УДК 373.5.016:004:[159.955:510.5]

# РОЗВИТОК АЛГОРИТМІЧНОГО МИСЛЕННЯ НА УРОКАХ ІНФОРМАТИКИ

**Пасічник Оксана Володимирівна,**
*учитель інформатики НВК «Школа-гімназія «Сихівська»,*
*oksanapas@yahoo.com.*

**Анотація.** У статті розглядається поняття обчислювального мислення (computational thinking) як складової шкільного курсу інформатики. Згідно з новими програмами, розділи, присвячені алгоритмізації та програмуванню, вивчаються, починаючи з другого класу. Оскільки для різних вікових груп зміст і методи навчання відрізняються, у статті описано низку матеріалів, котрі можна використати для формування обчислювального мислення в учнів 2–9 класів.

**Ключові слова:** обчислювальне мислення, програмування у середній школі, інформатика у школі, Code.Org.

Шкільний предмет інформатика, хоч і є одним із наймолодших, зазнає чи не найбільших змін протягом усього періоду свого існування. Якщо на початку впровадження інформатики у загальну середню освіту основна увага приділялась основам алгоритмізації та програмування, то з часом уроки було спрямовано на оволодіння навичками роботи з поширеними комп'ютерними програмами. Цей процес відбувався в усьому світі й супроводжувався зниженням ваги інформатики як окремого предмету, і паралельним широким застосуванням сучасних інформаційних технологій на уроках з інших галузей знань. У багатьох країнах предмет під назвою інформатика чи інформаційні технології взагалі вилучений із навчальних планів, оскільки цифрову грамотність учні опановують вдома і в повсякденному житті.

Проте саме через те, що світ стає все більш технологічно насиченим, виникає необхідність глибокого розуміння принципів роботи комп'ютерних систем і вміння обирати відповідні засоби для полегшення розв'язування задач і пошуку рішень. Часто відбувається корекція назви предмету: якщо ІКТ зосереджується на тому, як використовувати програмні засоби, то інформатика вивчає побудову цих засобів і принципи їх роботи. Останнім часом усе гучніше лунає думка про необхідність повернення вивчення основ алгоритмізації та програмування з метою розвитку так званого **обчислювального мислення** (computational thinking) [1] — підходу до розв'язування задач, що використовує інформатичні методи. В основі навчання лежить не вивчення комп'ютера, а вивчення того, як використати його можливості для розв'язання тих задач, які постають — це розуміння як потреб людини, так і перспектив і обмежень обчислювальної техніки. У результаті набуваються не рутинні навички, а фундаментальні уявлення, створюються математичні моделі, які взаємодіють з реальним світом. Дуже важливо, що набуті знання і навички мають практичну застосовність у повсякденному житті, і можуть бути пов'язані з різноманітними сферами: від медицини до мистецтва, від історичних досліджень до космічних польотів.

Складовими обчислювального мислення є декомпозиція, виявлення шаблонів, узагальнення й абстрагування та розробка алгоритмів [2]. Як бачимо, цей тип мислення відрізняється від алгоритмічного, котре по суті є його складовим. **Декомпозицією** є вміння розкладати задачу на елементи, оперуючи якими, можна чітко пояснити процес іншому виконавцеві або просто записати його для власного подальшого використання. Результатом часто є виявлення шаблонів і узагальнень, які дозволяють згодом розробити алгоритм. Наприклад, куштуючи нову страву, ми намагаємось впізнати знайомі інгредієнти, а число 256,37 розкладається на $2\cdot10^2+5\cdot10^1+6\cdot10^0+3\cdot10^{-1}+7\cdot10^{-2}$ тощо.

**Виявлення шаблонів** передбачає вміння знаходити тенденції, схожі елементи або відмінності, котрі дозволять робити прогнози. Наприклад, ми розпізнаємо шаблони або правила поведінки в класі чи в групі, коли вперше у неї потрапляємо, а досліджуючи площу прямокутників однакового периметра з'ясовуємо, що чим більшою є різниця між довжинами сторін, тим меншою є площа фігури.

