**Міністерство освіти і науки України**

**Національний технічний університет України**

**“Київський політехнічний інститут ІМ. І. СІКОРСЬКОГО”**

ННК «ІПСА» НТУУ “Київський Політехнічний Інститут”

(назва факультету, інституту)

Системного проектування

(назва кафедри)

**Пояснювальна записка**

до курсової роботи на тему: “Паралельний парсинг профайлів кіберспортсменів з веб-ресурсу liquipedia.net ”

**Студент групи** ДА-61 Калюжний Єгор Юрійович \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр групи) (прізвище, ім’я, по батькові) (підпис)

**Керівник проекту** *ас. Яременко В. С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(вчені ступінь та звання, прізвище, ініціали) (підпис)

|  |  |
| --- | --- |
| **Керівник:** | **Виконавець:** |
| ***Яременко В. С.*** | **ст . Калюжний Єгор**  **гр. ДА-61** |
| **Допущений до захисту** | **Зал. книжка** |
| **Захищено із оцінкою** |  |

Київ – 2019

Форма № У-6.01

Затв. наказом УРСР

від 3 серпня 1984р. № 253

ННК «ІПСА» НТУУ “Київський Політехнічний Інститут ім. І. Сікорського

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва вищого навчального закладу)

Системного проектування

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Технології розподілених систем і паралельних обчислень

Дисципліна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6.050101

Спеціальність\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6

3

ДА - 61

Курс\_\_\_\_\_\_\_\_\_\_Група\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Семестр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

Калюжний Єгор Юрійович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема курсової роботи

Паралельний парсинг профайлів кіберспортсменів з веб-ресурсу liquipedia.net \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Строк здачі студентом закінченого проекту (роботи)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Калюжний Єгор Юрійович

(підпис)

(підпис)

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Яременко В. С.*

« \_\_ » \_ \_ р.

**Зміст**

[**Завдання** 5](#_Toc8421799)

[**Теоретичні відомості** 7](#_Toc8421800)

[**Багатопотоковість** 7](#_Toc8421801)

[**Парсинг** 9](#_Toc8421802)

[**Постановка задачі** 10](#_Toc8421803)

[**Виконання задачі** 13](#_Toc8421804)

[**Модулі програми** 15](#_Toc8421805)

[**Висновки** 16](#_Toc8421806)

[**Додаток. Лістинг програми** 18](#_Toc8421807)

**Завдання**

**Варіант 8. Була запропонована власна тема.**

Парсинг профайлів кіберспортсменів з веб-ресурсу liquipedia.net . А саме ім’я лише латинськими літерами, нікнейму гравця, дата народження(якщо наявна), країна походження, роль у команді та скільки заробив за свою професійну кар’єру гравець. Парсинг має мати можливість паралельного виконання з налаштуванням користувача.

**Теоретичні відомості**

**Багатопотоковість**

**Багатопотоковість** — властивість операційної системи або застосунку, яка полягає в тому, що процес, породжений в операційній системі, може складатися з кількох нитей, що виконуються паралельно, або навіть одночасно на багатопроцесорних системах. При виконанні деяких завдань таке розділення може досягти ефективнішого використання ресурсів комп'ютера. Такі нитки виконання ще називають потоками.

Суттю багатонитковості є квазі-багатозадачність на рівні одного виконуваного процесу, тобто всі нитки виконуються в адресному просторі процесу. Окрім цього, всі нитки процесу мають не тільки спільний адресний простір, але і спільні дескриптори файлів. Процес, що виконується, має як мінімум одну (головну) нитку.

Багатонитковість (як доктрину програмування) не слід плутати ані з багатозадачністю, ані з багато-процесорністю, не зважаючи на те, що операційні системи, що реалізовують багатозадачність, як правило, реалізують і багатонитковість.

