**ЗМІСТ**

Вcтуп……………………………………………………………………………...5

1…………………………………………………………………………………...7

2…………………………………………………………………………………...8

3………………………………………………………………………………….10

3.1……………………………………………………………………………10

3.2…………………………………………………………………………....11

3.3……………..………………………………………...…………..…….....19

3.4...……………..…………..…………………………………………..…...23

4. ……………………………………………….………………………………..24

Висновки ………………………………………………………………………..27

Список використаних джерел………………………………………………….28

Додаток 1……………………...………………………………………………...30

Додаток 2…………………...…………………………………………………...32

Додаток 3……………………...………………………………………………...40

**ВСТУП**

Актуальність: досліджувана тема актуальна, тому що написання характеристик необхідне у сучасному світі через серйозне практичне значення. Постає проблема для студентів, класних керівників та деканатів в наданні оцінки ділових і моральних якостей, щоб кожну особу можна було охарактеризувати відповідно до її особливостей роботи й характеру. Є багато способів вирішення цієї проблеми, але автоматизація написання цього документа методами програмування вирішує її, суттєво полегшує даний процес, виключає неправильність самостійного складання, дозволяє зекономити час та є корисною у повсякденному житті.

Мета: вивчення та аналіз літературних джерел інформації, дослідження методів створення програмного продукту для автоматичного написання характеристик, на основі здобутих навичок програмування мовою С++ розроблення програмної реалізації алгоритмів, вивчення особливостей характеристики як документу, закріплення теоретичних знань та практичних навичок програмування, набутих при вивченні дисципліни “Основи програмування”.

Завдання: розробити програму, яка буде автоматично формувати текст на основі розробленого шаблону та за допомогою введених даних користувачем про особу, на яку писатиметься документ(ім’я, вік, стать, захоплення, особисті якості), та писати характеристику, створити зручний інтуїтивно зрозумілий користувацький інтерфейс.

Об’єктом дослідження курсової роботи є програмне забезпечення для автоматичного написання характеристик.

Предмет дослідження – особливості реалізації алгоритму написання програмного продукту для автоматизації написання документу.

Методи дослідження:

- пошук – збір інформації, вивчення основних джерел інформації, складових частин автоматизації характеристики;

- аналіз – перевірка знайденої інформації, кількісний та якісний аналіз даних, їх систематизація;

- порівняння – встановлення відмінностей між різними алгоритмами реалізації програми та знаходження спільних рис, вибір найбільш підходящого;

- рефлексія – дослідження конкретних явищ;

- експеримент – виконання дій для перевірки гіпотези, правильності алгоритму;

- формалізація – представлення об’єктів шляхом відображення їхнього змісту в знаковій формі, наприклад, у вигляді діаграми або таблиці.

// 2 сторінки

**РОЗДІЛ 1. АНАЛІЗ ЗАВДАННЯ**

Навчаючись або працюючи у школах, вищих навчальних закладах, багатьом доводилось писати характеристику — часто використовуваний документ, у якому в офіційній формі висловлено громадську думку про працівника, учня, студента як члена колективу, про його громадську або трудову діяльність, і який складається на вимогу або письмовий запит іншої установи для подання до цієї установи.

**1.1. Теоретична складова**

Ця програма спрямована на полегшення процесу написання характеристик, тому вона є актуальною для будь-якого віку й часу, адже написання вручну є досить важкою справою. Як і для будь-якого програмного продукту, необхідний чіткий алгоритм реалізації створення програмного забезпечення.

Характеристика вважається офіційним документом, адже надається однією певною установою іншій, тому стиль тексту обов’язково має бути офіційно-діловим. Також даний документ має бути об’єктивним, адже фіксує як і позитивні, так і негативні аспекти діяльності та риси характеру

Програмний застосунок повинен відповідати таким функціональним вимогам:

- користувач може обирати вид необхідної характеристики;

- при введенні даних про особу програма повинна перевірити їх та дати можливість повторного введення даних у разі неправильного вводу;

- базою даних слугує файл, в який записуються вхідні дані;

- за бажанням користувача має бути можливість створити нову папку для збереження файлу;

- на основі введених даних має бути сформована готова характеристика та записана у файл;

- в кінці документа обов’язково має бути дата написання та місце для підпису уповноваженої особи.

