УДК [372.8+0049]:378.14
DOI: https://doi.org/10.24919/2308-4634.2024.299138

Сергій Шаров, *кандидат педагогічних наук, доцент,*
завідувач кафедри комп’ютерних наук
Таврійського державного агротехнологічного університету імені Дмитра Моторного
Ганна Чернова, *кандидат педагогічних наук, доцент,*
доцент кафедри вищої математики та інформатики
Харківського національного університету імені В. Н. Каразіна
Юрій Сіциліцин, *доктор філософії,*
старший викладач кафедри комп’ютерних наук
Таврійського державного агротехнологічного університету імені Дмитра Моторного

МЕТОДИЧНІ АСПЕКТИ ВИКЛАДАННЯ ДИСЦИПЛІНИ “АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ” У ВИЩІЙ ШКОЛІ

Стаття присвячена висвітленню структури дисципліни “Алгоритмізація та програмування”, огляду методичних підходів, що можуть бути використані під час її викладання. Зазначається, що викладання дисципліни має певні особливості, що пов’язані з формуванням у здобувачів освіти алгоритмічного стилю мислення, усвідомлення роботи програмних конструкцій та ін. Значна увага повинна приділятися практичним завданням, що пропонуються здобувачам освіти у межах окремих навчальних тем. При викладанні дисциплін з програмування необхідно заохочувати здобувачів освіти до активного та творчого виконання практичних завдань, підтримувати їх самостійність та бажання самовдосконалюватися. На лекційних та практичних заняттях доречно застосовувати засоби мультимедіа, що дозволять візуалізувати навчальний матеріал та зробити його більш зрозумілим.

***Ключові слова**: програмування; алгоритмізація; програміст; методичні підходи; навчання.*
Літ. 15.

Sergii Sharov, *Ph.D. (Pedagogy), Associate Professor,*
Head of the Department of Computer Science
Dmytro Motornyi Tavria State Agrotechnological University
Ganna Chernova, *Ph.D. (Pedagogy), Associate Professor,*
Associate Professor of the Higher Mathematics and Informatics Department of
V. N. Karazin Kharkiv National University
Yurii Sitsylitsyn, *Doctor of Sciences (Philosophy),*
Senior Lecturer of the Computer Science Department of
Dmytro Motornyi Tavria State Agrotechnological University

METHODOLOGICAL ASPECTS OF TEACHING THE DISCIPLINE “ALGORITHMIZATION AND PROGRAMMING” IN A HIGHER SCHOOL

The article is devoted to highlighting the structure of the discipline “Algorithmization and programming”, an overview of methodological approaches that can be used during its teaching. It is noted that the information society functioning has led to a steady demand for computer science specialists who possess the appropriate digital competencies. The discipline “Algorithmization and programming” can be considered a basic educational component in future software engineers’ training. The discipline’s teaching has certain features related to the formation of students’ algorithmic style of thinking, awareness of the software structures work, etc. The importance of choosing a programming language, which will be learned within the discipline and may affect the specialization of a future specialist, is emphasized. It has been found that various methodological approaches can be used to improve the quality of teaching programming disciplines. A great attention should be paid to the practical tasks offered to students within the limits of individual educational topics. They should be practic-oriented, gradually become more difficult as the program material is studied, provide various options for program implementation, and consider the level of training of the students of education. Students should be encouraged to solve practical tasks actively and creatively within teaching programming disciplines, supporting their independence and desire for self-improvement. For lectures and practical classes, it is appropriate to use multimedia tools that will allow visualization of the educational material and make it more understandable. In addition, alternative means of acquiring knowledge and forming professional programming competencies, such as mass open online courses, programming games, and others, can be used. In future work, it is planned to investigate the differentiated educational tasks possibilities and methods of their differentiation.

***Keywords**: programming; algorithmization; programmer; methodical approaches; training.*

Постановка проблеми. Функціонування інформаційного суспільства сформувало стійкий попит на фахівців в ІТ-галузі, які володіють відповідними цифровими компентностями. Особливо це стосується програмістів, які залежно від специфікації беруть

участь у розробці Desktop-додатків, вебсайтів, мобільних застосунків тощо.

