УДК 377.6:61
DOI:

Олеся Власій, *кандидат технічних наук, доцент, доцент кафедри математики та інформатики і методики навчання, Прикарпатського національного університету імені Василя Стефаника*
Іван Яремій, *доктор фізико-математичних наук, професор, заступник декана фізико-технічного факультету з наукової роботи, Прикарпатського національного університету імені Василя Стефаника*
Марія Винничук, *магістрантка спеціальності "Середня освіта (Інформатика)", Прикарпатського національного університету імені Василя Стефаника*

ПРОБЛЕМА ПОСЛІДОВНОСТІ ВИВЧЕННЯ ПРОГРАМУВАННЯ

У статті, розглядаючи проблему послідовності вивчення програмування, запропоновано MIT App Inventor як перехідну ланку між вивченням Scratch та мов програмування високого рівня (на сьогодні це програмне середовище практично не вивчається в Україні, хоча деякі спеціалізовані школи вже звернули на нього увагу). При переході до вивчення мов програмування високого рівня після блокового програмування запропоновано використовувати ігровий підхід до розробки програм, що забезпечить послідовність навчання та взаємозв'язок із вивченим раніше матеріалом.

Ключові слова: *алгоритмізація; блокове програмування; ігровий підхід; інформатика; мови програмування високого рівня; послідовність вивчення програмування; програмний продукт; школярі.*

Літ. 15.

Olesia Vlasii, *Ph.D.(Engineering), Associate Professor of the Mathematics and Informatics and Methods of Teaching Department, Vasyl Stefanyk Precarpathian National University*
Ivan Yaremiy, *Doctor of Science (Physics and Mathematics), Professor, Deputy Director for Scientific Work of Faculty of Physics and Technology, Vasyl Stefanyk Precarpathian National University*
Mariia Vynnychuk, *Master's Student, Specialty "Secondary education (Informatics)", Vasyl Stefanyk Precarpathian National University*

PROBLEM OF CONSISTENCY IN TEACHING CODING AT SCHOOLS

In the article, the problem of teaching coding at schools is considered and the importance of consistency in the process is established. Coding is a common practice in many countries around the world, but it is noted that the process of creating a software product would make students confused because they should understand the process in general and use knowledge from different subjects. The importance of adopting teaching methods and learning tools to different ages of students is highlighted. Coding should be studied gradually by considering similar tasks from simple to complex ones; by the way it is not worth passing at once to code writing. That is why in the initial stages of learning algorithmization and programming, it is recommended to use the Scratch programming environment. Scratch's philosophy is to encourage children to learn programming, so instead of complex commands, bright blocks are used here, which make up the scripts for the characters and others. With its attractive appearance, elementary school students can easily learn the basic concepts of programming – conditions, loops, variables, commands, events, arrays, functions, etc., and Scratch is an object-oriented program that teaches children to think objectively. For high school students, it is quite acceptable to learn high-level programming languages (Python, C ++, Java, etc.), where a text editor is used to write code. As a transitional link between learning Scratch and high-level programming languages, the MIT App Inventor visual development environment for Android applications is proposed, which is similar in philosophy to Scratch but has a wider functionality. So, using App Inventor is recommended as a next step in teaching coding after Scratch. Moreover, there are extensions that allow converting programs from block to text (in particular, to the Java programming language) or vice versa, which allows to immediately combine the study of block programming in the App Inventor and the study of object-oriented high-level Java programming language. Today, MIT App Inventor is practically not studied in Ukraine, although some specialized schools have already paid attention to this programming tool. In the transition to the study of high-level programming languages after block programming, it is proposed to use a game approach to program development, which will ensure the consistency of learning and the relationship with the previously studied material.

Keywords*: algorithmization; block programming; game approach; Informatics; high-level programming*

Постановка проблеми. Вивчення інформатики у школі давно вийшло за межі дисципліни "Інформатика", адже цифровізація стосується освіти загалом і кожного предмета, зокрема. Знання з інформатики необхідні як у повсякденному житті, так і

практично в будь-якій майбутній професійній діяльності випускника школи. Тому інформатика нині є одним зі шкільних предметів, якому надається особлива увага. Це підтверджується тим, що за кількістю годин, виділених на її вивчення, вона поступається лише українській мові та математиці.