Переваги в багатонитковості такі:

* Спрощення програми в деяких випадках, за рахунок використання загального адресного простору
* Менші відносно процесу часові витрати на створення ниті і взаємодію між нитями
* Підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій вводу/виводу

**Розпарале́лювання програ́м** — процес адаптації алгоритмів, записаних у вигляді програм, для їх ефективного виконання на обчислювальній системі паралельної архітектури (останнім часом, як правило, на багатопроцесорній обчислювальній системі). Полягає або в переписуванні програм на спеціальну мову, яка описує паралелізм і яку розуміють транслятори цільової обчислювальної системи, або до вставки спеціальної розмітки (наприклад, інструкцій MPI або OpenMP).

Розпаралелювання може бути ручним, автоматизованим або напівавтоматизованим.

При розпаралелюванні важливо брати до уваги не лише формальний паралелізм структури алгоритму, але й те, що обмінні операції в паралельних комп'ютерах є, як правило, набагато повільнішими від арифметичних. З цим пов'язане існування левової частки накладних витрат на організацію паралелізму.

**Parallel.For Method**

Простір імен: System.Threading.Tasks

Assemblies: System.Threading.Tasks.Parallel.dll, mscorlib.dll, netstandard.dll

Виконує цикл for, в якому ітерації можуть виконуватися паралельно.

Метод Parallel.For дозволяє виконувати ітерації циклу паралельно. Він має таке визначення: For (int, int, Action <int>), де перший параметр задає початковий індекс елемента в циклі, а другий параметр - кінцевий індекс. Третій параметр - делегат Action - вказує на метод, який буде виконуватися один раз за ітерацію:

**Thread Class**

Простір імен: System.Threading

Assemblies: System.Threading.Thread.dll, mscorlib.dll, netstandard.dll

Основний функціонал для використання потоків в додатку зосереджений в просторі імен System.Threading. У ньому визначено клас, що представляє окремий потік - клас Thread.

Клас Thread визначає ряд методів і властивостей, які дозволяють управляти потоком і отримувати інформацію про нього.