**Узагальнення шаблонів й абстрагування** означає вміння відкидати відомості, які не є вагомими для розв'язання певної задачі, а також підсумовувати необхідну інформацію. Це дозволить подати ідею чи процес у загальних рисах, залучаючи змінні так, щоб можна було використати цю схему для розв'язання групи схожих задач. Наприклад, сторінка учнівського щоденника містить узагальнену і систематизовану інформацію про навчальні тижні, а за допомогою формули $(a+b)(a-b) = a^2-b^2$ можна розв'язати цілу низку прикладів із конкретними значеннями.

Останній етап — **розробка алгоритму** або вміння створювати покрокову стратегію для розв'язання задачі. Наприклад, це рецепт приготування страви або арифметичні операції для знаходження відсоткової частки величини $A$ в іншій $B$: 100 помножити на $A$ і поділити на $B$.

Перелічені навички обчислювального мислення можна розвивати різноманітними методами на уроках різних предметів. У процесі розв'язування задачі потрібно знати, на яких етапах доцільно застосувати обчислювальні пристрої і програми, як вони можуть полегшити роботу людини, як змусити комп'ютер працювати над шуканим рішенням. Уміння бачити проблему через

призму сучасних технічних засобів може допомогти її більш ефективному розв'язанню. Дехто вважає це сучасною грамотністю, адже ці навички є універсальними для найрізноманітніших галузей [3].

Інформатика є предметом, у якому навчання навичок обчислювального мислення є базовим, тож у багатьох країнах відбувається перегляд того, якою має бути шкільна інформатика. Так, у 2014 році нові навчальні програми запроваджують у **Великобританії**, передбачаючи вивчення предмету інформатика (computing) учнями, починаючи із 6-річного віку. Національна навчальна програма [4] має забезпечити для всіх учнів такі вміння:

- розуміння і застосування основних принципів і понять інформатики, у тому числі абстракції, логіки, алгоритмів та представлення даних;
- аналіз задач з обчислювальної точки зору, практичний досвід написання комп'ютерних програм для вирішення задач;
- оцінка і застосування інформаційних технологій, у тому числі нових і незнайомих, для аналітичного розв'язку задачі;
- відповідальне, компетентне, впевнене та творче використання інформаційних і комунікаційних технологій.

У **США** навчальні програми з інформатики [5] передбачають опанування п'яти основних напрямків, серед яких обчислювальний інтелект, навички співпраці, практична інформатика та програмування, комп'ютери та інші комунікаційні пристрої, розуміння цифрової спільноти та етики. Інформатика розглядається як наука про комп'ютери й алгоритмічні процеси, включаючи їх принципи, апаратну і програмну побудову, застосування і вплив на суспільство. У навчальних програмах розрізняються поняття ІТ-грамотності й володіння ІТ. Тоді як грамотність передбачає опанування нинішніх технологічних рішень, володіння ІТ означає, що людина може переходити на нові технології, опановувати їх, пристосовувати до власних потреб. Володіння ІТ тісно пов'язане з поняттям обчислювального інтелекту, тобто вмінням розглядати задачі так, щоб у їх розв'язанні міг допомогти комп'ютер, і включає в себе такі навички як абстракція, декомпозиція, рекурсія, аналіз даних, створення реальних і віртуальних об'єктів.

Як зазначено у навчальній програмі МОН **України** для 5–9 класів, метою навчання курсу «Інформатика» є формування і розвиток предметної ІКТ-компетентності та ключових компетентностей для реалізації творчого потенціалу учнів і їх соціалізації у суспільстві, що забезпечить готовність учнів до активної життєдіяльності в умовах інформаційного суспільства та їх спроможність стати не лише повноцінними його членами, а й творцями сучасного суспільства [6]. Аналіз змісту навчального матеріалу програми показує, що курс є здебільшого орієнтованим на опанування інформаційними технологіями.

