText("Картошка фрі", stopwatch.Elapsed));

await Task.Delay(1000)

.ContinueWith(task => ShowText("Кола", stopwatch.Elapsed));

await Task.Delay(5000)

.ContinueWith(task => ShowText("Бургер", stopwatch.Elapsed));

ShowText("Заказ", stopwatch.Elapsed);

stopwatch.Stop();

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П18");

}

static void ShowText(string name, TimeSpan time)

{

Console.WriteLine($"{name} Приготувався(-лась) за {time}");

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

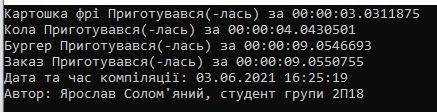


Рис 2. Результат роботи програми

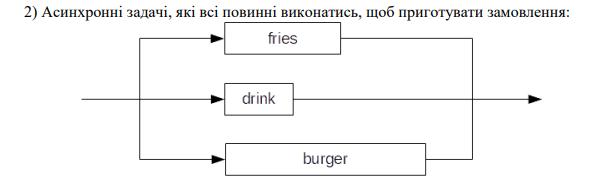


Рис 3. Постанова задачі

Лістинг 2. Код для виконання 2 пункту завдання 1.

using System;

using System.Diagnostics;

using System.Threading;

using System.Threading.Tasks;

using System.Text;

using System.Globalization;

namespace Practice\_10\_ex\_1\_McDonalds

{

class Program

{

static void Main(string[] args)

{

OrderAsync();

Console.ReadLine();

}

static async void OrderAsync()

{

Console.OutputEncoding = Encoding.UTF8;

var stopwatch = Stopwatch.StartNew();

var friesTask = Task.Delay(3000).ContinueWith(task => ShowText("Картошка фрі", stopwatch.Elapsed));

var drinkTask = Task.Delay(1000).ContinueWith(task => ShowText("Кола", stopwatch.Elapsed));

var burgerTask = Task.Delay(5000).ContinueWith(task => ShowText("Бургер", stopwatch.Elapsed));

await Task.WhenAll(friesTask, drinkTask, burgerTask);

ShowText("Заказ", stopwatch.Elapsed);

stopwatch.Stop();

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П18");

}

static void ShowText(string name, TimeSpan time)

{

Console.WriteLine($"{name} Приготувався(-лась) за {time}");

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

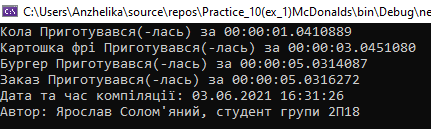


Рис 4. Результат роботи 2 пункту завдання 1.

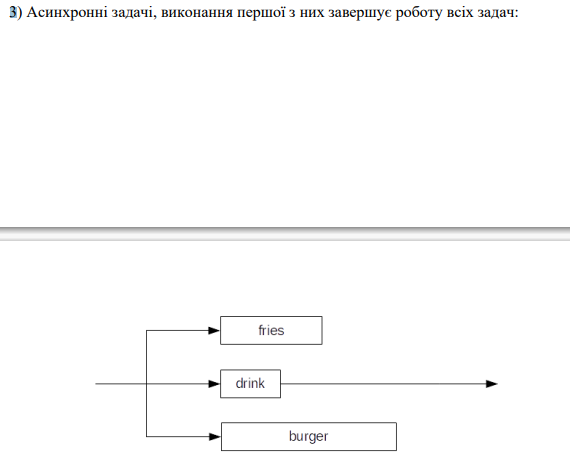


Рис 5. Підпункт 3 завдання 1.

Лістинг 3. Код для виконання підпункту 3

using System;

using System.Diagnostics;

using System.Threading;

using System.Threading.Tasks;

using System.Text;

using System.Globalization;

namespace Practice\_10\_ex\_1\_McDonalds

{

class Program

{

static void Main(string[] args)

{

OrderAsync();

Console.ReadLine();

}

static async void OrderAsync()

{

Console.OutputEncoding = Encoding.UTF8;

var stopwatch = Stopwatch.StartNew();

var friesTask = Task.Delay(3000).ContinueWith(task => ShowText("Картошка фрі", stopwatch.Elapsed));

var drinkTask = Task.Delay(1000).ContinueWith(task => ShowText("Кола", stopwatch.Elapsed));

var burgerTask = Task.Delay(5000).ContinueWith(task => ShowText("Бургер", stopwatch.Elapsed));

await Task.WhenAny(friesTask, drinkTask, burgerTask);