Нефункціональні вимоги до продукту:

- інтерфейс програми має бути інтуїтивно зрозумілим та зручним для всіх користувачів;

- програма повинна бути доступною з будь-якої операційної системи (Windows, MacOS) та мати коректне відображення в ній.

**1.2. Аналіз вхідних/вихідних даних**

Проаналізувавши літературні джерела та дослідивши приклади написання характеристики, можна уточнити завдання проектування та визначити основні вхідні дані.

Для написання характеристики необхідно вказувати такі основні реквізити про особу:

* прізвище, ім’я, по батькові того, кому видається характеристика;
* стать;
* рік народження;
* особисті якості людини;
* місце в колективі;
* захоплення.

Реквізитами для характеристики є:

* рейтинг характеристики(позитивна, негативна, наполовину позитивна);
* мова написання;
* призначення(куди пишеться).

Вище перелічені реквізити виступають вхідними даними у програмі. Комунікація з користувачем здійснюється за допомогою консолі. Програма повинна друкувати запрошення на ввід цих даних користувачем, тому введення відбувається у зручному та зрозумілому форматі, тому через простоту інтерфейсу користувач буде тратити дуже мало часу на розуміння. Має відбуватися зчитування вхідних даних, перевірка на правильність, а у випадку неправильного вводу знову дати можливість користувачеві ввести їх, а потім записати перевірені дані у файл.

Вихідними даними є сформована програмою характеристика, яка також записується у файл для зручності у використанні користувачем. Необхідною умовою є виведення дати створення характеристики.

**1.3. Вибір мови програмування та системи розробки**

Для роботи над алгоритмами та безпосереднього написання коду необхідно було обрати підходящу мову(C/C++) та середовище програмування. Було прийняте рішення обрати об’єктно-орієнтовану мову С++, а середовище розробки — Visual Studio. Для колективної розробки було використано веб-сервіс GitHub.

С++ — мова програмування високого рівня, що підтримує декілька парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Була розроблена у 1979 році.

Розвиток мови С ++ відбувався на базі мови С, і, за невеликим винятком, мова С була збережена як підмножина C ++. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм.

Серед основних причин вибору мови С++ є можливості об’єктно-орієнтованого програмування, можливе завдяки трьом основним поняттям на яких базується саме ООП:

1. Інкапсуляція. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкта, замість цього надаючи інтерфейс взаємодії з ним.

2. Успадкування. Це означає, що об'єкти (класи) можуть переймати деякі властивості у своїх прабатьків. Це призводить до повторного використання вже написаного одного разу коду. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюється ієрархія з класів, де від батьківського класу походять усі інші дочірні класи.

- Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по різному на однакові повідомлення. Це спричинене зміною в одного з класів якогось методу, шляхом запису іншого алгоритму.

Функція незалежності платформи C ++ дозволяє користувачеві легко запускати ту саму програму, в незалежності від виду операційних системах.

В якості середовища програмування було обране Visual Studio. Це інтегроване середовище розробки представляє собою багатофункціональну програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. У ньому міститься не тільки редактор і відладчик, а й компілятор, засоби автозавершення коду та велика кількість інших функцій для спрощення процесу розробки. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду, додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення.

Враховуючи всі переваги мови програмування С++ та середовища розробки Visual Studio, можна зробити висновок, що цей вибір є найоптимальнішим рішенням для створення програми за завданням курсової роботи.

**1.4 Вибір хмарного сервісу для колективної розробки**

Для виконання спільної роботи з розробки проекту та відслідковування змін у файлах необхідно було обрати один із хмарних сервісів контролю версій. Було прийнято рішення використовувати найпопулярнішу систему контролю версій GIT та сервіс онлайн-хостингу проектів Github.

