Компетентності рівня junior для C++ Developer	**GlobalLogic**
1. Основи C++	
1.1 Змінні та типи даних. Знає основні типи даних у C++ (int, float, char, bool, double, тощо) та вміє оголошувати і використовувати змінні різних типів. Розуміє поняття масивів та рядків, вміє оголошувати, ініціалізувати та маніпулювати масивами та рядками, включаючи використання стандартної бібліотеки C++ для роботи з рядками (std::string). Володіє базовими знаннями про вказівники: оголошення, присвоєння адреси, арифметика вказівників, розіменування та використання вказівників для динамічного розподілу пам'яті. Розуміє основні концепції управління пам'яттю, включаючи динамічний розподіл та звільнення пам'яті за допомогою операторів new і delete.	3
1.2 Структури даних. Розуміння і використання структур даних, таких як стеки, черги і дерева. Хеш-таблиці. Знає основні структури даних, такі як стеки, черги, дерева та хеш-таблиці, та їх реалізацію у C++. Вміє використовувати стандартні контейнери STL (Standard Template Library) для роботи з цими структурами, зокрема, вміє застосовувати std::stack, std::queue, std::priority_queue, std::map, std::unordered_map, std::set, та std::unordered_set. Розуміє принципи роботи кожної структури даних, їхні переваги та недоліки, а також ситуації, у яких доцільно застосовувати кожну з них. Вміє ефективно використовувати ці структури для вирішення типових задач програмування, таких як управління пам'яттю, пошук і сортування даних.	3
1.3 Оператори та цикли, інструкції мови, умовні конструкції Знає основні оператори C++ (арифметичні, логічні, порівняння, присвоєння) та вміє використовувати їх для виконання базових операцій над даними. Розуміє і вміє працювати з різними типами циклів (for, while, do-while) для ітераційних процесів. Володіє знаннями про умовні конструкції (if, else, switch) і може ефективно використовувати їх для контролю потоку виконання програм. Вміє застосовувати оператори та цикли для вирішення стандартних завдань програмування, таких як обробка масивів, робота з рядками та управління логікою програми.	3
1.4 Функції. Знає основні принципи створення та використання функцій у C++. Вміє оголошувати та визначати функції, правильно викликати їх у коді, а також розуміє механізми передачі параметрів за значенням та за посиланням. Здатен коректно повертати значення з функцій, використовуючи відповідні типи даних. Вміє використовувати функції для організації коду, забезпечення його структурованості та повторного використання, що сприяє підвищенню читабельності та ефективності програм.	3
1.5 Основи структури коду Знає основні компоненти структури коду C++, такі як оголошення заголовкових файлів (.h), файлів реалізації (.cpp), та головного файлу програми (main.cpp). Розуміє принципи використання простору імен (namespaces) для організації коду та уникнення конфліктів імен. Вміє організовувати код у функції, класи та об'єкти, забезпечуючи модульність та повторне використання. Вміє працювати з основними директивами препроцесора (#include, #define), а також розуміє значення інкапсуляції та модифікаторів доступу (public, private, protected) у класах для захисту даних.	3

