МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ  
«ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ЛУЦЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ»

Випускова циклова комісія Компʼютерних систем та інформаційних технологій



**Звіт**

**до практичної роботи №1.2**

з дисципліни “Технології розробки програмного забезпечення”

Виконав: студент **2** курсу, групи **22-ІСТ**

спеціальності: Інформаційні системи та технології

**Синиця Назар**

Викладач: Володимир Остапюк

Луцьк – 2024

**Зміст**

1. **Переваги використання IDE порівняно з простим текстовим редактором?**
2. **Основні функції виконує IDE?**
3. **Поради щодо вибору IDE для конкретної мови програмування або проекту?**
4. **Основні поняття програмування, які потрібно знати для початку розробки ПЗ?**
5. **Типові помилки, яких припускаються початківці при написанні коду?**

**Висновок**

**1. Переваги використання IDE порівняно з текстовими редакторами**

**1.1. Автодоповнення коду**

IDE підказує варіанти синтаксису та методів, зменшуючи кількість помилок і прискорюючи написання коду.

* 1. **Вбудоване налагодження**

Дає змогу перевіряти виконання програми поетапно, переглядати значення змінних і знаходити помилки без додаткових інструментів.

* 1. **Синтаксичний аналіз**

IDE автоматично визначає синтаксичні помилки та підсвічує їх ще до запуску програми.

* 1. **Інтеграція з системами контролю версій**

Завдяки підтримці Git та інших систем ви можете відстежувати зміни, створювати коміти й працювати в команді прямо з IDE.

* 1. **Візуалізація проекту**

Дозволяє легко переглядати структуру файлів і класів, полегшуючи орієнтування у великому проекті.

* 1. **Плагіни та розширення**

IDE підтримує численні плагіни, які додають нові функції та інтегруються з іншими інструментами.

* 1. **Інтеграція з фреймворками та бібліотеками**

Автоматичне налаштування фреймворків і швидкий доступ до потрібних бібліотек допомагають зосередитися на розробці.

* 1. **Візуальні інструменти**

Багато IDE мають вбудовані редактори для створення користувацьких інтерфейсів і роботи з графікою.

* 1. **Документація та підказки**

Надає інформацію про методи, функції та класи, що прискорює процес навчання і розробки.

**2. Основні функції IDE**

* 1. **Редагування коду**

Містить зручні інструменти для автозавершення, підсвічування синтаксису та автоматичного форматування коду.

* 1. **Налагодження**

Дає змогу запускати програму поетапно, встановлювати точки зупинки та аналізувати виконання коду.

* 1. **Компілляція та виконання**

Підтримує виконання і компіляцію програм прямо в середовищі розробки без необхідності в зовнішніх інструментах.

* 1. **Інтеграція з контролем версій**

Дозволяє відслідковувати історію змін, створювати гілки та працювати в команді над проектом.

* 1. **Візуальні редактори та інструменти**

Мають зручні графічні редактори для створення інтерфейсів і проектування програм.

* 1. **Підтримка плагінів і бібліотек**

Спрощує інтеграцію з популярними бібліотеками, розширюючи можливості розробника.

**3. Поради щодо вибору IDE для конкретної мови програмування або проекту?**

* 1. **Виберіть IDE відповідно до мови програмування**

Наприклад, IntelliJ IDEA ідеально підходить для Java, а Visual Studio — для C#.

* 1. **Оцініть потреби проекту**

Якщо ваш проект потребує специфічних фреймворків або технологій, перевірте їхню підтримку в IDE.

* 1. **Інтерфейс та продуктивність**

Вибирайте середовище, яке легко освоїти й яке швидко працює на вашому комп'ютері.

* 1. **Підтримка розширень**

Переконайтеся, що IDE підтримує потрібні вам плагіни або інструменти для розширення функціональності.

**4. Основні поняття програмування, які потрібно знати для початку розробки ПЗ?**

* 1. **Змінні та типи даних**

Змінні — це контейнери для зберігання даних, а типи визначають, які саме дані можна зберігати (числа, рядки тощо).

* 1. **Операції**

До операцій відносяться арифметичні, порівняння, логічні та маніпуляції з бітами.

* 1. **Умови та цикли**

Інструкції if, for та while дозволяють контролювати виконання програм залежно від умов.

* 1. **Функції**

Функції — це блоки коду, які виконують певну дію й можуть бути викликані повторно.

* 1. **Масиви та колекції**

Масиви зберігають групи даних одного типу, а колекції пропонують більшу гнучкість для роботи з даними.

* 1. **ООП (об'єктно-орієнтоване програмування)**

Основними принципами ООП є класи, об'єкти, наслідування, інкапсуляція та поліморфізм.

* 1. **Алгоритми та обробка помилок**

Алгоритми дозволяють ефективно вирішувати задачі, а обробка виключень запобігає збоям програми.

**5. Типові помилки, яких припускаються початківці при написанні коду?**

* 1. **Синтаксичні**

Це можуть бути пропущені дужки, коми чи невірно написані ключові слова.

* 1. **Неправильні назви змінних**

Незрозумілі або неконсистентні назви ускладнюють розуміння коду.

* 1. **Ігнорування обробки помилок**

Без конструкцій try-catch програма може несподівано завершуватися через помилку.

* 1. **Копіювання без розуміння коду**

Часто початківці копіюють фрагменти коду, не розуміючи їхньої логіки.

* 1. **Відсутність тестування**

Недостатнє тестування призводить до того, що помилки виявляються лише після завершення розробки.

**Висновок**

Вивчення програмування є складним, але захоплюючим процесом, який вимагає розуміння основних концепцій та навичок. Інтегровані середовища розробки (IDE) спрощують написання, тестування та налагодження коду, але важливо правильно вибрати IDE залежно від мови та проекту. Основи програмування, такі як змінні, цикли та функції, є основою для розуміння коду. Початківці часто роблять типові помилки, але усвідомлення їх і уникнення покращує якість коду. Успіх у програмуванні залежить від постійної практики, навчання та здатності адаптуватися до нових викликів.