Git - розподілена система контролю версій, яка дає можливість розробникам відстежувати зміни в файлах і працювати над одним проектом спільно з колегами. Вона була розроблена в 2005 році. Git відомий своєю швидкістю, простим дизайном, підтримкою нелінійної розробки, повної децентралізацією і можливістю ефективно працювати з великими проектами.

Переваги Git:

1. Безкоштовний і open-source.

Можна безкоштовно завантажити і вносити будь-які зміни у вихідний код.

1. Невеликий і швидкий.

Виконує всі операції локально, що збільшує його швидкість. Крім того, Git локально зберігає весь репозиторій в невеликий файл без втрати якості даних.

1. Резервне копіювання.

Git ефективний в зберіганні бекапів, тому відомо мало випадків, коли хтось втрачав дані при використанні Git.

1. Просте розгалуження.

В інших системах контролю версій створення гілок - втомлива і трудомістка задача, так як весь код копіюється в нову гілку. У Git управління гілками реалізовано набагато простіше і ефективніше.

Коли ви вперше починаєте використовувати git, він може вас налякати. Команди та концепції можуть здатися зовсім чужими, якщо в минулому ви ніколи не використовували будь-яку платформу контролю версій. Тому, давайте коротко розглянемо основні терміни Git.

Репозиторій (repository) часто згадується як репо. Репозиторій - це колекція файлів і папок, які ви використовуєте для відстеження git. Сховище складається з усієї історії змін вашої команди в проекті. Це той великий склад, на якому зберігається код, який додали ви і ваша команда.

Про фіксацію (commit) думайте як про збереження вашої роботи. Коли ви фіксуєте репозиторій, це схоже на те, що ви збираєте файли в тому вигляді, в якому вони існують в даний момент, і ставите їх в капсулу часу. Фіксація буде існувати тільки на вашому локальному комп'ютері, поки вона не буде відправлена на віддалений репозиторій.

Фіксація поміщає ваші файли в капсулу часу, а відправлення(push) - це те, що запускає капсулу в космос. Відправлення - це, по суті, синхронізація ваших збережень (фіксацій, коммітів) з хмарою (знову ж таки, ймовірно, Github). Ви також можете використовувати кілька коммітів одночасно. Ви можете працювати в автономному режимі, зробити багато роботи, а потім передати все це на Github.

Уявіть свій git-репо у вигляді дерева. Стовбур дерева, програмне забезпечення, яке запускається, називається майстер-гілкою (Master Branch). Це те, що є онлайн. Гілки цього дерева називаються, як не дивно, гілками. Це окремі екземпляри коду, який відрізняється від основної бази коду. Ви можете відгалузилося одну функцію або експериментальний патч. Розгалужуючись, ви можете зберегти цілісність основного програмного забезпечення і мати можливість відкотитися, якщо зробите щось зовсім божевільне. Це також дозволяє вам працювати над своїм завданням, не впливаючи на вашу команду (або вона на вас).

Коли гілка виправлена, не містить помилок (наскільки ви можете судити, по крайній мірі) і готова стати частиною первинної бази коду, вона буде об'єднана з головною гілкою. Об'єднання(merge) - це те, на що це схоже: злиття двох гілок. Будь-який новий або оновлений код стане офіційною частиною кодової бази. Той, хто відгалужується від точки злиття, також буде мати цей код в своїй галузі.

Github - сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій і функціональністю управління вихідним кодом - все, що підтримує Git і навіть більше. Також GitHub може похвалитися контролем доступу, багтрекінгом та управлінням завданнями. За допомогою широких можливостей Git програмісти можуть поєднувати свої [репозиторії](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D1%96%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F" \o "Репозиторій програмного забезпечення) — GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді [дерева](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%B2)).