**Парсинг**

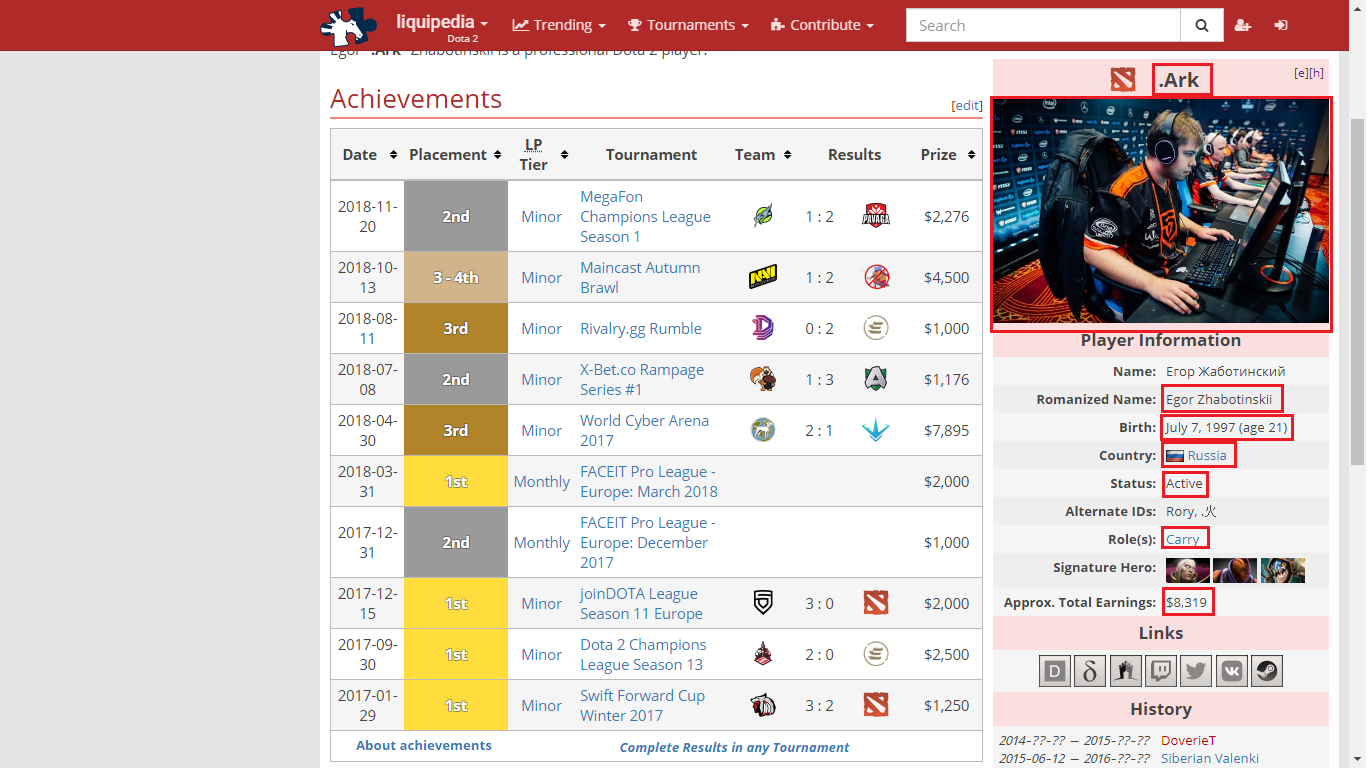
**Парсинг(від англ. Parse)** – процес аналізу або розбору певного контенту на складові за допомогою роботів-парсерів (спеціальних програм або скриптів). У SEO цим контентом є html-код сторінок сайтів. це процес збору деяких даних і складання з них бази. Наприклад, можна зібрати базу гостьових книг. Або базу каталогів сайтів. Ось навіщо це робити і робити взагалі — кожен вирішує сам. Найчастіше такі бази використовуються спамерами. Але не обов’язково.

Найвідоміші парсери в мережі – це пошукові роботи, які аналізують сторінки, зберігають дані аналізу у себе в базі і потім при пошуку видають релевантні та актуальні документи.

Часто парсинг плутають з граббінгом. Це близькі поняття, але все ж мають різні значення. Граббер дозволяє скачувати інформацію з мережі (html-сторінки, rss-стрічки, xml-документи) в свою базу, а парсер дозволяє виявити з цієї купи корисну інформацію і обробити її, залежно від поставлених завдань. Часто вебресурси обмежеють кількість запитів з одної адреси тоді раціональним є використання **Проксі-серверу.**

**Проксі-сервер** (від англ. proxy — «представник, уповноважений») — сервер (комп'ютерна система або програма) в комп'ютерних мережах, що дозволяє клієнтам виконувати непрямі (через посередництво проксі-сервера) запити до мережевих сервісів. Спочатку клієнт з'єднується з проксі-сервером і запитує який-небудь ресурс (наприклад, e-mail), розташований на іншому сервері. Потім проксі-сервер або підключається до вказаного сервера і отримує ресурс у нього, або повертає ресурс з власного кешу (у випадках, якщо проксі має свій кеш). У деяких випадках запит клієнта або відповідь сервера може бути змінена проксі-сервером з певною метою. Також проксі-сервер дозволяє захищати клієнтський комп'ютер від деяких мережевих атак і допомагає зберігати анонімність клієнта.

**Постановка задачі**

Сайті liquipedia.net є найбільшим інформаційним ресурсом у світі кіберспорту. На ньому міститься детальна інформація про професійних гравців їх профайли. Ресурс не має відкритого API аби отримати профайли у формі зручної для машинної обробки. Необхідно спарсити необхідні данні з профайлів та зберегти їх у зручному форматі для подальшого заповнення ними бази даних та аналізу. 

