2023 р. / за ред. М. С. Курача, І. В.Цісарук. Кременець: ВЦ КОГПА ім. Тараса Шевченка, 2023. 106-108 с.
3. Рак В. І. Особливості навчання студентів технологіям розробки цифрового освітнього відеоконтенту / В. І. Рак, М. Ю. Франко // Актуальні проблеми та перспективи технологічної і професійної освіти : матеріали VII всеукраїнської науково-практичної інтернет-конференції (м. Тернопіль, 20-21 квітня 2023 р.). – Тернопіль : ТНПУ ім. В. Гнатюка, 2023. – С. 68-69.

Юркевич Іван
Науковий керівник – доц. Ящик Олександр

ОСНОВНІ ПОНЯТТЯ АЛГОРИТМІЗАЦІЇ ТА ОСНОВ ПРОГРАМУВАННЯ

Алгоритмізація та основи програмування – це фундаментальні поняття в інформатиці, які лежать в основі створення програмного забезпечення. Алгоритмізація – це процес розробки чіткого і зрозумілого плану дій (алгоритму) для розв’язання певної задачі. Алгоритм повинен бути скінченним, точним, результативним і мати визначену послідовність кроків. У програмуванні алгоритми реалізуються мовами програмування, які дозволяють перетворити ці плани дій у форму, зрозумілу комп’ютеру. Основи програмування включають поняття змінних, типів даних, операторів, умовних конструкцій, циклів, функцій, масивів та структур даних. Важливим є розуміння логіки виконання програми, обробки помилок та способів зберігання інформації. Програміст повинен не лише знати синтаксис мови програмування, але й вміти аналізувати задачу, формулювати алгоритм її вирішення, а також оптимізувати його для досягнення найкращої ефективності. У результаті алгоритмізація та програмування тісно пов’язані між собою й утворюють основу для створення ефективних і надійних програмних продуктів. Основні поняття алгоритмізації охоплюють логічні та теоретичні основи побудови алгоритмів – чітких і зрозумілих інструкцій для розв’язання задач. Алгоритмізація є першим етапом у створенні програми, оскільки будь-яка задача, перш ніж бути реалізованою мовою програмування, повинна бути описана як послідовність логічних кроків.

Одним із головних понять є алгоритм – це скінченна послідовність дій, спрямованих на досягнення результату. Для того щоб алгоритм був правильним, він повинен мати такі властивості: визначеність (тобто кожен його крок має бути однозначно сформульований), результативність (він повинен приводити до кінцевого результату), скінченність (кількість кроків має бути обмеженою), масовість (можливість застосовувати до різних вхідних даних одного типу). Алгоритми можна описувати по-різному: у вигляді словесного опису, блок-схем, псевдокоду, або ж навіть за допомогою таблиць. Незалежно від способу подання, всі алгоритми мають будову на основі базових структур: лінійна (послідовне виконання дій), розгалуження (виконання дій залежно від умови) та циклічна структура (повторення певних дій до виконання умови). Також важливим поняттям є вхідні та вихідні дані – це ті значення, які алгоритм отримує на вході для обробки, і ті, які він повертає після виконання. Ще одне ключове поняття – операторна структура, що означає послідовність окремих команд або дій, які виконуються у визначеному порядку. У процесі алгоритмізації важливо враховувати не лише правильність

алгоритму, але й його ефективність – тобто, наскільки швидко і з якою кількістю ресурсів він виконує свою задачу. У більш складних задачах також враховується оптимізація – вибір найкращого шляху розв'язання з можливих.

Алгоритмізація не залежить від мови програмування – це загальний логічний підхід до вирішення задач, який потім уже реалізується за допомогою програмного коду. Саме тому вона є базовим етапом у вивченні інформатики та програмування.