1.6 Помилки та виключення Знає основні механізми обробки виключень у C++ (try, catch, throw). Розуміє, як використовувати конструкції try-catch для перехоплення та обробки помилок під час виконання програми, забезпечуючи стабільність та надійність коду. Вміє створювати та кидати власні виключення, а також використовувати стандартні виключення C++ для належної обробки помилкових ситуацій. Здатний писати код, який може ефективно реагувати на виключення, мінімізуючи їх вплив на виконання програми.	3
2. Теоретичний блок. Розуміння ООП	
2.1 Основні концепції ООП. Робота з класами, об'єктами і наслідуванням в C++. Знає основні принципи об'єктно-орієнтованого програмування (ООП) в C++, включаючи поняття класів та об'єктів, інкапсуляції, наслідування та поліморфізму. Вміє створювати класи, оголошувати та визначати їх властивості (поля) та методи (функції-члени). Розуміє, як використовувати конструктори та деструктори для управління життєвим циклом об'єктів. Здатний реалізовувати наслідування для створення похідних класів, розширення та переопределення їх функціональності. Володіє навичками використання ключових слів private, protected, public для управління доступом до членів класу, а також розуміє принципи поліморфізму та віртуальних функцій для забезпечення динамічного зв'язування під час виконання програми.	3
2.2 Управління ресурсами. Використання операторів new і delete для динамічного виділення та звільнення пам'яті. Файлові дескриптори. Знає основні принципи управління пам'яттю в C++ та використання операторів new і delete для динамічного виділення та звільнення пам'яті. Вміє коректно виділяти пам'ять для об'єктів і масивів, а також звільняти її, запобігаючи витокам пам'яті. Розуміє концепцію файлових дескрипторів, знає, як їх використовувати для роботи з файлами, відкриття, читання, запису та закриття файлів у програмі. Вміє управляти ресурсами ефективно, забезпечуючи стабільну роботу програми.	3
2.3 Перевантаження операторів. Створення власних реалізацій операторів для класів. Знає основні принципи перевантаження операторів у C++. Вміє створювати власні реалізації операторів для класів, таких як арифметичні, логічні, порівняльні та інші спеціальні оператори. Розуміє, коли та чому доцільно використовувати перевантаження операторів для покращення читабельності та функціональності коду. Вміє визначати перевантажені оператори як члени класу або як глобальні функції, та використовувати ключове слово friend для доступу до приватних і захищених членів класу.	3
2.4 Використання шаблонів для створення загальних алгоритмів і класів. Створення і використання абстракцій для розширення інтерфейсів та зменшення залежностей. Знає основні принципи використання шаблонів у C++ для створення узагальнених алгоритмів і класів, що дозволяють працювати з різними типами даних без повторного написання коду. Вміє створювати шаблони функцій та класів, а також використовувати стандартні шаблони з бібліотеки STL (Standard Template Library). Розуміє концепцію абстракцій для розширення інтерфейсів та зменшення залежностей між компонентами, що сприяє більшій гнучкості та повторному використанню коду. Вміє застосовувати ці принципи для підвищення ефективності та продуктивності розробки програмного забезпечення.	3
3. Розширені знання C++	
3.1 Робота з файлами. Читання і запис даних у файли за допомогою потоків вводу-виводу (fstream). Розуміє основи роботи з потоками вводу-виводу у C++. Вміє використовувати бібліотеку <fstream> для читання та запису даних у файли. Знає, як використовувати ifstream для відкриття файлів на читання, перевіряти успішність відкриття файлу та обробляти можливі помилки. Вміє зчитувати дані різних типів (текстові та числові) з файлів і записувати дані у файли за допомогою ofstream. Розуміє важливість закриття файлів після завершення роботи та вміє перевіряти досягнення кінця файлу за допомогою методу eof().	3
3.2 Бібліотека STL (Standard Template Library). Використання контейнерів (vector, map, set), алгоритмів (sort, search) та інших компонентів STL. Знає основні компоненти бібліотеки STL, включаючи контейнерні класи (vector, map, set) та алгоритми (sort, search). Розуміє, як використовувати ці компоненти для ефективного зберігання та обробки даних. Вміє вибирати відповідні контейнери для різних завдань, використовувати вбудовані алгоритми для сортування та пошуку, а також реалізовувати власні алгоритми на основі STL. Здатний інтегрувати ці елементи у програми для підвищення їх продуктивності та організації коду.	3