На сьогодні освіта залишається єдиним способом отримання необхідних знань та формування потрібних компетентностей. Відповідно до цього, опанувати навички програмування можна через навчання за відповідною освітньою програмою у закладі вищої освіти, на курсах підвищення кваліфікації (очних або в онлайн-форматі), під час самоосвітньої діяльності тощо. Якщо брати до уваги навчання в університеті, то на початкових курсах здобувачі вищої освіти опановують дисципліну “Алгоритмізація та програмування”, що дає змогу сформувати алгоритмічний стиль мислення, оволодіти базовими програмними конструкціями, підготуватися до вивчення більш фахових дисциплін з програмування тощо.

Аналіз основних досліджень і публікацій. Важливість професії програміста у забезпеченні розвитку інформаційного суспільства привела до появи значної кількості досліджень, що стосуються змістового наповнення фахової підготовки ІТ-фахівців та якості її реалізації. У цьому контексті В. Круглик, З. Сейдаметова розробляли авторські системи підготовки майбутніх інженерів-програмістів. У роботах Т. Гончаренко досліджено умови підготовки майбутніх ІТ-фахівців. Підготовку програмістів з урахуванням компетентнісного підходу проаналізовано у працях А. Власюка, П. Грицюка та ін.

Для підготовки програмістів використовуються різноманітні середовища і методології, технологія віртуальної реальності (Т. Вакалюк, М. Сідорко), мультимедійні технології (Л. Матвійчук), тренінгові форми роботи (І. Крашеніннік), активні методи навчання (К. Кірей) тощо. У студіях Л. Феклістової (L. Feklistova), М. Леппа (M. Lepp), П. Луика (P. Luik) та інших вчених вивчено можливості масових відкритих онлайн-курсів для вивчення програмування. Водночас внаслідок актуальності дисципліни “Алгоритмізація та програмування” та її складності для програмістів-початківців важливості набувають методичні підходи до її викладання.

Метою статті є висвітлення структури та особливостей дисципліни “Алгоритмізація та програмування”, огляд методичних підходів, які можуть бути використані під час її викладання у закладі вищої освіти.

Виклад основного матеріалу. Підготовка ІТ-фахівців, зокрема інженерів-програмістів, має унікальні особливості, що характеризуються динамічністю та складністю. По-перше, внаслідок стрімкого технологічного розвитку відбувається швидка зміна технологій і парадигм програмування. Сталі знання застарілої мови або технології програмування можуть призвести до зниження конкурентоспроможності розробника програмного забезпечення [2, 113]. З іншого боку, потреба у кваліфікованих програмістах на ринку праці пояснюється складністю опанування навичок програмування на високому рівні. Це актуалізує потребу у постійному самовдосконаленні та накопиченні практичного досвіду, обізнаності у сучасних технологіях розробки програмного забезпечення.

На думку О. Кучерук, процес підготовки майбутніх інженерів-програмістів повинен бути максимально наближений до сучасних вимог ІТ-індустрії [5, 10]. При виборі мови програмування для вивчення доцільно орієнтуватися на ті інструментальні засоби, які використовуються на конкретному підприємстві або є найбільш популярними. Також вагомими чинниками вибору є наявність безкоштовної ліцензії, функціональні можливості у вигляді додаткових бібліотек, складність у вивченні [15, 1890], наявність технічного документації, допомога з боку найближчого оточення тощо. На нашу думку, найбільш перспективними для вивчення є такі популярні мови, як C(C++), Java, JavaScript, Python. Оскільки мова програмування C++ займає високі позиції у міжнародних рейтингах, а також є основою для вивчення значної кількості інших мов програмування, саме її ми обрали для вивчення у межах дисципліни “Алгоритмізація та програмування”.

Структурно дисципліна “Алгоритмізація та програмування” складається з двох змістових модулів та передбачає лекції, практичні знання, а також самостійну роботу. Упродовж вивчення першого блоку дисципліни здобувачі освіти на практиці оволодівають базовими конструкціями мови програмування C++, зокрема арифметичними операціями та виразами, умовними конструкціями (if …else, switch), циклічними конструкціями (while, do… while, for…). Також цей блок передбачає опанування теоретичних знань про змінні, константи, типи даних, конструкцію програми на C++, випадки використання операторів break і continue тощо.