ShowText("Заказ", stopwatch.Elapsed);

stopwatch.Stop();

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П18");

}

static void ShowText(string name, TimeSpan time)

{

Console.WriteLine($"{name} Приготувався(-лась) за {time}");

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

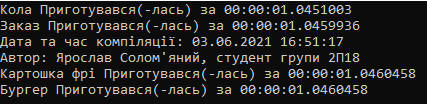


Рис 6. Результат виконання підпункту 3



Рис 7. Підпункт 4 завдання 1

Лістинг 4. Виконання підпункту 4 завдання 1

using System;

using System.Diagnostics;

using System.Threading;

using System.Threading.Tasks;

using System.Text;

using System.Globalization;

using System.Collections.Generic;

namespace Practice\_10\_ex\_1\_McDonalds

{

class Program

{

static void Main(string[] args)

{

OrderAsync();

Console.ReadLine();

}

static async void OrderAsync()

{

Console.OutputEncoding = Encoding.UTF8;

var stopwatch = Stopwatch.StartNew();

var friesTask = Task.Delay(3000).ContinueWith(task => ShowText("Картошка фрі", stopwatch.Elapsed));

var drinkTask = Task.Delay(1000).ContinueWith(task => ShowText("Кола", stopwatch.Elapsed));

var burgerTask = Task.Delay(5000).ContinueWith(task => ShowText("Бургер", stopwatch.Elapsed));

var tasks = new List<Task>() { friesTask, drinkTask, burgerTask };

while (tasks.Count > 0)

{

var task = await Task.WhenAny(tasks);

tasks.Remove(task);

Console.WriteLine($"Клієнт спожив продукт. Залишилось приготувати продуктів: {tasks.Count} ");

}

ShowText("Заказ", stopwatch.Elapsed);

stopwatch.Stop();

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П-18");

}

static void ShowText(string name, TimeSpan time)

{

Console.WriteLine($"{name} Приготувався(-лась) за {time}");

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

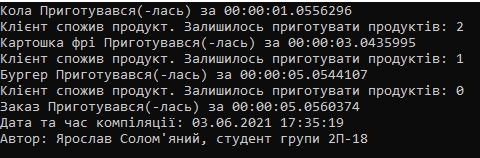


Рис 7. Результат виконання підпункту 4 завдання 1.

Завдання 4. Клас HttpClient – базовий клас для надсилання HTTP-запитів та отримання HTTP-відгуків від ресурсу, визначеного за допомогою URI. Код HTTP-відгуку визначає, чи було успішно виконано конкретний HTTP-запит. Помилки ВЕБ-сервера діляться на діапазони: 100-199 – Інформаційний. Повідомляє агенту що, запит агента прийнятий і обробляється. 200-299 – Запит агента оброблений успішно, сервер відправив клієнтові запитуваний документ. 300-399 – Запит агента змінений і агентові потрібно почати деякі дії для задоволення зміненого запиту. 400-499 – Проблеми при спробі виконати запит. 500-599 – Помилки сервера. Організуйте GET- та POST-запити до певного невеличкого сайту з мережі Інтернет на зразок http://example.com/ або http://webcode.me. − GET-запит має отримувати та виводити в консоль HTML-код веб-сторінки. Для цього стане в нагоді метод GetStringAsync() класу HttpClient. − POST-запит буде здійснюватись на сайт https://httpbin.org/post, який є тестовий онлайн-сервісом для розробників. Створіть об’єкт власного DTO-класу (унікальний для кожного із варіантів), серіалізуйте його дані в json-файл та асинхронно надішліть на вище вказаний сервер. Отримайте результати запиту та виведіть їх у консоль додатку.