3.3 Робота з багатопотоковим програмуванням і бібліотеками, такими як C++11 thread та mutex Знає основи багатопотокового програмування в C++, включаючи використання бібліотек C++11 для створення та управління потоками. Вміє використовувати класи std::thread для створення та запуску нових потоків, а також механізми синхронізації, такі як std::mutex, для уникнення проблем з доступом до спільних ресурсів. Розуміє принципи блокування та розблокування, а також може використовувати умови (std:: condition_variable) для синхронізації потоків. Здатний писати безпечний і ефективний код для обробки багатопоточних задач, уникати типових помилок, таких як дедлоки та гонки за даними (race condition).	3
3.4 Лямбда-функції: Використання анонімних функцій для покращення коду та зменшення повторень. Знає основи лямбда-функцій у C++, включаючи синтаксис і правила їх використання. Вміє створювати анонімні функції для виконання локальних обчислень без потреби в окремих функціях. Розуміє, як використовувати лямбда-функції для покращення читабельності коду, зменшення повторень і оптимізації роботи з колекціями. Вміє передавати параметри в лямбда-функції та управляти захопленням змінних з навколишнього контексту, що дозволяє ефективно працювати з функціональними об'єктами і алгоритмами STL.	3
3.5 Використання мов асемблера для оптимізації вузьких місць в програмному коді. Знає основи мов асемблера та їх роль у низькорівневій оптимізації програмного коду. Вміє ідентифікувати вузькі місця в коді на рівні C++ і використовувати асемблер для їх оптимізації, забезпечуючи підвищення продуктивності. Розуміє основи структури машинного коду та може писати прості фрагменти асемблерного коду для інтеграції з C++ програмами, використовуючи вбудовані асемблерні інструкції для досягнення критичних оптимізацій.	2
3.6 Системне програмування. Робота з операційними системами, системними викликами та системними ресурсами. Знає основи системного програмування в C++, включаючи принципи роботи з операційними системами. Вміє використовувати системні виклики для взаємодії з ОС, управляти системними ресурсами, такими як пам'ять і файли. Розуміє основи управління процесами та потоками, а також знає, як працювати з низькорівневими ресурсами системи, такими як файлові дескриптори і семафори. Вміє писати код, який взаємодіє з операційною системою, забезпечуючи ефективність та надійність системного програмного забезпечення.	2
3.7 Взаємодія між процесами в операційних системах. Сокети, черги повідомлень, спільна пам'ять і семафори. Концепції RPC та IPC. Знає основні механізми взаємодії між процесами в операційних системах, такі як сокети, черги повідомлень, спільна пам'ять і семафори. Розуміє концепції Remote Procedure Call (RPC) та Inter-Process Communication (IPC), а також їх роль у забезпеченні комунікації між різними процесами. Вміє реалізовувати прості програми, які використовують ці механізми для обміну даними між процесами, забезпечуючи ефективну синхронізацію і взаємодію в багатозадачних середовищах.	3
3.8 Взаємодія з мережевими протоколами та бібліотеками для створення мережевих додатків. Знає основи мережевих протоколів (TCP/IP, UDP) та принципи їх роботи. Вміє використовувати стандартні бібліотеки C++ (як-от Boost.Asio) для реалізації мережевих з'єднань, обробки запитів та відправлення даних. Розуміє, як налаштувати сервери та клієнти для взаємодії через мережу, а також як обробляти помилки та забезпечувати безпечну комунікацію. Може ефективно використовувати мережеві функції для створення додатків, що передають дані між різними системами або пристроями.	2
3.9 Розробка API для зовнішніх користувачів або для інтеграції з іншими системами. Знає основні принципи створення API на C++, включаючи концепції REST, HTTP-методів та форматів даних (JSON, XML). Вміє проектувати та реалізовувати інтерфейси API, дотримуючись стандартів і протоколів, таких як REST або SOAP. Вміє розробляти та документувати API для зовнішніх користувачів, а також інтегрувати системи шляхом створення та обробки HTTP-запитів і відповідей. Розуміє, як забезпечити безпеку і ефективність API, а також як тестувати та налагоджувати інтеграцію з іншими системами. Має навички у виборі та використанні бібліотек для роботи з HTTP-запитами і обробки даних у C++. Знає основні принципи розробки API у C++ для взаємодії з іншими системами або зовнішніми користувачами. Розуміє основи серіалізації/десеріалізації даних, обробки запитів і відповідей, а також управління помилками. Здатний забезпечити безпечний доступ до функціоналу та даних через API, враховуючи потреби інтеграції та взаємодії з іншими системами.	2
3.10 Розробка серверних додатків для обробки запитів та роботи з базами даних. Взаємодія з базами даних (SQLite або MySQL). Знає основи розробки серверних додатків на C++, включаючи обробку HTTP-запитів та управління сесіями. Вміє використовувати бібліотеки для роботи з базами даних, такі як SQLite або MySQL, для виконання запитів до бази даних, обробки результатів та управління з'єднаннями. Розуміє, як інтегрувати серверні додатки з базами даних для зберігання та отримання даних, а також забезпечувати ефективність та безпеку взаємодії з базою даних. Вміє реалізувати основні CRUD-операції (створення, читання, оновлення, видалення) та виконувати оптимізацію запитів для покращення продуктивності додатків.	3