Так виглядає сторінка гравця на сайті, маркером виділені поля, щл необхідно зберегти. Парсинг html-сторінок громіздкий та ресурсоємний процес адже для проходу по DOM дереву зазвичай використовують **регулярні вирази.** Це конструкція, що задає принципи пошуку інформції та обробки символьних рядків. Також існує звтрата часу на те щоб підєднатися до віддаленного ресурсу по протоколу htttp, а також повністю завантажити сторінку (**liquipedia** використовує спеціальний синтаксис для розмітки сторінки користувачами, аби дати можливість кожному створити та редагувати інформацію у відкритій енциклопедії, це підвищує час рендеру сторінки).

**Проблема.** Ресурс слідкує за навантаженням і блокує IP адресса у яких перевищенно лымыт запитів за проміжок часу. Якщо наш парсер не буде спроможний за раз спарсити велику кількість данних, в ньому стає менше сенсу.  
Для вирішення цієї проблеми будемо використовувати безкоштовні **Проксі-Сервери**, зауважимо що нам необхідні проксі з підтримкою **https**(SSL). Подивимся як використання проксі-серверу може викликати пониження продуктивності парсеру

Таким чином час на виконання запиту через проксі-сервер дорівнюватиме грубо кажучи (b+a), що довше аніж напряму межуючи проксі-сервер a+b>c

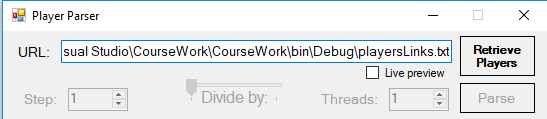
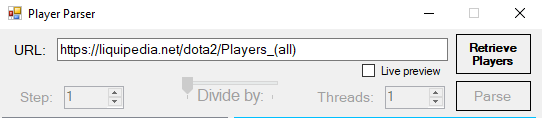
Підсумовуючи можемо побачити, що парсинг займає величезну кількість часу, для обробки однієї сторінки. Це підштовхує нас на використання технологій паралельних обчислень, для збереження часу на виконання задачі в цілому.

Задача полягає в тому, аби маючи список посилань на профайли гравців, розділити його між потоками для одночасної(паралельної обробки).

**Виконання задачі**

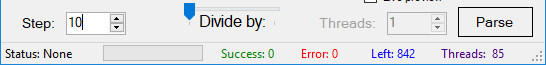
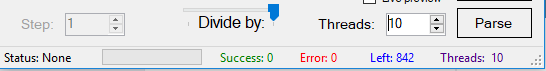
**Пункт 1.** Отримання списку посилань на профайли гравців.

Додаток має два варіанта постачання посилань.

* Текстовий файл зї списком посилань .
* Посилання на таблицю профайлів на **liquipedia .**

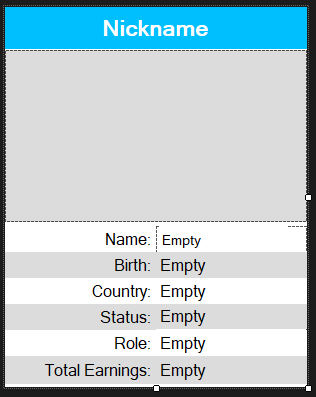
**Пункт 2.** Налаштування алгоритму розподілення на потоки

Додаток має спеціальний інтерфейс для вибору методу та введення додаткових параметрів.

* Режим кроку. Користувач задає величину кроку для розділення списку між потоками, кількість потоків рахується автоматично.
* Режим поткоів. Користувач явно задає кількість потоків для виконання задачі, розподіл списку посилань відбувається автоматично. 