Лістинг 5. Код виконання завдання №4

using Newtonsoft.Json;

using System;

using System.Globalization;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

namespace Ex\_4

{

class Program

{

static async Task Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

//GET

using var site = new HttpClient();

var content = await site.GetStringAsync("http://example.com/");

Console.WriteLine(content);

//POST

var person = new Person("Yaroslav\_Solomianiy", "Student");

var json = JsonConvert.SerializeObject(person);

var data = new StringContent(json, Encoding.UTF8, "application/json");

var url = "https://httpbin.org/post";

var response = await site.PostAsync(url, data);

string result = response.Content.ReadAsStringAsync().Result;

Console.WriteLine(result);

//TimeStamp

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П-18");

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

internal class Person

{

private string v1;

private string v2;

public Person(string v1, string v2)

{

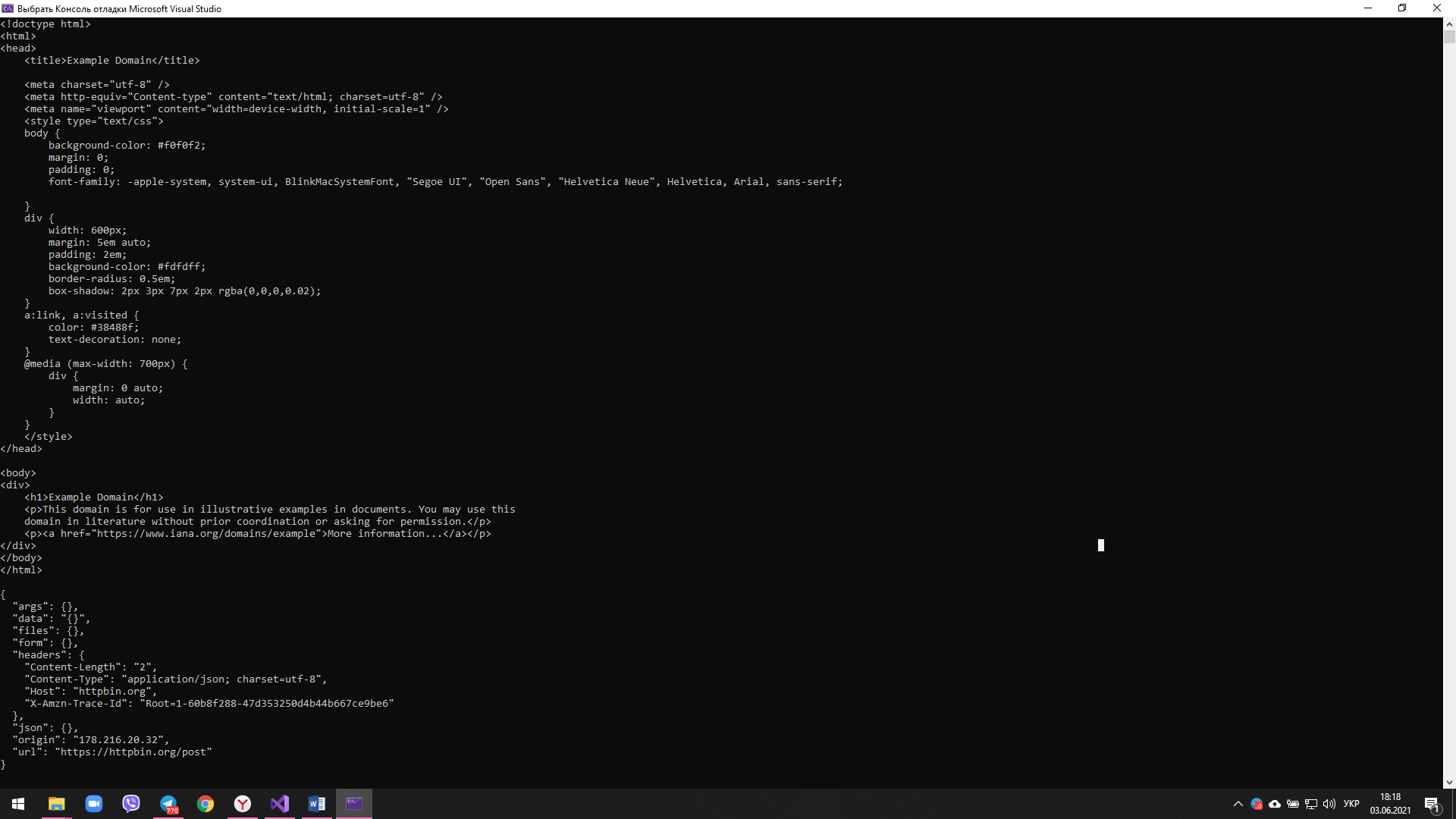
this.v1 = v1;

this.v2 = v2;