Після оволодіння простішими програмними конструкціями, якими можна скористатися при створенні програм, здобувачам освіти в межах робочої програми дисципліни пропонується вивчити складні структури, які передбачають більш розвинутий рівень абстрактного та логічного мислення. Майбутні фахівці комп’ютерних наук вивчають одномірні та багатовимірні масиви, рядки, покажчики, функції і структури, роботу з файлами та графікою.

На нашу думку, найбільше труднощів під час вивчення мови програмування відчувають новачки.

На початкових етапах вони повинні вивчити правила синтаксису та семантику мови [13, 176], оволодіти відповідною парадигмою програмування, базовими програмними конструкціями тощо. Тому при складанні робочої програми дисципліни ми враховували системний та компетентнісний підходи, а також принцип переходу від простого до більш складного. Для цього визначили перелік компетентностей в межах освітньої програми, які знадобляться для розробки програмного забезпечення, побудували певну послідовність навчальних тем у межах дисципліни, визначили кількість часу на їх вивчення [9, 107], сформували масив практичних завдань, питань для самоперевірки тощо.

Доцільно зазначити, що вивчення програмування неможливо без практичного застосування набутих знань. Водночас накопичення практичного досвіду у програмуванні має бути ефективним, заснованим на різноманітності практичних завдань, що повинні бути практико-орієнтованими. У цьому випадку викладач повинен застосувати міжпредметні зв’язки та створювати практичні завдання, що мають стосунок до математики (арифметичні обчислення, будування графіків, табулювання функцій та ін.), економіки (наприклад, обчислення суми вкладу, дебету, кредиту та ін.), архітектури та будівництва (обчислення об’єму фундаменту, кількості цегли, площі будівлі та ін.), навчального процесу (формування списку здобувачів освіти, обчислення середнього балу та ін.), інших галузей діяльності людини.

Процес підготовки інженера-програміста повинен передбачати виконання практичних завдань, які поступово змінюють складність – від простішого до більш складного [6, 150]. На нашу думку, складні програмні конструкції неможливо вивчити без знань та практичних навичок простих базових конструкцій, зокрема циклів і умовних операторів. Цю послідовність можна забезпечити формуванням практичних завдань, що поступово ускладняються. Наприклад, спочатку здобувачам освіти пропонується обчислити математичну формулу та вивести результат на екран у консолі. Потім при вивченні циклів та умовних операторів їм доречно запропонувати завдання на табулювання функції з перевіркою існування. Під час вивчення графічних операторів умовою завдання на табулювання функції може бути відображення результатів у вигляді графіка. Якщо вивчається тема “Робота з файловими змінними”, то результат табулювання функції можна зберегти у текстовому файлі.

Важливо зауважити, що навчальні завдання повинні бути сформульовані чітко. Це стосується вхідних та вихідних даних, формату виведення інформації, способів розв’язування задачі тощо. Таким чином, здобувач освіти зможе скласти чіткий алгоритм розв’язку відповідно до вимог. А от детально прописувати у завданні послідовність реалізації алгоритму недоречно. Принаймні, це стосується здобувачів освіти, які вже мають певний досвід у програмуванні. Згідно з дослідженням Л. Феклістової та інших вчених, якщо здобувачам освіти пропонувати багато джерел інформації та детальні вказівки на виправлення виниклих помилок у програмному коді, то вони можуть зробити більше спроб на виконання завдань, що негативно вплине на час роботи та оцінку [12]. У такому випадку доречно застосувати диференційований підхід до створення завдань з програмування, коли більш досвідчений здобувач освіти зможе швидко виконати завдання, проявивши креативність. А менш досвідчений здобувач освіти скористається більш деталізованою інструкцією та теж виконає завдання [10, 152]. Також ми радимо для більш досвідчених здобувачів освіти складати навчальні завдання, що передбачають різні варіанти розв’язання проблеми.

Під час вивчення програмування здобувачам освіти можна запропонувати у межах самостійної роботи створити прототипи простих ігор, наприклад, “Вгадай число”, “Більше – Менше”, “Колесо Фортуни” тощо. Більш комплексний підхід до застосування набутих знань та сформованих навичок програмування можна реалізувати з використанням методу проєктів. У такому випадку завдання розподіляються між мікрогрупами, які повинні створити власний алгоритм рішення, розробити закінчену комп’ютерну програму, публічно захистити проєкт у присутності всієї групи та викладача [4, 142].

