Етап 1 : вибір курсів для вивчення

Для вивчення було обрано курс «Основы разработки на C++ : красный пояс» спеціалізації «Искусство разработки на современном C++» на платформі Coursera

Посилання на курс : https://www.coursera.org/learn/c-plus-plus-red?specialization=c-plus-plus-modern-development#reviews

Посилання на спеціалізацію : https://www.coursera.org/specializations/c-plus-plus-modern-development

Чому було обрано саме ці курси :

1)На момент вибору уже було пройдено 2 курси цієї спеціалізації : «Основы разработки на C++ : белый пояс» та «Основы разработки на C++ : желтый пояс», тому було б логічним продовжити проходження спеціалізації.

2)Цей курс містить доволі цікаву програму, посилання на яку *подано нижче*

https://www.coursera.org/learn/c-plus-plus-red?specialization=c-plus-plus-modern-development#syllabus

3)Вивчаються сучасні можливості C++

Етап 2 : проходження курсів

- Тиждень 1 : макроси та шаблони класів

Розпочинається вивчення курсів з розгляду макросів, а саме :

-Принцип їхньої роботи

-Оператор#

-Макроси `__LINE__` та `__FILE__`

-Приклади поганого використання макросів

Також, під час розгляду цієї теми лектор показує, як за допомогою макросів можна покращити unit test фреймворк, який був розроблений на попередньому курсі(*https://github.com/ram333n/cppcoursera/blob/main/Red/Final/Part%202/test_runner.h*), а саме автоматизувати виведення інформації під час непроходження тесту.

Варто відмітити, що деякі задачі курсу на цю тему дають змогу самому переконатися у доцільності використання і уникнення макросів. Наприклад, як за допомогою макросів можна значно спростити функцію сортування для класу за окремими полями, вставивши компаратор у вигляді лямбда-функції у макрос з параметром назви поля, яке порівнюється; або змінити макрос таким чином, щоб він коректно працював, при цьому шукаючи його «слабкі місця».

Що нового дізнався : що таке оператор#, макроси `__LINE__` , `__FILE__` та як правильно використовувати макроси.

Далі вивчаються шаблони класів. Теми цього блоку :

-Класи та шаблони класів

-Інтеграція класу в range-based for

-Правила виводу типів у шаблонах класів

-Використання `auto`

В загальному, до цього я вже мав досвід роботи з шаблонами класів(наприклад, під час створення структур даних), тому це поняття було знайомим, однак блок містив цікаві для мене теми : що потрібно зробити, щоб для своїх класів можна було використовувати range-based for? І саме на одній з лекцій розповідається про це(сталося прозріння ☺). Також цікавими були лекції про правила виводу типів у шаблонах класів та про `auto`, де розповідали, як можна скоротити код, не пишучи великі шаблонні типи, і дали рекомендації щодо цих правил.

На основі цих знань потрібно було написати шаблон Paginator, який бере контейнер і «нарізає» його на сторінки(зазначу, що потім цей шаблон буде використовуватись на 5 тижні для багатопоточності)

Шаблон Paginator : *https://github.com/ram333n/cppcoursera/blob/main/Red/Week%201/paginator.cpp*

Що нового дізнався : Інтеграція класу в range-based for, рекомендації щодо використання `auto` та правил виводу типів у шаблонах класів

- Тиждень 2 : принципи оптимізації коду, ефективне використання потоків вводу, виводу та складність алгоритмів

У циклі занять з принципів оптимізації коду було наведено два основні правила оптимізації коду : уникнення передчасної оптимізації та вимірювання часу роботи програми, та написано простенький таймер для вимірювання часу роботи блоку коду(https://github.com/ram333n/cppcoursera/blob/main/Red/Final/Part%201/profile.h). Також, було дано декілька завдань, у яких потрібно було замінити певні контейнери, щоб збільшити швидкість виконання програми.

В принципі, поради були корисними, особливо про передчасну оптимізацію, коли розв'язував подальші завдання курсу.

Після цього розповідалося про потоки вводу, виводу : як саме вони влаштовані, в чому різниця між `std::endl` та '\n', та як зменшити час роботи на вводі та виводі.

