**УДК 004.4**

# ПРОБЛЕМИ ВИВЧЕННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Автор – Дар'я Цимбал[1], студ. гр. КН-21
Науковий керівник – ст. викл. каф. комп'ютерних наук, інформаційних технологій та прикладної математики Євген Плахтій[2]
[1]*dashatcimbal@gmail.com*, [2]*plakhtii.ev@gmail.com*
Придніпровська державна академія будівництва та архітектури

Об'єктно-орієнтоване програмування (далі ООП) є однією з найбільш впливових парадигм програмування. Воно широко використовується та майже кожен університет світу включає ООП у свої навчальні програми. Загалом, ООП – гарний інструмент для вивчення тих методів програмування, які вважаються потрібними для практичної роботи програміста. Однак, вивчення ООП вважається складнішим, ніж структурного програмування.

Раніше ООП вважалося предметом для просунутих і вивчалося на старших курсах ВУЗу або в магістратурі, але зараз майже кожен університет викладає ООП з перших років навчання. Основна причина цього – проблема переходу з однієї парадигми програмування на іншу. Навчитися програмувати в об'єктно-орієнтованому стилі здається дуже складно після процедурного стилю. У студентів не виникає багато труднощів з розумінням об'єктно-орієнтованих принципів, якщо вони зіткнуться з ними на початку. Тобто складно перейти на нову парадигму, а не власне ООП. Шлях до ООП за допомогою процедурного програмування є складним і довгим.

Багато людей використовують процедурне програмування як шлях до ООП. На це більш за все впливає мова C++. Це гібридна мова, яка підтримує процедурне («стиль C») програмування та ООП. Вона була розроблена як продовження мови C. Це призвело до непорозуміння: багато людей розглядають об'єктну орієнтацію як просто іншу мовну конструкцію, яку можна вивчити після головних структур, покажчиків та рекурсії [1]. Однак, об'єктна орієнтація – це основна парадигма, яка формує спосіб мислення, як перенести проблему на алгоритмічну модель навіть найпростіших програм [2].

Якщо необхідно, щоб студенти також програмували в процедурному стилі, то процедурна мова може бути представлена пізніше. Перехід від ООП до процедурного програмування набагато простіший.

Отже, вивчити ООП доволі просто, але все одно виникають труднощі. Можливо, причина всіх проблем полягає в тому, що використовуються

неправильні мови програмування та середовища. Мова програмування, яка погана в одному контексті, може бути чудовою в іншому (або навпаки). Обрана мова програмування для початківців повинна відповідати наступним критеріям:

1. Зрозумілі поняття – тобто ті, які потрібно вивчити, повинні бути представлені мовою в чистому, послідовному та легко зрозумілому вигляді. Мова впровадження повинна відображати рівень абстракції, який ми хочемо використовувати для наших концептуальних моделей.

2. Чиста об'єктна орієнтація. Під «чистою об'єктною орієнтацією» мається на увазі, що мова не повинна бути «гібридною» мовою, тобто такою, яка підтримує як об'єктну орієнтованість, так і процедурне програмування. Це може заплутати й ускладнити вивчення конкретної парадигми.

Зауважимо, що багато людей стверджували, що гібридний характер мови може полегшити шлях до ООП для студентів з попереднім досвідом програмування. Якщо студент вже знає С, така гібридна мова, як С++, може забезпечити простий шлях входу. Об'єктна орієнтація може бути поступово додана до існуючого обсягу знань, полегшуючи розуміння проблем [2; 3].

3. Високорівневість мови. Слід обрати мову, яка більш зрозуміла людині і в якій не потрібно власноруч виконувати дії, з якими легко впорається сама машина.

4. Читабельний синтаксис. Використовуваний синтаксис, повинен бути легко читаним та послідовним. Читабельна програма, вірогідно, буде правильною.

5. Відсутність надмірності – означає, що для всього, що ми хочемо зробити мовою, має бути один і лише один спосіб зробити це.

6. Малий обсяг мови. Мова повинна бути максимально невеликою, включаючи всі важливі особливості, які можна обговорити в курсі програмування. Потрібно менше часу на засвоєння і зменшується ризик заплутатись у поняттях.

Також сформуємо вимоги до навчального середовища:

1. Простота використання. Одним з найважливіших факторів у вирішенні придатності середовища розробки програмного забезпечення є простота використання. Навколишнє середовище повинно бути досить простим, щоб його використовували недосвідчені студенти. Це практично означає, що середовище повинно мати графічний інтерфейс користувача. Завдання, якими потрібно керувати середовищем внутрішньо, є досить складними: управління файлами, редагування, компіляція, налагодження, перегляд, тестування, виконання тощо. Тому простота є важливим критерієм.

2. Інтегровані інструменти. Вимога інтегрованих інструментів є прямим результатом вимоги простоти використання. Інтеграція інструментів може мати багато переваг: уніфікований та менший інтерфейс, підвищення продуктивності, краща функціональність тощо.

3. Доступність. Оскільки мова йде про навчання початківців, середовище розробки повинно бути доступним: безкоштовним, або за розумну ціну, потребувати оптимальні технічні характеристики, кількість пам'яті тощо.

Ми прийшли до висновку, що об'єктно-орієнтований підхід в програмуванні – це потужний і цінний інструмент навчання, але проблеми у вивченні ООП мають різну природу та різні масштаби для кожної з мов. Присутні труднощі в переході на об'єктно-орієнтовану парадигму від процедурного підходу. Обравши правильну мову і середовище програмування можна значно полегшити освоєння об'єктно-орієнтованої парадигми. Фактично, вимоги до мови і до середовища одні й ті самі: вони повинні легко представляти базові поняття та інструментарій, бути читабельними, простими та доступними у використанні, при цьому бажано мати невеликий обсяг для вивчення.

## Список використаних джерел

1. Ментинський С. М., Пелех Я. М. Основи програмування на C++ : навч. посіб. з курсу «Основи інформатики і програмування». Ч. 2. Львів : Галицька Видавнича Спілка, 2021. 256 с.

2. Трофименко О. Г., Прокоп Ю. В., Швайко І. Г., Буката Л. М. та ін. C++. Основи програмування. Теорія та практика : підруч. За ред. О. Г. Трофименко. Одеса : Фенікс, 2010. 544 с.

3. Stroustrup B. The Design and Evolution of C++, Addison-Wesley, Reading, MA. 1994.

4. Biddle R., Tempero E. Teaching C++ – Experience at Victoria University of Wellington. University of Wellington, Technical Report CS-TR94/18. 1994.