Ефективним підходом до вивчення програмування на початковому рівні є забезпечення інтерактивності та наочності. Частою перешкодою в опануванні навичок програмування є нерозуміння того, як поставлену проблему можна перетворити спочатку у відповідний алгоритм, а потім у програмний код на конкретній мові програмування. Як зазначається у роботі [11, 43], традиційні методи викладання, що передбачають оволодіння тільки теорією з програмування, досить часто є пасивними та не залучають здобувачів освіти до інтерактивної взаємодії із навчальним матеріалом. Як наслідок, початківці можуть допускати помилки у створенні комп’ютерних програм, що впливає на появу розчарувань та зниження у них мотивації на подальше навчання [14, 11]. Для подолання цієї проблеми радимо під час пояснення нового матеріалу демонструвати на дошці або екрані комп’ютера створення алгоритму поставленої задачі або розробку програмного коду на основі наявного алгоритму. Так усі здобувачі освіти мають змогу слідкувати за поступовим розв’язком задачі, після

чого готовий програмний код виконують в інтегрованому середовищі програмування або онлайн-компіляторі.

На важливості мультимедійних технологій під час викладання дисциплін інформатичного циклу наголошено у роботі Л. Матвійчук. На думку дослідниці, їх використання надасть можливість деталізувати інформацію у вигляді графічних зображень, графіків, діаграм, продемонструвати навчальне відео, поєднати аудіо- і відеосупровід та ін. Це допоможе розвинути у здобувачів освіти образне мислення та фантазію, змоделювати реальні процеси, зробити об’єкти для вивчення більш динамічними [7, 257]. На нашу думку, під час лекційних занять варто використовувати мультимедійні презентації, що відображають основні тези в межах навчальної теми, демонструють фрагменти програмного коду, синтаксис програмних конструкцій тощо. Якщо мати на увазі виконання практичних робіт, то доречно продемонструвати здобувачам освіти типові приклади відповідно до теми, а потім запропонувати виконати аналогічні практичні завдання.

З метою підвищення пізнавального інтересу та забезпечення інтерактивності навчання під час оволодіння навичками програмування можна використовувати принцип гейміфікації. У роботі М. Медведєвої звертається увага на таких ігрових онлайн-сервісах, як CodinGame (ігрова онлайн платформа для програмістів), CodeCombat (багатокористувацька гра для навчання програмуванню), CodeMonkey (гра для початківців у програмуванні), Codewars (інтерактивна онлайн платформа для навчання на різних мовах програмування) та ін. [8, 252].

У процесі підготовки майбутніх інженерів-програмістів К. Кірей пропонує використовувати принцип активного навчання, що передбачає розвиток фахових компетентностей через активну та керовану діяльність здобувачів освіти [3, 66]. У контексті вивчення програмування це можна реалізувати під час лекційних занять, коли в межах однієї лекції здобувач вищої освіти готує невелику доповідь про використання різних програмних конструкцій для розв’язання однієї задачі.

Під час вивчення дисципліни “Алгоритмізація та програмування” необхідно заохочувати здобувачів освіти до використання додаткових електронних ресурсів, що дають можливість отримати більше знань та сформувати додаткові фахові компетентності. Наприклад, під час самостійної роботи доречно пропонувати онлайн-курси, розміщені на платформах масових відкритих онлайн-курсів (МВОК). Для вивчення програмування існує значна кількість МВОК, які пропонують різноманітні онлайн-курси з урахуванням рівня підготовки слухача, вартості, мови програмування тощо. Так, станом на липень 2021 р. на платформі МВОК Alison були доступні 50 онлайн-курсів з програмування, на Edx – 53 онлайн-курси, платформа Codecademy пропонувала вивчити 117 онлайн-курсів. Найбільше (6759) курсів з програмування на різних мовах пропонувала МВОК Udemy [15, 1890].