Ознайомившись з термінологією та принципами роботи git та github, було розпочато спільну розробку. Перш за все, було створено відповідний репозиторій, та видано ролей учасників(contributors) усім розробникам. Для проекту було створено 2 гілки: master(для фінальної версії проекту) та test(для розробки). При додаванні функціоналу кожним з учасників додавалися фіксації змін(commit). Доробивши проект до кінця і протестувавши його, гілка test має об’єднатися з головною гілкою. Таким чином, при розробці програми учасники мали змогу ознайомитися з функціоналом та використати системи контролю версій Git та веб-сервіс Github.

**1.5 Вибір готових бібліотек для реалізації проекту**

Для виконання завдання необхідно було обрати готові бібліотеки мови C++ для роботи файлового вводу та виводу. Для роботи з файлами в стандартній бібліотеці визначено заголовок fstream, який визначає базові типи для читання і запису файлів. Всього є 3 основних класи для реалізації цієї задачі:

1. Ifstream – файловий ввід.
2. Ofstream – файловий вивід.
3. Fstream – файловий ввід та вивід одночасно.

Останній з класів було обрано для реалізації зчитування та запису інформації у файл, тому що він є універсальним для обох задач.

При роботі з файлом можна виділити наступні етапи:

1. Створення об'єкт класу fstream.
2. Зв'язування об'єкт класу fstream з файлом, який буде використовуватися для операцій введення-виведення.
3. Здійснення операції введення-виведення в файл.
4. Закриття файлу.

При операціях з файлом спочатку необхідно відкрити файл за допомогою функції open (). Ця функція має дві версії:

- open (шлях);

- open (шлях, режим).

Для відкриття файлу в функцію необхідно передати шлях до файлу у вигляді рядка. І також можна вказати режим відкриття. Список доступних режимів відкриття файлу:

1. ios::in: файл відкривається для введення (читання). Може бути встановлений тільки для об'єкта ifstream або fstream;
2. ios::out: файл відкривається для виводу (запису). При цьому старі дані видаляються. Може бути встановлений тільки для об'єкта ofstream або fstream;
3. ios::app: файл відкривається для дозапису. Старі дані не видаляються;
4. ios::ate: після відкриття файлу переміщує покажчик в кінець файлу;
5. ios::trunc: файл усікається при відкритті. Може бути встановлений, якщо також встановлено режим out;
6. ios::binary: файл відкривається в бінарному режимі.

Якщо при відкритті режим не вказано, то за замовчуванням для об'єктів ofstream застосовується режим ios::out, а для об'єктів ifstream - режим ios::in. Для об'єктів fstream поєднуються режими ios::out і ios::in.

У випадку розробки цього проекту в якості параметрів було обрано лише ios::in та ios::out.

Після завершення роботи з файлом його слід закрити за допомогою функції close (). Також варто відзначити, то при виході об'єкта потоку з області видимості, він видаляється, і у нього автоматично викликається функція close.

Запис інформації у файл у програмі ведеться за допомогою оператора вставки потоку (<<) так само, як цей оператор використовується для виведення інформації на екран. Єдина відмінність полягає в тому, що замість об'єкта cout використовується об'єкт fstream.

При зчитуванні інформації з файлу у програмі використовується оператор вилучення потоку (>>), так само, як цей оператор використовується для введення інформації з клавіатури. Єдина відмінність полягає в тому, що замість об'єкта cin використовується об'єкт fstream.

Таким чином, використовуючи готову бібліотеку та класи С++ можна реалізувати процес зчитування та запису у файл.

**РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ**

Розроблений проект був створений методами колективної розробки з постійним редагуванням та вдосконаленням програми новим функціоналом. Додавались доопрацювання як для правильного функціонування програми, так і для зручності у використанні.

**2.1. Загальний опис продукту**

Весь код програми було вирішено розподілити в декілька файлів для розбиття на окремі модулі, розділення інтерфейсу з реалізацією та для зручності в роботі.

