Список використаних джерел:

1. Laudon K. C., Laudon J. P. Management Information Systems. New York: Pearson, 2021. 560 p.
2. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2021. 816 p.
3. IBM Corporation. COBOL Systems and Business Applications. URL: https://www.ibm.com (дата звернення: 15.03.2026).
4. ENISA. Threat Landscape 2023. Luxembourg: Publications Office of the European Union, 2023. 120 p.
5. Sebesta R. W. Concepts of Programming Languages. 12th ed. Boston: Pearson, 2022. 720 p.

УДК 004.42

МЕХАНІЗМИ УНАОЧНЕННЯ АЛГОРИТМІВ ТА ЕФЕКТИВНІ СПОСОБИ ЇХ ВІЗУАЛІЗАЦІЇ ЗАСОБАМИ ВЕБТЕХНОЛОГІЙ

Близнюк В. П.
iterbleiker1@gmail.com
Черкаський державний фаховий бізнес-коледж
Марченко С. В.
м.Черкаси, Україна

Унаочнення алгоритмів використовується як засіб підтримки розуміння обчислювальних процесів, налагодження програм і дослідження поведінки структур даних. На відміну від статичних описів алгоритмів, інтерактивні візуальні представлення відображають зміну станів під час виконання, що може бути корисним як у навчальних, так і в інженерних задачах. Сучасні вебзастосунки дають змогу поєднати в одному середовищі код, візуальну модель, історію виконання та керування сценаріями взаємодії. Автори [1] вказують, що корпус емпіричних досліджень у цій галузі все ще є відносно обмеженим, тобто наявність відкритих питань щодо того, які саме механізми унаочнення є справді результативними.

Проаналізувати сучасні підходи до унаочнення алгоритмів і визначити механізми візуалізації, які в умовах вебсередовища мають найбільшу практичну цінність для розуміння, налагодження та дослідження поведінки алгоритмів.

У сучасних оглядових роботах візуалізацію програм розглядають уже не як окремий навчальний прийом, а як багатошаровий засіб аналізу програмної поведінки. Зокрема, в огляді [2] подано класифікацію за етапами виконання: до запуску (для аналізу структури коду й потенційних помилок), під час виконання (для розуміння алгоритму, налагодження та моніторингу станів), після виконання (для оцінювання продуктивності, оптимізації та виявлення аномалій). Така рамка є важливою, оскільки зміщує фокус із привабливої анімації на інструментальну корисність візуального подання.

Лише факт наявності візуалізації ще не гарантує кращого розуміння алгоритму. Найбільш переконливі результати з'являються там, де візуальне подання поєднане з керуванням виконанням, відображенням проміжних станів і можливістю перевіряти власні припущення користувача.

Перший механізм – покрокове трасування з явним поданням станів. Його суть полягає в фіксації стану даних, керівних переходів і наслідків кожної операції. Такий підхід зберігає значення для класичних алгоритмів, але в сучасних системах він дедалі частіше доповнюється історією виконання та поверненням до попередніх кроків. Це переводить візуалізацію з демонстраційного режиму в режим дослідження. Огляд [2] прямо відносить такі сценарії до середньої фази виконання, де візуалізація підтримує розуміння, налагодження і моніторинг.

Другий механізм – живе програмування (live programming), тобто безперервне оновлення візуального подання під час редагування та виконання коду. У праці [3] на прикладі середовища Algot досліджено, наскільки такий режим допомагає у розумінні програм. Автори показали, що ефект не є універсальним, проте для задач, пов'язаних із матрицями та деревами, зафіксовано кращі результати; крім того, учасники дещо вище оцінили зручність

і задоволення від роботи із середовищем. Цей кейс демонструє вибіркову ефективність такого підходу залежно від типу задачі.

Автори [4] дійшли висновку, що безперервне відображення поточних значень може зменшувати когнітивне навантаження під час валідації та послаблювати як надмірну довіру, так і надмірну недовіру до підказок ШІ. Для теми унаочнення алгоритмів це показово, оскільки візуалізація тут працює не лише як навчальний засіб, а як інструмент інженерної перевірки рішень.

Третій механізм – візуалізація власного коду користувача, а не лише заздалегідь підготовлених прикладів. У системі PVLS блок-схеми та подання процесу виконання динамічно генерувалися зі студентського коду. За результатами дослідження [5] зафіксовано статистично значуще покращення загального результату та власне програмної успішності, але не отримано значущого приросту для розуміння процесу виконання як окремого показника. Отже, такий механізм виглядає перспективним для підтримки написання й налагодження коду, проте не варто беззастережно ототожнювати його з глибшим алгоритмічним розумінням.