Природньо, що зміст шкільної програми з інформатики не може охоплювати абсолютно всі її аспекти, але має знайомити учнів з усіма розділами цієї науки, яка включає теорію інформації і кодування, теорію обчислень, алгоритми та структури даних, теорію мов програмування, формальні методи, штучний інтелект, комп'ютерний дизайн й інженерію, графіку і візуалізацію, безпеку і криптографію, комп'ютерні мережі, паралельні й розподілені системи, бази даних, програмну інженерію тощо. Можна зауважити,

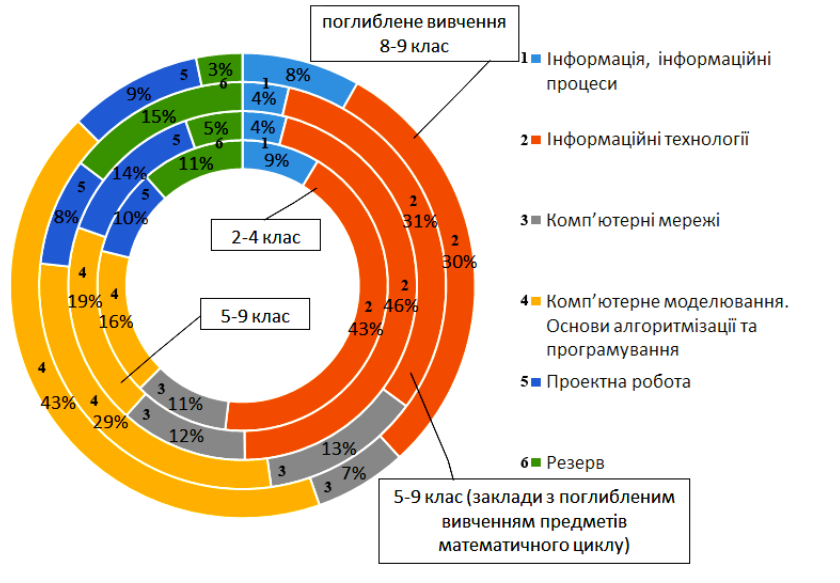



**Рис. 1. Зміст навчальних програм для 2–4 класів, 5–9 класів та 5–9 класів закладів з поглибленим вивченням предметів математичного циклу за новим Державним стандартом базової і повної загальної середньої освіти**

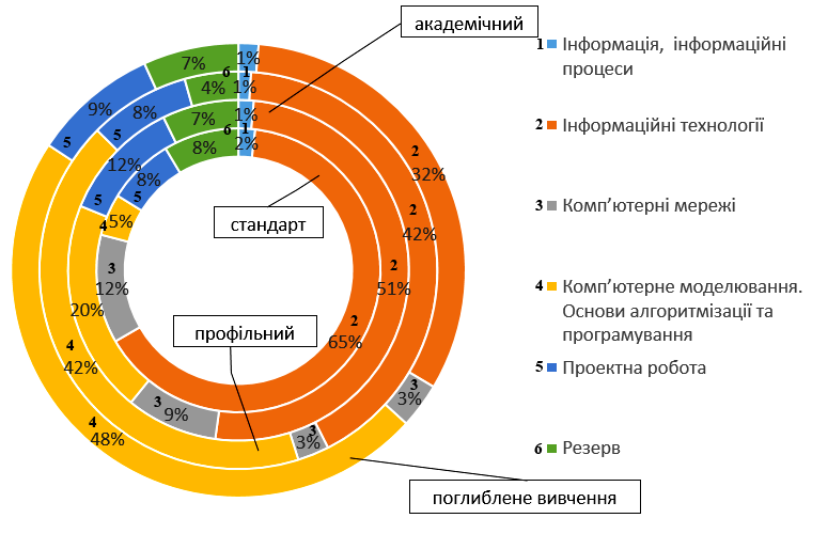



**Рис. 2. Зміст навчальних програм для 9–11 класів рівня стандарту, академічного, профільного та поглибленого вивчення інформатики для 8–11 класів**

що у цьому переліку немає того, що називається інформаційними технологіями в сенсі оволодіння прикладними програмами.

На рис. 1 і 2 наведено узагальнене порівняння часток годин, відведених на вивчення різних розділів у навчальних програмах, рекомендованих МОН України для загальноосвітніх навчальних закладів.

Можна стверджувати, що у чинних навчальних програмах з інформатики найбільше часу відводиться саме на **вивчення інформаційних технологій**, за винятком поглибленого і профільного вивчення інформатики (де більшу частку має тема алгоритмізації та програмування), котрі практикуються у дуже незначній кількості навчальних закладів країни.