Особливе місце в предметі "Інформатика" посідає вивчення теми "Алгоритми та програми". У навчальних програмах, затверджених Міністерством освіти та науки України для 2–4 класів (2016 р.), 5–9 класів (2017 р.) та 10–11 класів (2018 р.) вивчення програмування передбачено кожного року, починаючи з початкової школи. Вивчення програмування часто викликає труднощі в учнів. Адже для розуміння його сутності необхідний цілий комплекс вмінь та знань, серед яких – вміння логічно та критично мислити, розуміти причинно-наслідкові зв'язки, ґрунтовні знання з математики тощо. Особливої актуальності набуває переосмислення процесу навчання програмування школярів при впровадженні компетентнісного підходу до навчання, ідей STEAM-освіти та інтеграційних зв'язків між предметами. Адже саме програмування, а, краще сказати – процес розробки програмного продукту, вимагає знань з різних предметів, комплексного підходу до розв'язання задачі – від її формулювання до розв'язання та поширення. Тут варто звернути увагу на відомий заклик розробників Scratch: "Уяви, запрограмуй, поділись!". Уважаємо, що такий мотиваційний заклик доречний не тільки для скретчерів, але і при подальшому вивченні програмування. Таким чином, виникає потреба переосмислення підходів до вивчення програмування у школі, яке зараз має дещо хаотичний характер, адже різноманітність сучасних мов програмування призводить до постійної дискусії педагогів, ІТ-спеціалістів та методистів щодо доцільності вивчення тієї чи тієї мови у школі, а технічна база закладів освіти та рівень підготовки учителів можуть кардинально відрізнятися у освітніх закладах одного і того ж рівня.

Аналіз останніх досліджень і публікацій. Проблему вивчення програмування в закладах освіти досліджували багато дослідники. Так, різні аспекти професійної підготовки педагогічних кадрів для сучасної школи, зокрема особливості підготовки майбутніх учителів інформатики, висвітлили Н. Апатова, Н. Балик, В. Биков, Л. Білоусова, І. Булах, О. Гончарова, В. Дем'яненко, М. Жалдак, В. Лапінський, М. Лапчик, Н. Морзе, В. Осадчий, К. Осадча, В. Руденко, Є. Смірнова-Трибульська та ін.; проблеми компетентнісного підходу в процесі формування інформаційно-комунікаційної компетентності особистості проаналізували О. Білоус, О. Гриценчук, І. Іванюк, О. Кравчина, М. Лещенко, І. Малицька, О. Овчарук, Д. Рождественська, Н. Сороко, В. Ткаченко, М. Шиненко, А. Яцишин та ін. Однак поза увагою дослідників залишилися окремі аспекти комплексного підходу до навчання програмування в початковій та середній школі за умов цифровізації освіти.

Мета статті – висвітлити особливості вивчення школярами програмування, увиразнити послідовність цього процесу.

Виклад основного матеріалу. Навчання дітей шкільного віку програмування – поширена практика в багатьох країнах світу, але процес створення власного програмного продукту часто насторожує та лякає учнів, адже вони починають усвідомлювати всю складність вивчення тієї чи тієї мови програмування [7]. Програмування передбачає не тільки складання тексту самої програми, але й урахування різних проблемних моментів, прогнозування наслідків тощо. Для цього необхідно вміти правильно аналізувати та робити висновки. Що швидше дитина почне вивчати програмування, то швидше в неї формується логічне мислення [3]. І навпаки, уміння логічно мислити – необхідний фундамент для успішного вивчення програмування.

