Лекція 1. Вступ.

Версія 25 лютого 2023 р.

Анотація

Ця лекція охоплює знайомство з мовою програмування C++, огляд її основних властивостей, мотивація, налаштування середовища.

Зміст

1 Чому слід вивчати та використовувати C++?

Мова програмування C++ має велику історію, широку стандартну бібліотеку та підтримку, велику спільноту розробників, дослідників та різного штибу прихильників, які використовують її у своєму професійному житті та/або для власних проектів.

В наукових, інженерних або дослідницьких колах вона часто допомагає писати більш універсальний, а головне швидкий та ефективний код, коли простіші інструменти на кшталт MatLab або Python не підходять. Такі програми можуть зазвичай виконуватись на більшій кількості платформ.

Сучасні стандарти (C++17 та C++20) роблять з неї у багатьох сенсах високорівневу мову програмування, яка тим не менш залишає розробнику можливість оптимізувати свої програми на більш "низькому" рівні, і яка максимально зберігає свою славетну швидкість та ефективність.

Більшості, C++ відома як об'єктно-орієнтована мова програмування, але вона прекрасно годиться для інших парадигм програмування, або їх

комбінації. C++, особливо останні стандарти, дозволяє розробнику (автору) чітко та неприховано висловлювати свої наміри при написанні коду, що майже робить непотрібними взагалі.[1] Це досягається побудовою та використанням доречних структур та класів для своєї задачі, назвами, що допомагають зрозуміти роль та основні властивості того, що вони охрестили, та написанням продуманих абстракцій.

2 Що потрібно для того, щоб почати працювати з C++?

Щоб мати змогу отримувати файл, який можна запустити, з вашого вихідного коду, треба мати встановленим C/C++ компілятор. Для операційних систем Windows популярним вибором є Visual Studio C/C++ Compiler, для GNU/Linux та MacOS – GCC від GNU або Clang від LLVM. В зв'язці з окремо взятим компілятором, як правило, їдуть набір супроводжуючих інструментів для форматування, аналізу, налагодження (debugging), заміру різних характеристик вихідного коду та скомпільованих програм. Радимо ознайомитись з можливостями того комплекту інструментів, який ви збираєтесь встановлювати.

Для безпосереднього написання та налагодження роботи програми існує два підходи:

(a) використання інтегрованих середовищ розробки (IDE), наприклад, Visual Studio, Eclipse, CLion та ін.,

(a) використання сучасних гнучких текстових редакторів.

Роблячи вибір для себе, слід зауважити наступне

- IDE це великі програми, які роблять багато чого. Часом надто багато. Вони можуть приховувати деякі технічні деталі того, що відбувається, що само по собі призвано привносити зручності і простоту в процес роботи. Проте, справжній спеціаліст мусить розуміти все, що відбувається за лаштунками, тому тим, хто що навчається, не варто пропускати якісь кроки або деталі.
- Знову ж таки, IDE це великі програми, вони намагаються максимально спростити роботу для вас, і тому вони насичені різноманітним функціоналом, якій не дуже може бути потрібним чи корисним, але на практиці це виливається в більшу витрату ресурсів комп'ютера при роботі, повільну роботу з текстом.
- Цілі IDE або деякі їх функції можуть бути платними.

[1] Дійсно, більшість програмістів вважають їх недоречними в сучасних програмах, окрім дуже окремих випадків.

- На противагу ним, сучасні текстові редактори на кшталт Visual Studio Code, Sublime Text та деякі інші, не відстають від найпросунутіших IDE в тому що стосується аналізу кода, навігації, автоматизації задач, статичному аналізі вихідного коду.
- Вони у деякому сенсі більш гнучкі – деякі функції можуть не бути там за замовченням, [2] але вони можуть буди задані через системні команди, прописані в конфігураційних файлах. Це більш гнучкий, універсальний і природний для професійних програмістів підхід, ніж вишукування налаштування в графічному інтерфейсі окремо взятої IDE.
- Робота з ними більш плавна та вони забирають менше системних ресурсів.
- Для популярних редакторів, таких як Visual Studio Code, існує безліч безкоштовних додатків, які розширюють його можливості до рівня повноцінного IDE.

IDE, саме, часто просто інтегрують out-of-the-box інтеграцію з загальнодоступними інструментами, такими як

- `clang_format` для форматування вихідного коду
- `cpplint`, `cppcheck`, `Clang-Tidy` і багато інших для аналізу коду до моменту компіляції.

Також, зазначимо важливість вибору і дотримання стилю написання та оформлення коду. Це стосується відступів, назв, розміру функцій перед тим як їх варто розділити на під-функції, максимальна ширина строки, положення дужок, заборона на використання потенційно "небезпечних" можливостей, переваги при виборі типів і багато-багато іншого. Непоганою вихідною точною є Google Code Style Sheet. Також корисним буде ознайомитись з C++ Core Guidelines.

Наостанок, нагадаємо, що робота над складними, багатокомпонентними проектами не обмежується використанням одного тільки інструмента, як C++. Як мінімум, інженер повинен бути впевненим користувачем системи контролю версій Git, а також навичками роботи в Linux/Unix терміналі. Для освоєння цих систем, існує безліч корисних безкоштовних відеоматеріалів, онлайн курсів, статей та ін. Наприклад:

- Книга про Git.
- YouTube відео: вступ до Git.
- YouTube відео: вступ до Bash.
- YouTube відео: комбінування команд в Bash.

[2] Велика вірогідність, що для вашого випадку вже існує безкоштовне розширення!

Завдання

Налаштувати свій робочий комп'ютер під роботу з C++: встановити все необхідне програмне забезпечення, скомпілювати програму "Hello world" та інші доступні на своєму комп'ютері, переконатись в коректній роботі.