Висновки. Отже, дисципліна “Алгоритмізація та програмування” може вважатися базовим освітнім компонентом у підготовці майбутніх інженерів-програмістів. Її викладання має певні особливості, що пов’язані з формуванням у здобувачів вищої освіти алгоритмічного стилю мислення, усвідомлення роботи програмних конструкцій та ін. Для підвищення пізнавального інтересу майбутніх фахівців щодо вдосконалення навичок програмування можна застосовувати різні методичні підходи. По-перше, практичні завдання, які їм пропонуються для виконання, мають бути практично-орієнтовані. По-друге, практичні завдання повинні поступово ускладнюватися у міру вивчення програмного матеріалу. По-третє, при поясненні та практичному опрацюванні навчального матеріалу доречно застосовувати засоби мультимедіа. При викладанні дисциплін з програмування важливо заохочувати здобувачів освіти до активного та творчого виконання практичних завдань. Додатково можна застосовувати альтернативні засоби отримання знань та формування фахових компетентностей з програмування через використання масових відкритих онлайн-курсів, ігор з програмування тощо.

ЛІТЕРАТУРА

1. Базурін В.М. Середовища програмування як засіб навчання учнів основ програмування. *Інформаційні технології і засоби навчання*. 2017. № 59 (3). С. 13–27.

2. Власюк А., Грицюк П. Підготовка фахівців з інформаційних технологій у контексті сучасних вимог. *Нова педагогічна думка*. 2013. № 1.1. С. 109–114.

3. Кірей К.О. Використання активних методів навчання у процесі фахової підготовки майбутніх інженерів-програмістів. *Наукові праці. Педагогіка.* 2019. № 314 (326). С. 64–67.

4. Кривонос О.М. Компетентнісно-орієнтовані завдання в курсі “Програмування”. *Науковий часопис Національного педагогічного університету імені М.П. Драгоманова*. 2014. Т. 5. № 47. С. 138–144.

5. Кучерук О.Я. Компетентнісний підхід у підготовці майбутніх інженерів-програмістів. *Міжнародний науковий журнал “Науковий огляд”*. 2014. № 2 (3). С. 1–10.

6. Лебедь Г.М. Освітня складова змісту фахової підготовки майбутніх програмістів. *Науковий вісник Ізмаїльського державного гуманітарного університету*. 2017. № 36. С. 149–152.

7. Матвійчук Л.А. Мультимедійні технології в підготовці майбутніх фахівців інженерів-програмістів. *Педагогічні науки: теорія, історія, інноваційні технології*. 2012. № 3 (21). С. 253–259.

8. Медведєва М.О., Жмурко О.І., Криворучко І.І., Ковтанюк М.С. Використання ігрових онлайн-сервісів у процесі вивчення мов програмування. *Актуальні питання гуманітарних наук*. 2021. Т. 2, № 36. С. 248–255.

9. Сіциліцин Ю.О. Формування змісту професійної підготовки майбутніх інженерів-програмістів у галузі паралельних обчислень. *Педагогічні науки: теорія та практика*. 2022. № 2 (42). С. 105–111.

10. Шаров С.В. Використання диференційованих навчальних завдань під час самостійної роботи студентів. *Педагогіка, психологія та медико-біологічні проблеми фізичного виховання і спорту*. 2011. № 3. С. 151–153.

11. Bhatti S., Dewani A., Maqbool S., Memo M.A. A Web based Approach for Teaching and Learning Programming Concepts at Middle School Level. *International Journal of Modern Education and Computer Science*. 2019. № 11(4). pp. 46–53.

12. Feklistova L., Lepp M., Luik P. Learners' performance in a MOOC on programming. *Education Sciences*. 2021. № 11 (9). pp. 2–18.

13. Malik S.I. et al. Enhancing problem-solving skills of novice programmers in an introductory programming course. *Computer Applications in Engineering Education*. 2022. № 30 (1). pp. 174–194.

14. Mironova O., Amitan I., Vilipõld J. Programming Basics for Beginners. Experience of the Institute of Informatics at Tallinn University of Technology. *International Journal of Engineering Pedagogy*. 2017. № 7 (4). p. 7–18.

15. Sharov S., Kolmakova V., Sharova T., Pavlenko A. Analysis of MOOC on Programming for IT Specialist Training. *TEM Journal*. 2021. № 4 (10). pp. 1884–1894.

REFERENCES

1. Bazurin, V.M. (2017). Seredovyshcha prohramuvannia yak zasib navchannia uchniv osnov prohramuvannia [Programming environments as a means of teaching pupils to programming basics]. *Information technologies and teaching tools*. No. 59 (3), pp. 13–27. [in Ukrainian].