}

}

}



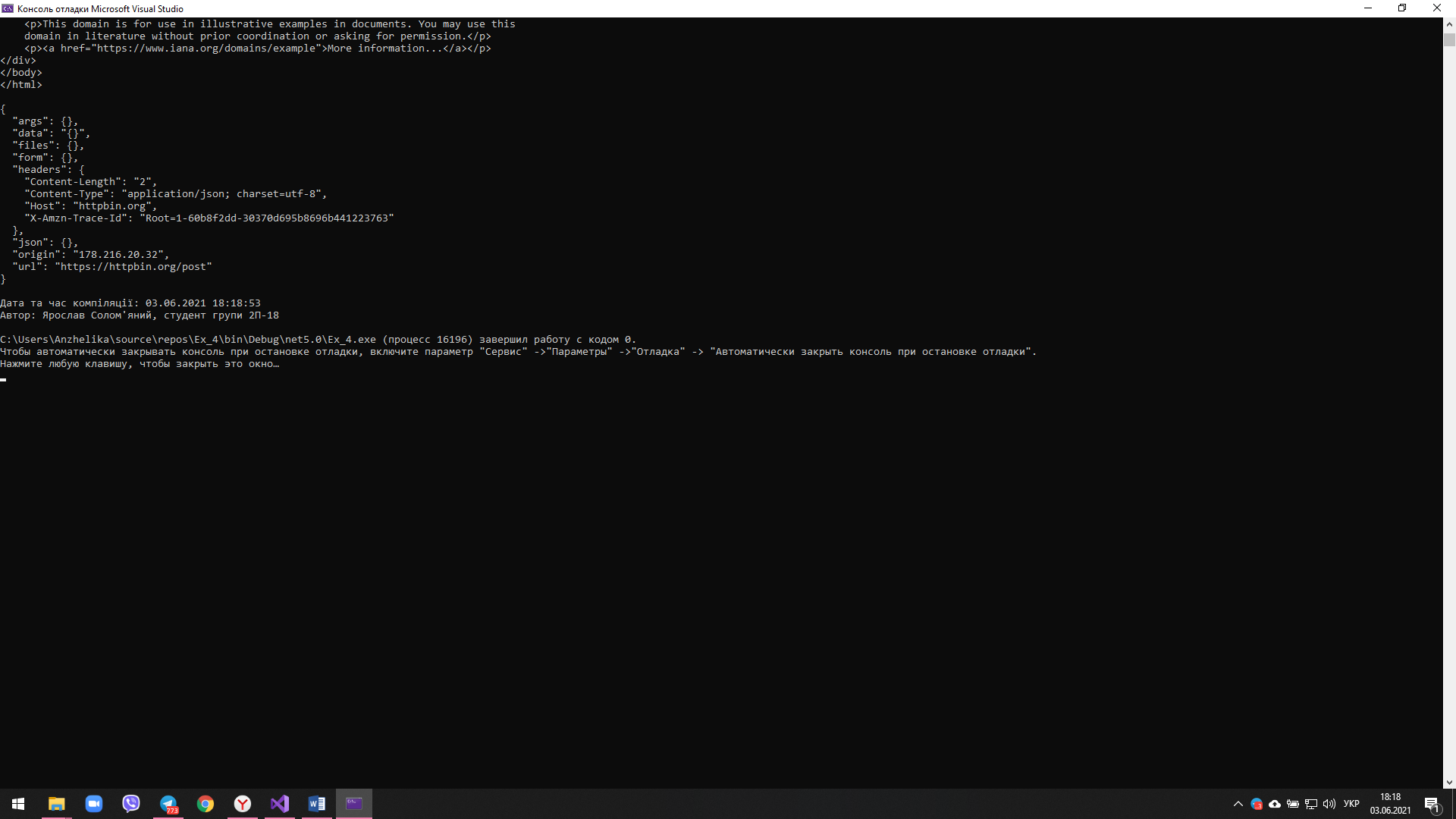


Рис 8,9. Результат виконання запитів

Завдання 6. Розгляньте та детально опишіть у звіті туторіал з обробки кількох асинхронних задач по мірі їх завершення. Продемонструйте роботу додатку в звіті.