Головним аспектом якісного навчання є правильна мотивація. Мотивація – використання соціальних (розуміння соціальної значущості, потреби, орієнтація на майбутнє) і пізнавальних (інтереси, емоції, ідеали, прагнення) мотивів. Щоб зацікавити навчальним матеріалом усіх учнів, на початкових етапах варто використовувати емоційне навіювання, доведення та переконування. Потрібно постійно стимулювати інтереси, наводячи приклади успішних програмістів. Варто зазначити, що програмістом може стати будь-хто у будь-якому віці, наприклад, першим програмістом у світі була жінка – Ада Лавлейс.

Підтримка мотивації здійснюється шляхом ігрових методів навчання. Ігровий підхід до набуття нових знань прямо впливає на зацікавленість людини в процесі навчання, мотивує її продовжувати процес та формує задоволення від досягнутого. Він дає змогу істотно підвищити ефективність, у тому числі швидкість засвоєння та якість набутих знань [1]. Орієнтуючись на гасло "Програмування – це гра" та педагогічний досвід освітян, особливу увагу приділяють ігровим технологіям навчання. На думку С. Іщерякова, новачки починають програмувати з двох причин – навчання й розвага, причому в першому випадку

інтерес згасає через незрозумілість доцільності процесу (окрім як отримати оцінку за цей процес) і нерозуміння логіки алгоритмів, у другому випадку, де програмування розцінюється як гра, простежується зацікавленість і захоплення учнів програмуванням, які бачать результат розробки і діляться ним з іншими, що й зумовлює подальші успіхи [4]. Під час навчання важливими моментами є створення сприятливої атмосфери, стимулювання до висловлювань без побоювань допустити помилку, формувати прагнення учнів знаходити свій спосіб роботи, вчитися аналізувати та виявляти учнівську ініціативу, самостійність у самовираженні [3; 7].

При реалізації ігрового підходу при послідовному підході до вивчення програмування, необхідно правильно адаптувати методи та інструменти до різного віку здобувачів освіти. Наприклад, вивчення у старших класах мов програмування високого рівня (C++, Python, Java і т.п.), у процесі якого для написання коду використовується інтегроване середовище розробки, доречно було би розглянути у контексті матеріалу, вивченого раніше. І навпаки – при вивченні програмування у молодших та середніх класах – розглядати завдання, які можна було б зреалізувати згодом при вивченні "дорослих" мов програмування.

Якщо щодо вибору мови програмування, яку варто вивчати в старших класах, досі немає однозначного підходу і ведуться дискусії, то початок вивчення наскрізної лінії "Алгоритмізація та програмування" шкільного курсу інформатики у початковій школі починається з вивчення Scratch.

Scratch – середовище та інтерпретована візуальна мова програмування. Програма складається зі скриптів, які необхідно збирати із кольорових блоків. Філософія Scratch полягає в заохоченні дітей до вивчення програмування, тому замість складних команд тут яскраві блоки, з яких складаються скрипти для ігрових персонажів, центральним і впізнаваним серед яких є герой багатьох шкільних проєктів – Рудий кіт. Варто зауважити, що логіка використання класичних базових алгоритмічних структур при блоковому програмуванні залишається та сама, що і при використанні інших мов програмування для розв'язання аналогічних задач, просто початківець зосереджується не на вивченні лексики та синтаксису мови, а на самій логіці процесу розробки, що є визначальним у цьому процесі. Також у Scratch можемо знайти прототипи для використання основних об'єктів програмування (як, наприклад, змінна, подія, функція, масив і т. п.). У цьому середовищі можливо змішувати різні типи інформації, реалізуючи творчі ідеї. Є можливість внесення змін до програми навіть тоді, коли вона запущена, що дає змогу експериментувати з новими ідеями під час виконання завдання. У результаті виконання простих команд створюється складна модель, у якій взаємодіють безліч об'єктів, наділених різними властивостями [10].