3.11 Безпека додатків. Захист від вразливостей, таких як буферні переповнення та ін'єкції. Знає основні принципи безпеки додатків та вразливості, специфічні для C++. Вміє захищати код від буферних переповнень за допомогою перевірок меж масивів та використання безпечних функцій. Розуміє небезпеки, пов'язані з ін'єкціями, та вміє писати код, який запобігає цим атакам. Здатний використовувати інструменти для статичного та динамічного аналізу безпеки коду, аналізувати результати і впроваджувати необхідні виправлення.	3
3.12 Налаштування та автоматизація процесу збірки проєкту із застосуванням інструментів збірки, таких як CMake або Make. Встановлення та керування залежностями проєкту Знає основні інструменти для налаштування та автоматизації процесу збірки проєкту, зокрема CMake та Make. Розуміє функціонал кожного інструменту, включаючи створення та налаштування файлів конфігурації (CMakeLists.txt або Makefile), встановлення та керування залежностями проєкту. Вміє аналізувати потреби проєкту та вибирати відповідні інструменти для ефективного управління процесом збірки, а також налаштовувати автоматизовані скрипти для збирання та тестування коду.	2
3.13 Використання бібліотек і фреймворків, таких як Boost та Qt. Розробка GUI. Знає основні можливості та функціонал бібліотек Boost та Qt для розробки програм на C++. Вміє використовувати Boost для виконання задач загального призначення, таких як робота з контейнерами, багатопотоковість, регулярні вирази та інші корисні утиліти. Розуміє принципи роботи з бібліотекою Qt для створення графічних інтерфейсів користувача (GUI), включаючи створення вікон, віджетів, обробку подій та використання сигналів і слотів. Вміє налаштовувати проєкт для використання цих бібліотек та фреймворків, інтегрувати їх у власні програми та забезпечувати кросплатформенність застосунків.	2
4. Testing	
4.1 Основи тестування Знає основні принципи тестування, його типи (Unit testing, Integration testing, End to End testing) та важливість для розробки програмного забезпечення, здатний виконувати кожен з перелічених типів. Ідентифікує відмінності різних типів тестування в рівні складності та обсязі тестових сценаріїв. Вміння писати невеликі, швидкі та автоматичні тестові сценарії Unit testing для перевірки правильності роботи окремих модулів або методів. Розуміє процес інтеграційного тестування та використання фреймворків, які допомагають тестувати взаємодію між різними компонентами програмного забезпечення. Вміє проводити тестування програмного забезпечення в реальному середовищі з урахуванням всіх компонентів системи. Практикує автоматизацію процесу тестування для забезпечення ефективності і повторюваності тестових сценаріїв. Здатний аналізувати результати тестування, виявляти помилки та робити відповідні корективи, співпрацювати з розробниками для забезпечення якості та правильності коду	2
4.2 Тестові фреймворки для C++, такі як Catch або Boost.Test. Бібліотека Google Test. Знає основні тестові фреймворки для C++ (Catch, Boost.Test, Google Test) та їх функціонал для написання та виконання тестів. Вміє створювати та налаштовувати тестові середовища, писати юніт-тести для перевірки функціональності коду, а також використовувати асерти для валідації результатів тестів. Розуміє принципи написання модульних тестів, тестування на основі фікстур та створення тестових випадків для різних сценаріїв. Здатний аналізувати результати тестів та виявляти помилки для забезпечення високої якості програмного забезпечення.	3
5. Інструментарій розробника	
5.1 Системи контролю версій - GIT Розуміє git flow - методологію роботи з гілками для ефективного управління процесом розробки. Здатний створювати репозиторії та ініціалізувати проєкти з використанням Git. Розуміє та використовує команди Git (commit, push, pull, fetch, checkout, add, status, revert та інші). Вміє працювати з гілками (branches) для розробки функціональності відокремлено від основної гілки (наприклад, розробка нових функцій у власних гілках та їх об'єднання в основну гілку). Знайомий з процедурою створення пулл-реквестів для обговорення та злиття змін з різних гілок.	3