А от далі пішла найскладніша тема цього тижня : складність алгоритмів. Взагалом, проблем з визначенням асимптотики алгоритму проблем не виникло, але задачі цього блоку були одні з найскладніших, тому що треба було продумати правильні комбінації контейнерів, щоб складність була достатньою для зарахування(https://github.com/ram333n/cppcoursera/tree/main/Red/Week%202)

Що нового дізнався : принципи оптимізації коду, як написати свій таймер для вимірювання блоків коду, потоки вводу, виводу, `std::endl,` *складність алгоритмів, амортизаційна складність*

- Тиждень 3 : модель пам'яті в C++

Протягом 2,5 курсів ні разу не розповідали про вказівники. І саме на цьому тижні було це «прокляття» розвінчено ☺.

Тиждень було розпочато з вивчення моделей пам'яті : stack, heap; вказівників, проблеми роботи з ними(memory leak), оператори new, new[], delete, delete[] : різниця між ними, арифметика вказівників.

Хочу відмітити ілюстративну лекцію з оператором delete, на якій показали необхідність цього оператора, виділивши 15 гб неочищеної пам'яті.

Далі розповідалося, як написати свій простий вектор, використовуючи вивчений матеріал.

Що нового дізнався : stack, heap, як написати свій вектор.

- Тиждень 4 : ефективне використання лінійних контейнерів

Цей і наступний тиждень були для мене найцікавішими, адже містили багато задач, які вимагали знання про контейнери, їх операції та складність, що є дуже важливим.

Тиждень містив лекції про лінійні контейнери, їх будова та властивості(інвалідація ітераторів, інвалідація посилань, складності певних операцій)

Цікавим було порівняння операцій цих контейнерів на великих обсягах даних, вимірюючи час(так званий benchmark)

Контейнери, які розглядалися :

- `std::vector`
- `std::deque`
- `std::list`
- `std::forward_list(в тестах)`
- `std::array`
- `std::string_view`

Матеріал цього тижня навчив мене заздалегідь оцінювати складності основних операцій цих контейнерів, знаючи їх будову.

Також для мене новим та специфічним був контейнер `std::string_view,` який з'явився у C++ 17. Спочатку було доволі важко зрозуміти його використання, але після розв'язання декількох задач стало зрозуміло.

Що нового дізнався : інвалідація ітераторів та посилань, будова `std::vector, std::deque, std::array, std::string_view,` ефективне використання лінійних контейнерів.

- Тиждень 5 : Move-семантика та базова багатопоточність

Цей тиждень містив абсолютно нову інформацію для мене.

Спочатку йшла мова про тимчасові об’єкти: коли вони з’являються; поняття переміщення об’єкта, функція `std::move` : її застосування та коли вона не працює.

На основі цього вводяться поняття операторів та конструкторів переміщення, копіювання, rvalue, lvalue посилання, NRVO, copy elision.

Задачі цього блоку були розраховані на роботу з некопійованими об’єктами. Наприклад, було завдання покращити вектор з третього тижня так, щоб він міг зберігати некопійовані об’єкти; дописати оператори та конструктори копіювання, переміщення.

Що нового дізнався : тимчасовий об’єкт, функція `std::move,` rvalue, lvalue посилання, оператори та конструктори переміщення та копіювання, як працювати з некопійованими типами, що таке NRVO, copy elision.

Остання тема курсу – базова багатопоточність, в якій пояснюють концепцію багатопоточного програмування, як запустити асинхронну операцію, використовуючи `std::future, std::async;` data race та `std::mutex, std::lock_guard.` Це все було проілюстровано прикладами задач, наприклад, паралельне сумування матриць, застосовуючи також шаблон Paginator.

Що нового дізнався : `std::future, std::async;` data race та `std::mutex, std::lock_guard.`

- Тиждень 6 : Фінальна задача(«Пошукова система»)

Найскладніша задача курсу, в якій треба було розробляти простеньку реалізацію пошукової системи.

Задача складалася з 2 частин :

Частина 1: Дано правильну реалізацію цієї системи, яка проходить успішно всі unit-tests, але, вона є недостатньо швидкою. Треба було оптимізувати її, щоб час роботи програми вкладався в певні часові межі.

Ключовим моментом цієї частини було знання певних контейнерів, адже за допомогою цього можна було зменшити кількість

елементарних операцій. Наприклад, знаючи, що доступ до елемента за ключем в `std::map` займає логарифмічний час, а доступ до елемента за індексом в `std::vector` – константа(у даному випадку ключі в словнику мали тип `size_t`, а використання `std::unordered_map` не передбачалось), можна було оптимізувати 2 вкладені цикли, зменшивши к-ть операцій в 16 раз.