Діаграми ілюструють деякі зміни за новим Державним стандартом освіти у співвідношенні часу на вивчення розділів, пов'язаних з **алгоритмізацією та програмуванням**, а також поняттям інформації, інформаційних процесів і систем. У контексті глобального перегляду змісту предмету інформатика у школі й ролі розділу з вивчення основ програмування за новими програмами, цікавим було обговорення стосовно того, яку мову програмування слід вивчати у школі, котре тривало у низці номерів журналу «Комп'ютер у школі та сім'ї» (№7, 8, 2013 р., №1, 2, 2014 р.) і дозволило фахівцям поділитись аргументами на користь того чи іншого середовища. Переважна більшість тих, хто висловився, підтримують думку про важливість різноманітності вибору, а також потребу в змінах підходу до викладання цієї теми. Практично усі дописувачі звертали увагу на необхідність вивчати не лише безпосередньо певну мову програмування, а скоріш принципи алгоритмізації, виховувати в учнів мислення, котре дозволяє формулювати задачу так, щоб до її розв'язання можна було залучити обчислювальні можливості сучасної комп'ютерної техніки. З огляду на те, що подібне бачення декларується у проекті **Code.Org** [7], розглянемо детальніше концепцію цієї ініціативи і запропоновані на ньому навчальні ресурси, які можуть бути використані на уроках інформатики у середній школі. Зазначимо, що матеріали проекту перекладені українською мовою, що уможливлює їх широке використання у школах (рис. 3).

Неурядова організація Code.Org створена у США у 2013 році з метою залучення учнів середніх шкіл до вивчення програмування. Розроблені відеоролики, інтерактивні уроки та інші матеріали дозволили провести достатньо успішну кампанію, у рамках якої майже **40 мільйонів учасників** з усього світу взяли участь у проекті «Година коду». До промоції заходу долучились велика кількість комп'ютерних фірм і відомих людей, котрі своїми зверненнями заохочують учнів до вивчення програмування: про свій досвід написання програм розповіли Біл Гейтс, Марк Цукенберг, Стів Джобс, Ештон Катчер, Шакіра та інші. Фахівці з компаній Google, Microsoft, Facebook, Twitter створили інтерактивні завдання на основі відомих ігор Angry-Birds та Plants vs. Zombies. Використання героїв знайомих комп'ютерних ігор як виконавців проклало своєрідний місточок між тим, щоб грати в гру і створювати її. До ініціативи долучився навіть Президент США Барак Обама, звернення якого розпочало «Годину коду» у грудні 2013 року: «Цього тижня я пишаюся приєднанням до учнів, викладачів, бізнесу та громадських організацій у здійсненні нових кроків з підтримки інформатики у школах Америки та всього світу. Навчання цих навичок не просто важливе для вашого майбутнього, це важливо для майбутнього вашої країни. Ось чому я прошу вас взяти участь. Не просто купити нову відеогру, а створити її. Не просто завантажити останній додаток, а допомогти розробити його. Не просто бавитись на телефоні, а запрограмувати його» [8].

Матеріали проекту «Година коду» і використані прийоми значно відрізняються від тих, що звикли використовувати на уроках інформатики. Вправи спроектовані так, що ознайомлення з основними алгоритмічними конструкціями відбувається у близькому до ігрового режимі **протягом 1 години** з використанням візуального середовища Blockly. На екрані бачимо сформульоване ігрове завдання, лабіринт із персонажами гри і команди (рис. 4), які вони вміють виконувати. Потрібно правильно зібрати блоки у робочій області і запустити програму. Якщо відповідь неправильна або неефективна, з'явиться відповідна підказка. Усі блоки візуального середовища можна переглянути в режимі коду, який нагадує команди мови Java Script, наприклад:

```
for(var count=0;
count<5; count++) {
 moveForward();
}
```

Кожен учень опрацьовує матеріал і виконує вправи **в автономному режимі**, роль вчителя зводиться лише до розв'язання технічних проблем і супроводу в особливо складних завданнях. Із кожним кроком завдання ускладнюються, і приблизно за першу годину роботи учень знайомиться з такими конструкціями: цикли з параметром, цикли з умовою (передумовою і післяумовою), повне та неповне розгалуження, які складаються у прості програми, метою котрих є проходження лабіринтів різної складності. Розв'язавши останнє завдання, учні формують код, за