Основи програмування – це сукупність понять, принципів і навичок, які потрібні для створення програм, тобто інструкцій для комп'ютера, що дозволяють автоматизувати виконання різних задач. На відміну від алгоритмізації, яка зосереджена на логіці дій, програмування стосується їх технічного втілення за допомогою конкретної мови програмування. У програмуванні центральну роль відіграють змінні – іменовані області пам'яті, в яких зберігаються значення. Кожна змінна має тип даних, що визначає, який вид інформації в ній може зберігатися: наприклад, цілі числа, дійсні числа, символи, рядки або логічні значення. Це дозволяє комп'ютеру ефективно працювати з даними, виділяючи потрібну кількість пам'яті та виконуючи відповідні операції. Керування виконанням програми відбувається за допомогою операторів – окремих інструкцій, які можуть змінювати значення змінних, виконувати арифметичні обчислення, здійснювати перевірку умов чи викликати інші частини програми. Важливим елементом є умовні конструкції, які дозволяють виконувати різні дії залежно від результату логічного виразу – наприклад, конструкції if...else. Ще одним ключовим елементом є цикли – інструменти, що дозволяють повторювати певні дії багаторазово, поки виконується умова (while, for). Це особливо важливо для обробки великих обсягів даних або виконання рутинних операцій. Програмування також передбачає використання функцій або процедур – окремих блоків коду, які можна багаторазово викликати. Це дозволяє структурувати програму, зробити її більш зрозумілою, логічною і повторно використовуваною. Велика програма зазвичай складається з багатьох функцій, що взаємодіють між собою. Ще одна важлива складова – масиви та структури даних, які дозволяють зберігати та обробляти велику кількість значень одночасно. Масив – це набір елементів одного типу, доступ до яких здійснюється за індексом. У більш складних випадках використовуються списки, множини, словники тощо. Програмування також вимагає розуміння ввідних і вивідних операцій, щоб програма могла отримувати дані від користувача та повертати результат. Розуміння відлагодження та тестування програм також є невід'ємною частиною – це процес пошуку і виправлення помилок у коді. Усе це вимагає логічного мислення, точності у формулюваннях і дисципліни в оформленні коду. Знання основ програмування є універсальним – вони застосовуються незалежно від конкретної мови, а перехід між різними мовами стає легшим після засвоєння цих базових понять.

Сучасні мови програмування – це інструменти, за допомогою яких розробники створюють програмне забезпечення для різних сфер: від вебсайтів і мобільних додатків до

систем штучного інтелекту та вбудованих пристроїв. Кожна мова має свої особливості, сфери застосування, сильні й слабкі сторони.

Однією з найпопулярніших мов у світі є Python. Її цінують за простий синтаксис, читабельність і потужні бібліотеки для роботи з даними, машинним навчанням та автоматизацією. Python часто обирають початківці, але також вона широко використовується в наукових дослідженнях, фінансах, кібербезпеці та штучному інтелекті.

JavaScript є основною мовою для створення динамічних веб-інтерфейсів. У поєднанні з HTML та CSS вона дозволяє будувати повноцінні клієнтські вебзастосунки. Завдяки середовищу Node.js JavaScript також активно використовується для розробки серверної логіки, що зробило її універсальною мовою повного стеку (full-stack development).

Java залишається популярною завдяки своїй стабільності, безпеці та кросплатформеності. Вона активно використовується у великих корпоративних системах, банківському секторі, а також для розробки Android-додатків.

C# є основною мовою для розробки на платформі Microsoft .NET. Вона широко застосовується у створенні бізнес-додатків, комп'ютерних ігор (зокрема в Unity) та вебсервісів.

C++ – потужна мова низького рівня, що дає великий контроль над ресурсами комп'ютера. Вона використовується у розробці ігор, операційних систем, драйверів та програм з високими вимогами до продуктивності.

Також стрімко набирають популярності новіші мови, як-от Go (Golang) від Google, яка відзначається простотою, швидкістю компіляції та високою продуктивністю, що робить її зручною для серверних застосунків. Ще одна сучасна мова – Rust, яка поєднує безпечне управління пам'яттю з високою швидкістю, тому її використовують для системного програмування, замінюючи C або C++.