5.2 Навички використання IDE Вміє працювати з популярними IDE, такими як Visual Studio Code, CLion, Eclipse, або Visual Studio. Використовує редактор коду, налаштування розширень та плагінів для поліпшення продуктивності розробки. Застосовує вбудовані інструменти IDE для налагодження коду, аналізу помилок, рефакторингу та контролю якості коду.	3
5.3 Навички використання командного рядка (Terminal) Має базове розуміння навігації по файловій системі через командний рядок. Використовує команди для перегляду, створення, зміни, видалення та архівації файлів та тек через командний рядок. Застосовує через командний рядок команди для роботи з Git, таких як клонування репозиторію, створення гілок, комітів та інші. Вміє користуватися дебаггерами, такими як GDB (GNU Debugger) та Visual Studio Debugger, для знаходження та виправлення помилок у програмному коді, встановлювати breakpoints, оглядати стан програми під час виконання та аналізувати значення змінних.	2
5.4 Знання Linux-середовища Розуміє основні концепції та принципи роботи з операційною системою Linux. Вміє працювати з командним рядком Linux, виконувати команди, керувати файловою системою, управляти користувачами та правами доступу. Працює зі скриптовими мовами, такими як Bash, для автоматизації рутинних завдань та конфігурації середовища.	2
6. Розмовна англійська	
6.1. Правильна побудова фрази з узгодженням часу Вміє використовувати відповідні часові форми дієслів і структури речень для точного вираження часу в комунікації. Здатний правильно використовувати часові форми, такі як Present Simple, Present Continuous, Past Simple, Past Continuous, Future Simple, для передачі правильного значення в повідомленні.	2
6.2 Професійна it термінологія в розмові з замовником Знає і використовує професійні IT-терміни та скорочення, які використовуються в IT-індустрії, для чіткого та зрозумілого спілкування зі замовником. Вміє перекладати складні технічні концепції на доступну мову для замовника, не знайомого з IT-галуззю.	2
6.3 Професійна it термінологія у діловому звіті Знає і використовує спеціалізовану IT-термінологію та фразеологію при написанні ділових звітів. Вміє передати інформацію про технічні питання та проєкти зрозуміло та конкретно, використовуючи адекватні IT-терміни та абревіатури.	1
6.4 Професійна термінологія для C++ Знає та розуміє специфічну термінологію, що використовується в контексті програмування C++ Розуміє та використовує специфічні терміни, патерни, алгоритми, фреймворки, бібліотеки та інші інструменти, які використовуються в C++	
6.5 Неформальне спілкування (small talks) Вміє вести неформальну розмову, розпочинати діалоги та підтримувати невимушену атмосферу. Знає загальновживану термінологію та фрази, які використовуються у неформальних розмовах, таких як загальні питання про погоду, цікаві події, особисті захоплення, спорт, фільми тощо.	1

Коментарі до таблиці

Оцінки компетентностей вказані станом на 2024/2025 рр., по шкалі від 0 до 3.

«3» - компетентність вкрай важлива, якщо кандидат нею не володіє, його подальше просування на вакансію

«2» - компетентність, знання якої обов'язково знадобиться кандидату при подальшому професійному зростанні.

«1» - некритична компетентність, якою можна знехтувати.

«0» - неактуальна компетентність.