Взагалом, найскладнішим тут було правильно спроектувати взаємодію між об'єктами, щоб все правильно і швидко працювало.

Частина 2: Використовуючи реалізацію з першої частини, треба було покращити пошукову систему так, щоб вона могла обробляти запити з декількох потоків.

В цій частині основним було правильно використати можливості багатопоточного програмування : запуск асинхронних операцій, захист мьютексом даних. Якщо це грамотно застосувати, то ця частина була нескладною.

Етап 3 : підсумки вивчення

1. Наскільки зміст онлайн-курсу відповідав Вашим очікуванням щодо нього, чи коротким описам/оглядам, які дивились до початку вивчення?

   Курс повністю відповідав очікуванням

2. Наскільки зрозумілими було викладення матеріалу, пояснення?

   Курс був майже повністю зрозумілий. Якщо щось навіть не розумів щось, то можна було написати в Telegram спеціалізації, де можна оперативно отримати пояснення .

3. Чи був матеріал проілюстрований прикладами? Наскільки вдалими були ці приклади, чи вони демонстрували якісь важливі аспекти?

Так, прикладів було достатньо. Вони були дуже вдалими, показували конкретне призначення певних конструкцій та правил на певних практичних задачах.

4. Чи були викладені матеріали занадто складними для розуміння, чи навпаки, занадто простими і очевидними, які не варто було так детально пояснювати?

   Матеріал містив інформацію оптимальної складності : не зовсім складну і не зовсім очевидну.

5. Наскільки вдало було організовано структуру матеріалів? Чи всі потрібні передумови для розуміння певної теми були викладені перед описом цієї теми (або хоча б було посилання, де можна знайти ці передумови)?

   Матеріал викладався поступово, тому проблем з розумінням лекцій через нерозуміння попереднього матеріалу не було(принаймні у мене).

6. Наскільки вдалими були завдання, які треба було виконувати в рамках онлайн-курсу? Наскільки вони відповідали темі курсу, сприяли вивченню матеріалів? Наскільки вони були складними для реалізації, чи не потребували додаткових знань та навичок, які не розглядались в курсі? Як багато часу знадобилось на їх реалізацію? Як було організовано перевірку завдань, наскільки вона була корисною для опанування матеріалами курсу?

   Майже всі завдання були вдалими, повністю відповідали темі курсу, але деякі з них мали не зовсім чітку умову, що іноді трохи дратувало. Складність більшості задач було оптимальною, деякі – складними(фінальна задача). Завдання відповідали темі курсу, іноді сприяли вивченню документації(як ж без цього☺). Задачі вимагали не стільки багато коду, скільки правильно спроектувати, підібрати контейнери, і це доволі багато часу

займало(наприклад, на фінальну задачу пішло декілька днів) , але з часом розумієш, що став краще проектувати програми.
Завдання перевірялися тестовою системою, яка компілювала код і «проганяла» програму на unit-тестах.

7. Чи не було в матеріалах онлайн-курсу якихось помилок, недоліків оформлення, застарілої інформації?

   Не було, оскільки автори досі підтримують курси : оновлюють деяку інформацію, додають нові задачі(так було з другим курсом спеціалізації).

8. Наскільки цікавим було вивчення тем з використанням цього онлайн-курсу?

   На всі 100%. Практичні приклади + цікаві задачі робили все, щоб вивчення C++ було цікавим.

9. Наскільки корисним виявився цей онлайн-курс?

   Майже усі знання про сучасний C++ я отримав саме з цього циклу курсів, тому, курс виявився корисним на всі 100%.

10. Загалом, що Вам сподобалось в цьому онлайн-курсі?

    Матеріал та його викладання, майже всі задачі – в принципі майже все.

11. Загалом, що Вам не сподобалось в цьому онлайн-курсі?

    Деякі задачі мали не зовсім чітку умову, і для того, щоб написати код так, щоб задовільнити тестову систему, потрібно було розглянути всі варіанти, але на щастя, цього вдалося уникнути через спілкування в чаті спеціалізації в Telegram.

12. Чи рекомендували б Ви проходження цього онлайн-курсу іншим студентами з приблизно таким самим рівнем початкових знань?

Безперечно так

Посилання на репозиторій з розв'язками задач :

https://github.com/ram333n/cppcoursera/tree/main/Red

Посилання на сертифікат :

https://www.coursera.org/account/accomplishments/verify/7FLUZLP966BQ