Четвертий механізм – поєднання візуалізації з діалоговим або пояснювальним супроводом. У платформі VisualCodeMOOC динамічні візуалізації інтегровано з розмовним агентом VisualCodeChat і персоналізованим зворотним зв’язком. Автори [6] позиціонують систему як цілісне середовище активного навчання, де візуалізація не ізольована, а працює разом з поясненням, вправами й навігацією за завданням.

Окремої уваги заслуговує розширення цієї тематики на складніші алгоритмічні системи. Наприклад, у контексті машинного навчання потреба в унаочненні виникає не лише після побудови моделі, а й на різних етапах конвеєра аналізу для розуміння, діагностики й удосконалення моделей [7].

Разом із тим, новітні підходи не знімають низку обмежень. По-перше, інтерактивність ускладнює інтерфейс і створює ризик когнітивного перевантаження. По-друге, позитивний ефект часто виявляється локальним для певного типу задач або певної аудиторії, як це видно з кейсів Algot [3] і PVLS [5].

По-третє, навіть у найбільш сучасних роботах значна частина рішень залишається орієнтованою на навчальні сценарії, тоді як інтеграція з реальними середовищами розробки і системами налагодження ще не стала загальним стандартом.

У вебсередовищі реалізація таких механізмів унаочнення пов’язана з використанням клієнтських технологій відображення та керування станом виконання. Зокрема, застосовуються векторні графічні формати SVG і Canvas-відтворення для динамічного оновлення структури даних, а також механізми реактивного програмування, що забезпечують синхронізацію між змінами стану алгоритму і візуальним поданням. Для організації покрокового виконання використовуються подієві моделі браузера, асинхронні виклики та часові контролери анімацій. Крім того, обчислювально інтенсивні фрагменти алгоритмів можуть переноситися у WebAssembly-модулі, що дає змогу поєднати інтерактивність вебінтерфейсу з достатньою продуктивністю виконання.

Сучасний стан досліджень дає підстави розглядати унаочнення алгоритмів не як сукупність окремих анімацій, а як клас інтерактивних механізмів аналізу виконання. Для вебсередовища найбільш перспективними виявляються: покрокове трасування зі збереженням історії станів, живе оновлення подання під час роботи з кодом, побудова візуального подання на основі власного коду користувача, а також поєднання візуалізації з пояснювальними та діалоговими засобами. Водночас емпірична база ще не є достатньо щільною, а результати окремих кейсів свідчать, що ефективність залежить від типу задачі, форми взаємодії та цілей застосування. Саме тому подальші дослідження доцільно спрямувати на порівняння механізмів унаочнення в однакових умовах, а також на їх інтеграцію з інструментами налагодження та аналізу програмної поведінки.

Список використаних джерел:

1. Liu J., Poulsen S., Goodwin E., Chen H., Williams G., Gertner Y., Franklin D. Teaching Algorithm Design: A Literature Review // ACM Transactions on

Computing Education. 2025. Vol. 25. Issue 2. Article No. 17 P. 1-20. DOI: 10.1145/3727987.

2. Zhang W., Wen Z., Pan J., Chen W. Visualization for Computer Program: A Survey // Journal of Computer-Aided Design & Computer Graphics. 2023. Vol. 35, No. 8. P. 1139–1149. DOI: 10.3724/SP.J.1089.2023.19893.
3. Graf O., Thorgeirsson S., Su Z. Assessing Live Programming for Program Comprehension // Proceedings of the 2024 on Innovation and Technology in Computer Science Education. Vol. 1. 2024. P. 520–526. DOI: 10.1145/3649217.3653547.
4. Ferdowsi K., Huang R., James M. B., Polikarpova N., Lerner S. Validating AI-Generated Code with Live Programming // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024. Article No. 143. P. 1–8. DOI: 10.1145/3613904.3642495.
5. Lai C.-H., Lin P.-W., You S.-H. Development and Evaluation of a Dynamic Code Visualization System for C Programming Education: The PVLS Approach // Social Sciences & Humanities Open. 2025. Vol. 12. Art. 101962. DOI: 10.1016/j.ssaho.2025.101962.
6. Li M., Wang D., Purwanto E., Selig T., Zhang Q., Liang H.-N. VisualCodeMOOC: A Course Platform for Algorithms and Data Structures Integrating a Conversational Agent for Enhanced Learning Through Dynamic Visualizations // SoftwareX. 2025. Vol. 30. Art. 102072. DOI: 10.1016/j.softx.2025.102072.
7. Wang J., Liu S., Zhang W. Visual Analytics for Machine Learning: A Data Perspective Survey // IEEE Transactions on Visualization and Computer Graphics. 2024. Vol. 30, No. 12. P. 7637–7656. DOI: 10.1109/TVCG.2024.3357065.