Однією з головних концепцій мови Scratch є ідея творчого навчання 4P (Projects, Passion, Peers, and Play). Згідно з ним, діти працюють над проєктом від ідеї аж до самого завершення. Учнів потрібно заохочувати слідувати своїм захопленням, тому що так вони зможуть довше та наполегливіше стояти перед викликами, розв'язуючи певні завдання [2]. Також, згідно з цією ідеєю, навчання – це соціальний процес, а не індивідуальна справа, тому його потрібно здійснювати в групі однолітків. І останнє – в Scratch реалізується навчання через гру, у якій діти можуть експериментувати, ризикувати та пробувати щось нове [11].

Коли школярі працюють над своїми проєктами, вони рухаються по спіралі творчого навчання, тобто уявляють те, що хочуть зробити, створюють щось на основі своїх ідей, граються із власною розробкою, діляться ідеями та розробкою з іншими, розмірковують над власним досвідом, що змушує їх знову уявляти нові ідеї та розробки, тож всі етапи знову повторюються. Власне ця спіраль і реалізовується під вищезгаданим гаслом "Уяви, запрограмуй, поділись!".

Процес створення проєкту в Scratch є доступним для всіх, хто хоче навчитися програмувати. Завдяки привабливому вигляду, дітям легко засвоїти базові поняття програмування – умови, цикли, змінні, команди, події, масиви, функції тощо, до того ж Scratch є ще й об'єктноорієнтованим середовищем, що вчить дітей думати об'єктно [3; 4; 10; 11].

Основи програмування учні починають вивчати в другому класі, знайомляться з поняттями "команда", "виконавець". Продовжується ця тема в третьому класі вивченням поняттям "алгоритм". На цьому кроці учні вже створюють лінійні програми. У четвертому класі школярі мають уміти застосовувати у своїх програмах різні види алгоритмів: лінійний, розгалужений, циклічний. Після закінчення вивчення мови Scratch учитель може самостійно вибрати мову програмування, що викликає дискусії з-поміж освітян.

Наступним інструментом у процесі вивчення

інформатики може стати MIT App Inventor 2 [5; 8] – середовище візуальної розробки android-додатків, що вимагає від користувача мінімальних знань програмування. MIT App Inventor наслідує ту ж ідею, що і Scratch, а саме: програмування шляхом перетягування кольорових блоків. Різницею між цими двома середовищами є те, що в MIT App Inventor розробляються проєкти для мобільних телефонів на базі Android.

В Україні цей інструмент ще системно не вивчається, але все більше привертає увагу дослідників. Цікавими розробками в цьому напрямі є: "App Inventor 2: Create Your Own Android Apps" (David Wolber, Hal Abelson, Ellen Spertus, Liz Looney), "Android App Inventor for the Absolute Beginner" (Lakshmi Prayaga, Jeffrey Hawthorne, Alex Whiteside). Окрім них, можна виокремити також курси "Tutorials for App Inventor" (розміщений на сайті AI2) та "Developing Android Apps with App Inventor", створений Гонконгським університетом науки і технологій (розміщений на платформі Coursera).

Розробка мобільного додатка в MIT App Inventor відбувається у два етапи. Перший етап – проєктування інтерфейсу користувача "Як це буде виглядати", другий – програмування елементів програми "Як вони будуть себе вести". Ці два процеси реалізуються в окремих вікнах, іншими словами, це два різні режими роботи в середовищі з MIT App Inventor [5]. "Дизайнер" – режим, у якому створюється інтерфейс додатка. Цей режим використовують для вибору і розміщення різних елементів додатка: кнопок, написів, зображень та ін., які відображаються на екрані мобільного пристрою під час запуску програми. Режим "Блоки" використовується для програмування поведінки додатка і його елементів у процесі різних дій користувача. Завдяки простоті цього середовища, діти зможуть розробляти самостійно додатки вже з першого уроку.

MIT App Inventor може стати проміжним етапом між вивченням Scratch та мов програмування високого рівня, зокрема, Java. Наприклад, існує інструмент App Inventor Java Bridge, який перетворює проєкт середовища App Inventor у Java код. Цей інструмент є в розширеній версії App Inventor, яка має відмінність лише в одному пункті меню – Java Bridge.