За допомогою Task.WhenAny можна запускати кілька завдань одночасно і обробляти їх по одній у міру завершення, а не в порядку їх запуску.

Лістинг 6.WhenAny Асинхронні задачі по мірі їх завершення

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Globalization;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

namespace Ex\_6

{

class Program

{

static Task Main() => SumPageSizesAsync();

static async Task<int> ProcessUrlAsync(string url, HttpClient client)

{

byte[] content = await client.GetByteArrayAsync(url);

Console.WriteLine($"{url,-60} {content.Length,10:#,#}");

return content.Length;

}

static async Task SumPageSizesAsync()

{

var stopwatch = Stopwatch.StartNew();

IEnumerable<Task<int>> downloadTasksQuery =

from url in s\_urlList

select ProcessUrlAsync(url, s\_client);

List<Task<int>> downloadTasks = downloadTasksQuery.ToList();

int total = 0;

while (downloadTasks.Any())

{

Task<int> finishedTask = await Task.WhenAny(downloadTasks);

downloadTasks.Remove(finishedTask);

total += await finishedTask;

}

stopwatch.Stop();

Console.WriteLine($"\nTotal bytes returned: {total:#,#}");

Console.WriteLine($"Elapsed time: {stopwatch.Elapsed}\n");

//TimeStamp

PrintTimeStamp("Ярослав Солом'яний, студент групи 2П-18");

}

static readonly HttpClient s\_client = new HttpClient

{

MaxResponseContentBufferSize = 1\_000\_000

};

static readonly IEnumerable<string> s\_urlList = new string[]

{

"https://docs.microsoft.com",

"https://docs.microsoft.com/aspnet/core",

"https://docs.microsoft.com/azure",

"https://docs.microsoft.com/azure/devops",

"https://docs.microsoft.com/dotnet",

"https://docs.microsoft.com/dynamics365",

"https://docs.microsoft.com/education",

"https://docs.microsoft.com/enterprise-mobility-security",

"https://docs.microsoft.com/gaming",

"https://docs.microsoft.com/graph",

"https://docs.microsoft.com/microsoft-365",

"https://docs.microsoft.com/office",

"https://docs.microsoft.com/powershell",

"https://docs.microsoft.com/sql",

"https://docs.microsoft.com/surface",

"https://docs.microsoft.com/system-center",

"https://docs.microsoft.com/visualstudio",

"https://docs.microsoft.com/windows",

"https://docs.microsoft.com/xamarin"

};

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

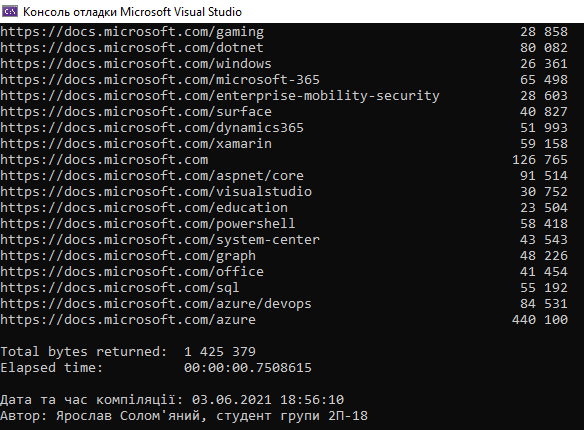


Рис 10.Результат роботи програми

У наступному прикладі використовується запит для створення колекції завдань. Кожне завдання завантажує вміст зазначеного веб-сайту. У кожній ітерації циклу while очікуваний виклик WhenAny повертає завдання з колекції завдань, яка першою завершує свою завантаження. Це завдання видаляється з колекції і обробляється. Цикл виконується до тих пір, поки в колекції ще є завдання.

Метод починається зі створення екземпляра і запуску Stopwatch. Потім він включає запит, який при виконанні створює колекцію завдань. Кожен виклик ProcessUrlAsync в наступному коді повертає Task <TResult>, де TResult - ціле число.

Через відкладеного виконання за допомогою LINQ для запуску кожного завдання викликається Enumerable.ToList.

Чекає finishedTask, що повертається при виклику ProcessUrlAsync. Мінлива finishedTask є Task <TResult>, де TResult - ціле число. Завдання вже завершенt, але вона очікується для отримання розміру завантаженого веб-сайту, як показано в наступному прикладі. У разі збою завдання await видасть перший виняток дочірнього елемента, що зберігається в AggregateException, на відміну від зчитування властивості Task <TResult> .Result, яке видасть AggregateException.