В головному файлі CourseWork.cpp, що містить функцію main(), на початку відбувається підключення бібліотеки <iostream>, що надає засоби введення-виведення для стандартної консолі. Містить два основних класи: cin – для обробки вводу з клавіатури, cout – для виводу в консоль даних.

Для розділення оголошення від реалізації було створено заголовні файли. Метою заголовних файлів є зручне зберігання набору оголошень об'єктів для їх подальшого використання в інших програмах. Підключені заголовні файли:

- "Functions.h" — містить прототипи всіх існуючих функцій, які не є членами класу(getDate(), hasOnlyDigits(const string s));

- "Student.h" — містить оголошення класу Student;

- "Characteristic.h" — містить оголошення класу Characteristic.

Для контролювання, щоб конкретний заголовний файл при компіляції підключався строго один раз, на початку кожного заголовного файлу підключається #pragma once — поширена директива препроцесору.

**2.2. Опис головних класів, змінних та функцій програми**

Вся програма розділена на декілька підрозділів та функції main, яка зв’язує ці підрозділи та утворює повноцінний програмний продукт.

Створено 2 класи:

* Student;
* Characteristic.

В класі Student ініціалізовані поля для запису користувачем інформації про людину, на яку буде писатись характеристика. З таблиці 2.1. можна побачити основні поля класу Student та їх призначення.

Таблиця 2.1. Поля класу Student

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| studentName | string | ім'я студента |
| studentSex | int | стать студента |
| studentAge | string | вік студента |
| studentHobby | int | види захоплень |
| numOfQualities | int | кількість особистих якостей |
| studentQualities | string\* | масив для запису особистих якостей |
| studentQuality | string | окремо взята якість студента |

Методи, за допомогою яких можна виконувати дії над даними класу Student:

- void studentInput() — функція, за допомогою якої користувач може ввести дані про студента/учня;

- void studentOutput() — функція для виведення в консоль вхідних даних про особу користувачем;

- bool studentCharact(string inputFile, string outputFile) — функція, що записує сформовану характеристику у файл, де параметр string inputFile є ім’ям файлу, з якого зчитуються дані, а string outputFile є ім’ям файлу, в якому безпосередньо створюється готова характеристика. Функція повертає булеве значення, яке характеризує успішність написання характеристики;

- ~Student() — деструктор для звільнення пам’яті.

В класі Characteristic містяться основні відомості про характеристику, які зображено в таблиці 2.2.

Таблиця 2.2. Поля класу Characteristic

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| charactRating | int | рейтинг характеристики |
| charactLanguage | int | мова |
| charactPurpose | int | призначення характеристики |
| nameOfStudentFile | string | ім'я файлу для запису відомостей про людину |
| fileExtension | string | розширення файлу |
| directoryName | string | назва директорії для збереження файлів |
| allowableExtensions | string | масив із можливих розширень файлів |

В класі Characteristic містяться такі необхідні методи:

- void charactInput() — функція для введення та вибору даних про тип характеристики;

- void charactOutput() — функція для виведення в консоль введених даних про характеристику;

- string charactChoice() — функція для вибору необхідного шаблону на основі введених користувачем даних, повертає назву необхідного шаблону;

- string getNameOfStudentFile() — використовується для отримання назви файлу, в який ведеться запис, повертає його назву;

- void makeDirectory() — для створення папки, в якій будуть зберігатися файли;

- bool matchExtension(string str) — для перевірки дозволених розширень файлів, де параметр string str — задане розширення, повертає булеве значення, яке означає, чи є задане розширення підходящим, чи ні.

У програмі створено дві окремі функції, які не є членами класів. Функція string getDate() визначає поточну дату написання характеристики та повертає її, після чого ця дата вставляється у вихідний файл. Функція bool hasOnlyDigits(const string s) використовується для перевірки рядка на наявність символів.

Про існуючі глобальні змінні у програмі можна дізнатися з таблиці 2.3.