Негативним моментом використання App Inventor в освітній діяльності є те, що частина учнів користуються виняткова iOS пристроями. За даними statcounter [14] (вересень 2021 р.), 85 % усіх смартфонів в Україні – це Android пристрої, тому може постати запитання: що ж робити решті 15 % користувачів? Тут на допомогу приходить інший інструмент. Дослідники розширили середовище App Inventor, додавши функціонал та блоки, утворивши нові онлайн-середовища. Яскравими прикладом є Thunkable [9], у якого набагато ширший функціонал, порівняно з його попередником.

У Thunkable розробляються додатки як для Android, так і для iOS. Це середовище може стати ще одним засобом продовження вивчення програмування після Scratch.

На початковому етапі переходу учнів до написання коду можна використати інструмент Blockly: Code [15]. Його особливістю є можливість перетворення програм із блокового вигляду на текстові або ж навпаки. Тобто учні спочатку створюють звичну для них програму через перетягування блоків, а тоді мають можливість переглянути вигляд цієї програми на таких мовах програмування, як JavaScript, Python, PHP, Lua, Dart, XML.

Для реалізації запропонованого авторами підходу можна розглянути мову Python, яка за останніх років набула великої популярності. За рейтингом TIOBE (2020 р.), ця мова посідає третє місце за популярністю. Її часто рекомендують для початківців, адже вона має багато переваг (простота коду, лаконічність – те, що іншими мовами програмування займе багато рядків коду, на Python може поміститись в одному рядку, швидка у вивченні, на Python можна розробляти програми для різних платформ (комп'ютерні, мобільні, веб та інші)). Також існують різні бібліотеки (turtle, graphics, tkinter, pygame), які дають можливість створювати цікаві та унікальні інтерфейси програм, що сприяє творчому розвитку учнів. Зокрема, Pygame – це набір модулів Python, призначених для написання відеоігор. Ця бібліотека уможливлює створювати повнофункціональні ігри та мультимедійні програми мовою Python. Бібліотека проста у використанні. Ігри можуть створювати як діти, так і дорослі. Pygame використовується у проєкті OLPC [12] і викладається на курсах для маленьких дітей та студентів коледжів.

Ще однією цікавою мовою програмування є мови С та С++. Вони посідають перше та четверте місце в рейтингу TIOBE відповідно. Саме цими мовами найчастіше пишуться ігри за рахунок швидкодії написаних на них програм. Мова С – потужна процедурна мова, але в цьому випадку не вистачає багатьох сучасних і корисних конструкцій, тому для вивчення зараз рекомендується саме мова С++. Зазначимо, що мова С++ є досить складною для сприйняття, але дає широке розуміння принципів програмування.

За її допомогою зручно вивчати основи програмування. Розглянемо плюси програмування:

- Швидкість. C++ є в топі найшвидших мов програмування, тобто код, написаний цією мовою, виконується набагато скоріше, аніж інші. Саме ця мова (разом із C) загалом використовується для написання ігор, найпопулярніші з них GTA5 та World of Tanks.

- Універсальність. З допомогою C++ можна писати програми для різних платформ та операційних систем. Більшість десктопних (для комп'ютерів) додатків розробляється на C++, Java і Python. Саме на C++ Adobe пише свої продукти Photoshop і Acrobat.

- Фундамент для навчання. Java, JavaScript, C# і ще багато інших популярних мов програмування базуються на принципах C++. Побутує думка, що якщо учень зміг вивчити C++, то і з будь-якою іншою мовою в нього не виникне труднощів.

Для візуалізації програм, написаних мовою C++, можна використовувати фреймворк Qt або "бібліотеку тупого художника" (txLib). Qt – один з найбільших інструментів розробки графічних додатків мовою C++. Середовище поділене на дві частини – розробка дизайну та програмування компонентів (і як тут не звернути увагу на аналогію з MIT App Inventor). Важливим фактором Qt є також кросплатформовість.