Завдання 2. Оператори async-await використовуються для асинхронного запуску методу. Напишіть програму, яка розпочинає тривалий метод зі зчитуванням інформації з великого файлу. Цей метод повинен відображати повідомленнястатус та запускати асинхронне його зчитування. Протягом цього тривалого зчитування продемонструйте можливість виконання інших операцій, зокрема вводу тексту та його елементарну обробку.

Лістинг 7. Код для виконання завдання №2.

using System;

using System.Globalization;

using System.IO;

using System.Threading.Tasks;

namespace Ex\_3\_Practice10

{

class Program

{

async static Task Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

await FileNotMethod();

}

async static Task TaskMethod()

{

await Task.Run(() =>

{

if (File.Exists(@"D:\1.txt") == true)

{

Console.WriteLine("1.txt-файл Існує");

while (File.Exists(@"D:\1.txt"))

{

}

}});

}

static async Task FileNotMethod()

{

Task t1 = Task.Run(() => TaskMethod());

Task t2 = Task.Run(() =>

{

Console.WriteLine("Введіть число:");

int n = Int32.Parse(Console.ReadLine());

Console.WriteLine($"Квадрат числа = {n \* n}");

PrintTimeStamp("Солом'яний Ярослав, студент групи 2П-18");});

await Task.WhenAll(new[] { t1, t2 });

} public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

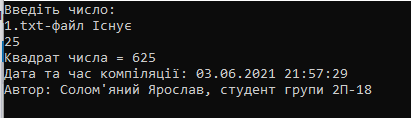


Рис 11. Результат роботи програми

Завдання 3.створіть програму, яка дозволятиме асинхронно записувати в текстовий файл 5000 рядків тексту. Для демонстрації асинхронної роботи додатку також запускайте інший, синхронний метод, який буде виводити на екран повідомлення про своє виконання одночасно з асинхронним записом.

Лістинг 9. Код для виконання завдання

using System;

using System.Globalization;

using System.IO;

using System.Threading.Tasks;

namespace Ex\_3\_Practice\_10

{

class Program

{

async static Task Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

await AsyncgMethod();

}

async static Task TaskMethod()

{

await Task.Run(() =>

{

if (File.Exists(@"D:\1.txt") == true)

{

Console.WriteLine("Початок зчитування файлу 1.txt");

while (File.Exists(@"D:\1.txt"))

{

}

}

}); ;

}

public static async Task AsyncgMethod()

{

Task t1 = Task.Run(() => TaskMethod());

Task t2 = Task.Run(() =>

{

string writePath = @"D:\1.txt";

try

{

Console.WriteLine("Ви можете ввести текст, але ваш ліміт 5 тисяч символів.");

int A = 1;

while (A < 5000)

{

string text = Console.ReadLine();

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLineAsync(text);

}

Console.WriteLine($"Запись виконана {A++} раз");

}

Console.WriteLine("Запись закінчена");

PrintTimeStamp("Солом'яний Ярослав, Студент групи 2П-18");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

});

await Task.WhenAll(new[] { t1, t2 });

}

public static void PrintTimeStamp(string name)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

DateTime localDate = DateTime.Now;

var culture = new CultureInfo("ru-RU");

Console.WriteLine("Дата та час компіляції: {0}", localDate.ToString(culture));

Console.WriteLine("Автор: {0}", name);

}

}

}

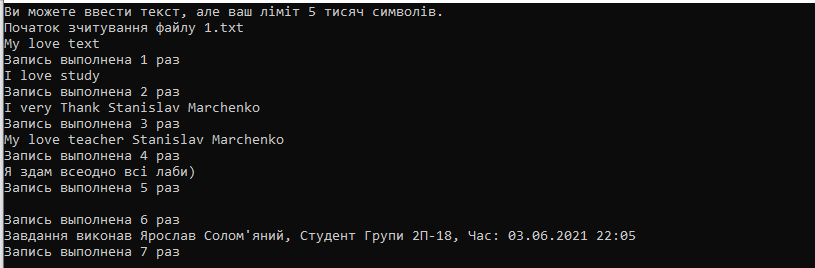


Рис 12.Результат роботи програми