2. Vlasiuk, A. & Hrytsiuk, P. (2013). Pidhotovka fakhivtsiv z informatsiinykh tekhnolohii u konteksti suchasnykh vymoh [Training of information technology specialists in the context of modern requirements]. *A new pedagogical thought*. No. 1.1, pp. 109–114. [in Ukrainian].

3. Kirey, K.O. (2019). Vykorystannia aktyvnykh metodiv navchannia u protsesi fakhovoi pidhotovky maibutnikh inzheneriv-prohramistiv [Applying active learning methods to software engineering education]. *Scientific works. Pedagogy*. No. 314 (326), pp. 64–67. [in Ukrainian].

4. Kryvonos, O.M. (2014). Kompetentnisno-oriientovani zavdannia v kursi “Prohramuvannia” [Competence and Directed Task in Course “Programming”]. *Scientific journal of the National Pedagogical University named after M. Drahomanov.* No. 5 (47), pp. 138–144. [in Ukrainian].

5. Kucheruk, O.Ya. (2014). Kompetentnisnyi pidkhid u pidhotovtsi maibutnikh inzheneriv prohramistiv [Competence approach in the training of future software engineers]. *International scientific journal “Scientific Review”.* No. 2 (3), pp. 1–10. [in Ukrainian].

6. Lebed, G.M. (2017). Osvitnia skladova zmistu fakhovoi pidhotovky maibutnikh prohramistiv [Educational component of content of professional preparation of future programmers]. *Scientific Bulletin of the Izmail State Humanitarian University*. No. 36, pp. 149–152. [in Ukrainian].

7. Matviichuk, L.A. (2012). Multymediini tekhnolohii v pidhotovtsi maibutnikh fakhivtsiv inzheneriv-prohramistiv [The multimedia technology in the preparation of future professionals of software engineers]. *Pedagogical sciences: theory, history, innovative technologies.* No. 3 (21), pp. 253–259. [in Ukrainian].

8. Medvedieva, M.O., Zhmurko, O.I., Kryvoruchko, I.I. & Kovtaniuk, M.S. (2021). Vykorystannia ihrovykh onlain-servisiv u protsesi vyvchennia mov prohramuvannia [Use of online game services in the study of programming languages]. *Topical Issues in the Humanities.* No. 2 (36), pp. 248–255. [in Ukrainian].

9. Sitsylitsyn, Yu. O. (2022). Formuvannia zmistu profesiinoi pidhotovky maibutnikh inzheneriv-prohramistiv u haluzi paralelnykh obchyslen [Formation of the content of professional training of future software engineers in the field of parallel computing]. *Pedagogical sciences: theory and practice.* No. 2 (42), pp. 105–111. [in Ukrainian].

10. Sharov, S.V. (2011). Vykorystannia dyferentsiiovanykh navchalnykh zavdan pid chas samostiinoi roboty studentiv [Use of the differentiated educational tasks during independent work of students]. *Pedagogy, psychology and medical and biological problems of physical education and sports.* No. 3, pp. 151–153. [in Ukrainian].

11. Bhatti, S., Dewani, A., Maqbool, S. & Memo, M.A. (2019). A Web based Approach for Teaching and Learning Programming Concepts at Middle School Level. *International Journal of Modern Education and Computer Science*. No. 11 (4), pp. 46–53. [in English].

12. Feklistova, L., Lepp, M. & Luik, P. (2021). Learners' performance in a MOOC on programming. *Education Sciences*. No. 11 (9), pp. 2–18. [in English].

13. Malik, S.I. et al. (2022). Enhancing problem-solving skills of novice programmers in an introductory programming course. *Computer Applications in Engineering Education*. No. 30 (1), pp. 174–194. [in English].

14. Mironova, O., Amitan, I. & Vilipõld, J. (2017). Programming Basics for Beginners. Experience of the Institute of Informatics at Tallinn University of Technology. *International Journal of Engineering Pedagogy*. No. 7(4), pp. 7–18. [in English].

15. Sharov, S., Kolmakova, V., Sharova, T. & Pavlenko, A. (2021). Analysis of MOOC on Programming for IT Specialist Training. *TEM Journal*. No. 4(10), pp. 1884–1894. [in English].

Стаття надійшла до редакції 26.02.2024