Також варто звернути увагу на ідеї, запропоновні при використанні "Бібліотеки тупого художника" (TX Lib) – бібліотека двовимірної графіки для Windows. Ідеєю TX Library є допомогти здійснити перші кроки в програмуванні на C++ та підштовхнути до творчості та самостійності. Принципи, які закладені в цю бібліотеку: зроби сам; подивись в Help, подивись, як зроблено, подивись, як можна зробити по-іншому, вийди за межі гри [6].

Окрім подібних середовищ програмувань, є також різноманітні середовища, які дають можливісь розробляти ігри та вивчати середовища одночасно. Серед таких – Blockly Games, CodeMonkey, CodeCombat, JavaRush, CODINGAME, Py.CheckiO.

Висновки та перспективи подальших досліджень. Процес створення власного програмного продукту дає можливість усвідомити необхідність та доцільність вивчення програмування та інтеграції знань з різних галузей. Програмування необхідно вивчати поступово – від простого до складного. Саме тому на початкових етапах вивчення алгоритмізації та програмування рекомендується використовувати середовище Scratch. Для реалізації послідовності у процесі вивчення програмування запропоновано використовувати ігровий підхід, який започатковується при вивченні Scratch, та правильно адаптувати методи і форми навчання до різного віку здобувачів освіти. Після вивчення Scratch варто звернути увагу на середовище MIT App Inventor, яке за своєю філософією подібне до Scratch, але має ширший функціонал. Наявність доповнень, які дають можливість перетворення програм із блокового вигляду та текстові (зокрема, на мову програмування Java) і навпаки, дає змогу поєднати одразу вивчення блокового програмування і вивчення мови програмування високого рівня. Також доцільно звернути увагу на можливість реалізації ігрового підходу при подальшому вивченні програмування на мовах Python, Java чи C++, які для цього мають широкі можливості, однак здебільшого вивчаються за класичним підходом. Таким чином, одним із шляхів забезпечення послідовності при вивченні програмування у школі є використання ігрового підходу в контексті розробки програмних продуктів, а також використання перехідної ланки, наприклад – MIT App Inventor, між вивченням Scratch та мов програмування високого рівня,

Перспективами подальших досліджень є проблема підготовки майбутніх учителів до навчання блокового програмування школярів.

ЛІТЕРАТУРА

1. Горєлов В. О., Саля Д. Гейміфікація навчання. Інформаційні технології та комп'ютерне моделювання: матеріали Міжнародної науково-практичної конференції, м. Івано-Франківськ, 23–28 травня 2017 р. Івано-Франківськ, 2017. С.136–139. URL: http://lib.pnu.edu.ua:8080/bitstream/123456789/4708/1/Goryelov.pdf

2. Дудка О. М., Власій О. О., Магомета Н. М. Реалізація компетентнісного підходу до вивчення програмування на Scratch. *Відкрите освітнє е-середовище сучасного університету*. 2018. Вип. 5. С. 88–96.

3. IT-Школа. Навіщо вчити дітей програмуванню? URL: http://itschool.ck.ua/navishho-vchyty-ditej-programuvannyu.html

4. Іщеряков С. Навчання програмуванню: завдання школи чи університету? Освітня політика. Портал громадських експертів. 2017 р. URL: http://osvita.ua/school/54063/

5. Ливенец М. А., Ярмахов Б. Б. Программирование мобильных приложений в MIT App Inventor. Практикум. 2016. 100 с. URL: http://www.mkpochtoi.ru/AppInventor_rus.pdf

6. Проектно-ориентированное программирование на C++. URL: http://projectlis.blogspot.com/p/1.html

7. Семеніхіна О. В., Руденко Ю. О. Проблеми навчання програмувати учнів старших класів та шляхи їх подолання. *Інформаційні технології і засоби навчання*. 2018. Т. 66, №4. С. 54–64.

8. App Inventor Java Bridge. URL: http://www.appinventor.org/jbridge.

9. Create Your Own Native Apps With No-Code. URL: https://thunkable.com

10. Gomez Zermeco, M. G., & Dayane Blanco M. (2020). Development of Significant Learning through Scratch Programming Logic of Secondary School Students. *The International Journal of Technologies in Learning,* 27 (2). pp. 21–36.