**Пункт 3.** Кастомний елемент відображення інформації кіберспортсмена

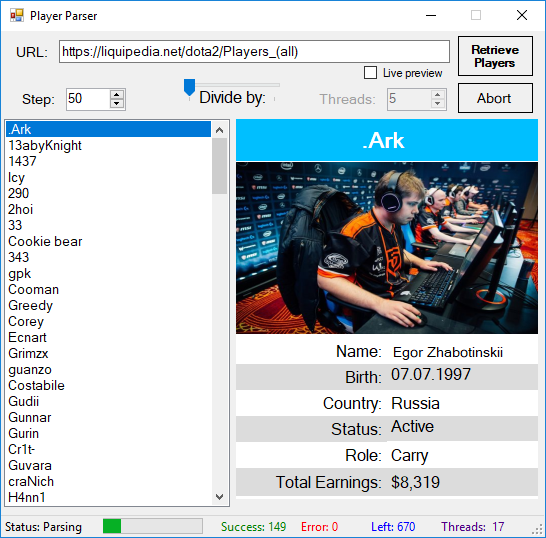
C# .NET Дозволяє розробнику створювати власні елементи керування інкапсулючи в них логіку відображення та введення інформації, що зручно при написанні великих проектів, а також подальше перевикористання користувацького елементу.



Так виглядає кастомний елемент для відображення інформації про гравця на стадії проектування(розробки)

**Пункт 4**. Парсинг профайлів

Парсинг працює у фонових потоках тому не заважає користуватися графічним інтерфейсом. Переглядати уже відпрацьовані профайли можна одразу у додатку. В будь-який час виконання парсинг можна зупинити натиснувши “Abort”. Справа знаходиться список-навігація по вже відпрацьованим сторінкам.



Стан парсеру та супроводжуючу інформацію можна спостерігати у статусному рядку.

Існує функція «Лайв превью», містить розважальний характер, демонструє активне заповнення профайлів гравців .

**Модулі програми**

* **Form1 –** Описує форму(графічний інтерфейс) користувача, графічно зображує роботу програми виконує базові перевірки вхідних даних та встановлює обмеження параметрів налаштувань.
* **PlayerView** – Спеціальний засіб, що дозволяє створювати власні елементи керування, завдяки нього створено елемент що відображає інформацію про одного конкретного гравця, інкапсулюючи таким чином логіку відображення інформації.
* **Player –** Клас описує сутність гравця з усіма необхідними властивостями для серіалізації .
* **ParserHelper** – Клас реалізує функціонал зберігання опрацьованих профайлів та їх серіалізацію. Отримання списку посилань с веб-таблиці, розпаралелення списку посилань.
* **Parser** – Реалізація логіки обробки сторінки профайлу, робота з бібліотекою для парсингу.

**Висновки**

У ході даної курсової роботи було створено додаток для отримання(парсингу) даних про кіберспортсменів, а саме їх профайли на сайті енциклопедії [https://liquipedia.net](https://liquipedia.net/). Використали декілька засобів для розпаралелення задач, що дало можливість опрацьовувати декілька сторінок одночасно, а в результаті скорити час виконання задачі. Також додаток має користувацький інтерфейс для взаємодії, відображення інформації про гравців, вибору метод за яким буде вирахувано кількість потоків для обробки та введення додаткових параметрів. Існує можливість збереження результатів у файлі у форматі JSON. Паралельне виконання дійсно дає значний приріст, але в той самий час виникають проблеми зі спільним доступом до ресурсів. А в нашому випадку проблемою стала ще й необхідність використати проксі на кожному з потоків. Використовуючи безкоштовні проксі-сервери, не маємо гарантії що всі вони будуть у робочому стані, та матимуть більш-менш непоганий час відгуку.

**СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ**

1. Турчак Л.И. Основы численных методов. - М.: Наука, 1987. - 320 с.
2. Петренко А.І. Обчислювальна математика. Суми.: ВМУРоЛ “Україна”,
3. Лекції по технології паралельного обчислення і росподільчих систем КПІ Яременко В. С.

Посилання на GitHub: <https://github.com/Camamber/PlayerParser>

**Додаток. Лістинг програми**