Таблиця 2.3. Глобальні змінні

|  |  |  |
| --- | --- | --- |
| Назва змінної | Значення | Призначення |
| EXTENSIONS\_NUM | 3 | кількість дозволених розширень файлу |
| MAX\_SIZE | 255 | Максимальний розмір введеної строки |
| MIN\_SIZE | 10 | Мінімальний розмір введеної строки |

**РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАЧА**

В даній курсовій роботі інтерфейс програми реалізований в консолі середовища розробки Visual Studio.

**3.1. Опис інтерфейсу**

Інтерфейс виконує функцію взаємодії з користувачем. Саме він забезпечує зручне, зрозуміле і естетичне подання інформації і її обмін з користувачем.

// діаграма використання

**//3.2. Перевірка роботи програми**

**ВИСНОВКИ**

# В результаті виконання курсової роботи було розроблено та реалізовано алгоритм автоматичного написання характеристики. Програма коректно реалізовує всі необхідні функції, результат відповідає вимогам написання документу характеристики.

В першому розділі записки було проведено детальний аналіз поданого завдання, вивчення теоретичних питань, сформовано постановку задачі.

У другому розділі було обґрунтовано вибір середовища розробки та мови програмування, описано структуру, основні підрозділи, функції та класи програми.

У третьому розділі …

Отже, всі пункти розробки були виконані в повному обсязі. Надалі програмне забезпечення має багато можливостей покращення, вдосконалення та оптимізації, наприклад, шляхом додання графічного інтерфейсу. Це досягається зрозумілою архітектурою та поділом системи на підсистеми, що виконують свою окрему функцію. Основною перевагою, на яку було звернено увагу – це простота інтерфейсу користувача, оскільки чим менше часу йому знадобиться на розуміння інтерфейсу, тим швидше і ефективніше буде проводитись взаємодія з програмою.

При виконанні завдання курсової роботи були освоєні та вдосконалені навички розробки та проектування програмного продукту методами програмування С++.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

<http://programming.in.ua/programming/basisprogramming/25-oop.html>

<https://sebweo.com/ru/chto-takoe-git-i-github-rukovodstvo-dlya-nachinayuschih/>

<https://tproger.ru/translations/difference-between-git-and-github/>

<https://metanit.com/cpp/tutorial/8.2.php>

**ДОДАТОК 1**

Специфікація

НТУУ «КПІ» ТЕФ АПЕПС ТВ-01

Листів 1

Київ – 2021

|  |  |  |
| --- | --- | --- |
| Призначення | Найменування | Примітки |
| Головний файл програми з основною функцією | CourseWork.cpp | Використання функції main() |
| Файл програми, що містить оголошення класу Student | Student.h | Містить клас Student |
| Файл програми, що містить функціонал класу Student | Student.cpp | Містить функції класу Student: studentInput(), studentOutput(), studentCharact(string inputFile, string outputFile), getName() |
| Файл програми, що містить оголошення класу Characteristic | Characteristic.h | Містить клас Characteristic |
| Файл програми, що містить функціонал класу Characteristic | Characteristic.cpp | Містить функції класу Characteristic: charactInput(), charactOutput(), charactChoice(), getNameOfStudentFile(), makeDirectory(), matchExtension(string str) |
| Файл програми, що містить оголошення функцій | Functions.h | Містить прототипи функцій getDate(), hasOnlyDigits(const string s) |
| Файл програми, що містить функції | Functions.cpp | Містить функції getDate(), hasOnlyDigits(const string s) |

**ДОДАТОК 2**

Текст / Код програми

НТУУ «КПІ» ТЕФ АПЕПС ТВ-01

Листів

Київ – 2021

//CourseWork.cpp

**ДОДАТОК 3**

Опис програми

НТУУ «КПІ» ТЕФ АПЕПС ТВ-01

Листів «кількість»

Київ – 2021

Розписати на 3-5 сторінок відповідно до інструкцій з файлу «Оформлення записки»

Оформити за правилами оформлення основного текст