11. Introduction to Creative Learning. Learning Creative Learning. URL: https://lcl.media.mit.edu/

12. OLPC: Learning happens everywhere. URL: https://laptop.org/

13. P. Rose S, Habgood MPJ, & Jay T. (2020). Designing a Programming Game to Improve Children's Procedural Abstraction Skills in Scratch. *Journal of Educational Computing Research,* 58(7). pp. 1372–1411.

14. StatCounter: Mobile Operating System Market Share Ukraine. URL: https://gs.statcounter.com/os-market-share/mobile/ukraine

15. Try Blockly. URL: https://developers.google.com/blockly/

REFERENCES

1. Goryelov, V. O. & Sala, D. (2017). Heimifikatsiia navchannia [Gamification of Education]. *Proceedings of the Conference "Information technology and computer modeling", Ivano-Frankivsk, 23–28 May* 2017. Ivano-Frankivsk, pp. 136–139. Available at: http://lib.pnu.edu.ua:8080/bitstream/123456789/4708/1/Goryelov.pdf [in Ukrainian].

2. Dudka, O., Vlasii, O., & Mahometa, N. (2018). Realizatsiia kompetentnisnoho pidkhodu do vyvchennia prohramuvannia na Scratch [Implementation of the competence-based approach to learning programming on Scracth]. *Open educational e-environment of modern University*, No. 5, pp. 88–96. [in Ukrainian].

3. IT-School. Che. Navishcho vchyty ditei prohramuvanniu? [Why teach children programming? Available at: http://itschool.ck.ua/navishho-vchyty-ditej-programuvannyu.html [in Ukrainian].

4. Ishcheriakov, S. (2017). Navchannia prohramuvanniu: zavdannia shkoly chy universytetu? [Learning programming: the task of school or university?]. *Educational policy. Portal of public experts*. Available at: http://osvita.ua/school/54063/ (accessed 02 Oct. 2021). [in Ukrainian].

5. Livenets, M. A., & Yarmakhov, B. B. (2016). Programmirovanie mobilnykh prilozheniy v MIT App Inventor. Praktikum [Programming of mobile applications in MIT App Inventor. Workshop]. Available at: http://www.mkpochtoi.ru/AppInventor_rus.pdf [in Ukrainian].

6. Proektno-orientirovannoe programmirovanie na S++ [Project-oriented programming on C++]. Available at: http://projectlis.blogspot.com/p/1.html [in Russian].

7. Semenykhina, O. V., & Rudenko, Yu. O. (2018). Problemy navchannia prohramuvaty uchniv starshykh klasiv ta shliakhy yikh podolannia [Problems of educating to programming of students and way of their overcoming]. *Information Technologies and Learning Tools*. Vol. 66, No. 4. pp. 54–64. [in Ukrainian].

8. App Inventor Java Bridge. Available at: http://www.appinventor.org/jbridge. [in English].

9. Create Your Own Native Apps With No-Code. Available at: https://thunkable.com [in English].

10. Gomez Zermeco, M. G. & Dayane Blanco, M. (2020). Development of Significant Learning through Scratch Programming Logic of Secondary School Students. *The International Journal of Technologies in Learning*. 27 (2). pp. 21–36. [in English].

11. Introduction to Creative Learning. Learning Creative Learning. Available at: https://lcl.media.mit.edu/ [in English].

12. OLPC: Learning happens everywhere. Available at: https://laptop.org/ [in English].

13. Rose, P. S., & Habgood MPJ, Jay, T. (2020). Designing a Programming Game to Improve Children's Procedural Abstraction Skills in Scratch. *Journal of Educational Computing Research,* 58(7). pp. 1372–1411. [in English].

14. StatCounter: Mobile Operating System Market Share Ukraine. Available at: https://gs.statcounter.com/os-market-share/mobile/ukraine [in English].

15. Try Blockly. Available at: https://developers.google.com/blockly/ [in English].

Стаття надійшла до редакції 07.10.2021