Існують також мови, що спеціалізуються на певних галузях. Наприклад, Swift розроблена Apple для створення застосунків під iOS та macOS, а Kotlin – мова, яка набула великої популярності у середовищі Android-розробки. У сучасному світі програмістам часто доводиться знати кілька мов програмування, обираючи найвідповіднішу залежно від завдань, з якими вони працюють. Уміння аналізувати задачу та формулювати алгоритм її розв'язання є ключовим етапом у створенні будь-якої програми. Це не просто технічна навичка, а здатність мислити логічно, послідовно й структуровано. Саме від якості цього етапу залежить ефективність, простота та зрозумілість майбутнього коду.

Аналіз задачі починається з глибокого розуміння умови: що саме потрібно зробити, які є вхідні дані, які очікувані результати, які обмеження існують. Дуже важливо розрізнити основне й другорядне, побачити суть задачі без зайвих подробиць. Це вміння називається формалізацією – перетворенням опису проблеми з повсякденної мови на чіткі математичні або логічні поняття.

Наступний етап – це пошук способу розв'язання, тобто побудова логіки дій, що приведуть до результату. Тут розробник має мислити, як "машина": що потрібно зробити в

першу чергу, які дії виконати наступними, як поводитися з різними варіантами вхідних даних. Якщо задача складна, її розбивають на дрібніші підзадачі, кожну з яких легше зрозуміти та розв'язати окремо.

Формулювання алгоритму – це логічна схема дій, що має початок, послідовність переходів і завершення. На цьому етапі вже можна уявити, як виглядатиме рішення у вигляді блок-схеми, псевдокоду або простого опису кроків. Важливо врахувати всі можливі ситуації, включаючи граничні випадки (наприклад, порожній масив, нульові значення тощо). Хороший алгоритм повинен бути не лише правильним, а й ефективним, тобто виконуватись за мінімальну кількість кроків і не використовувати зайві ресурси.

Після формулювання алгоритму настає етап перевірки – подумки або на папері алгоритм "проганяється" на прикладах, щоб пересвідчитись, що він дійсно працює правильно. Це допомагає виявити логічні помилки ще до того, як почнеться програмування. Таким чином, вміння аналізувати задачу і формулювати алгоритм – це основа для створення надійного програмного рішення. Воно вимагає терпіння, уважності до деталей і здатності мислити структуровано. І чим краще розробник справляється з цим етапом, тим простішим буде подальше програмування.

Отже, алгоритмізація та основи програмування є фундаментом для розв'язання задач за допомогою комп'ютера. Вони формують логічне мислення, вчать аналізувати умови, чітко формулювати кроки розв'язання та реалізовувати їх мовами програмування. Розуміння цих основ дозволяє не лише створювати ефективні програми, але й краще орієнтуватися в сучасному цифровому світі. Успішне застосування знань з алгоритмізації й програмування базується на вмінні аналізувати задачі, будувати правильні алгоритми, структурувати код та мислити системно.

ЛІТЕРАТУРА

1. Машина Тюрінга як універсальний виконавець алгоритмів та її застосування в процесі поглибленого вивчення алгоритмізації і основ програмування старшокласниками / О. Б. Ящик // Інформаційні технології і засоби навчання. Information Technologies and Learning Tools. – 2016. – № 2(52). [Електронний ресурс]. – Режим доступу : http://journal.iitta.gov.ua/index.php/itlt/article/view/1365.
2. Система задач як засіб забезпечення розвитку системно-логічного мислення старшокласників в процесі поглибленого вивчення алгоритмізації та основ програмування / О. Б. Ящик // Комп'ютерно-орієнтовані системи навчання : зб. наук. праць / відп. ред. М. І. Жалдак. – К. : НПУ ім. М. П. Драгоманова, 2015. – № 17(24). – С. 75–81.
3. Формування системно-логічного мислення у процесі навчання основ алгоритмізації та програмування з допомогою систем комп'ютерної математики / О. Б. Ящик // Інформаційні моделі, системи та технології : матеріали IV науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 15-16 травня 2014 р.). – Тернопіль : ТНТУ ім. І. Пулюя, 2014. – С. 39.