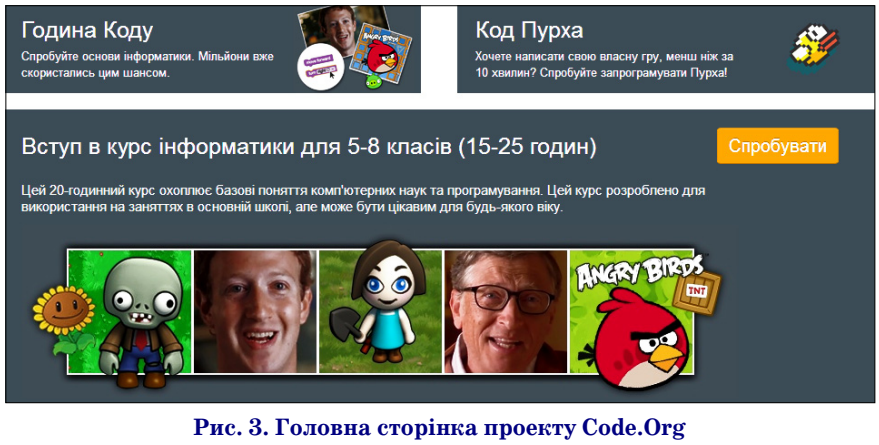



**Рис. 3. Головна сторінка проекту Code.Org**

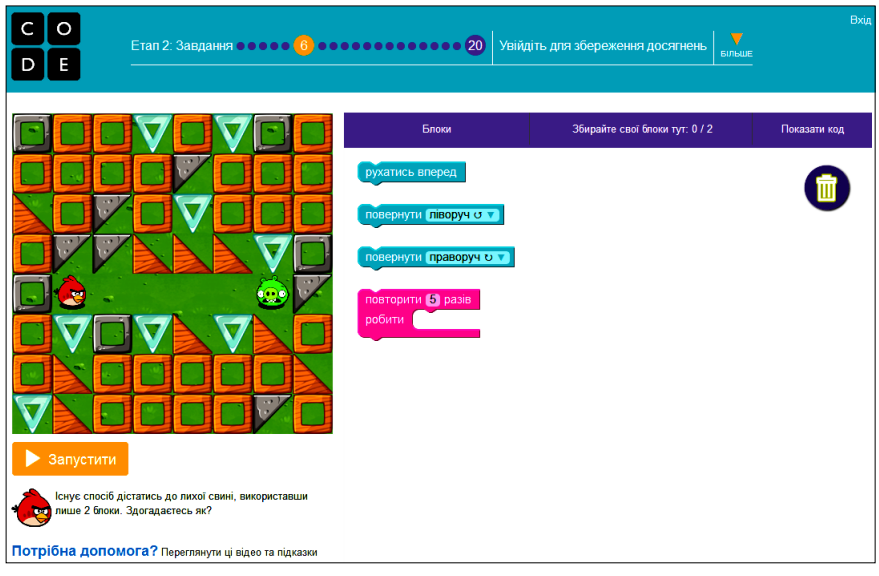



**Рис. 4. Одне із завдань Години коду у модифікованому візуальному середовищі Blockly**

яким робот (зомбі, птах, марсохід, автомобіль на автоматичному керуванні без водія) зможе знайти дорогу через лінійний лабіринт чи тунель будь-якого розміру.

У подальших завданнях поєднуються вправи з іншими виконавцями (Художник, Фермерка, Пурх, Бджола та інші) і безмашинні вправи, спрямовані на розвиток обчислювального мислення. Ці вправи демонструють **зв'язок алгоритмічних понять і конструкцій з повсякденним життям:** ідея повторення у контексті приспіву пісні, поняття функцій з параметром в інструкції складання браслету з намистинок, пересилання повідомлень різними способами тощо. У процесі виконання вправ учні заповнюють власні електронні картки досягнень, заробляючи своєрідні трофеї.

Для того щоб учитель міг спостерігати за статистикою успішності своїх учнів, існує система реєстрації учасників проекту. Так **зберігається інформація про виконані вправи** й отримані трофеї кожним з учнів (рис. 5). Окремо позначаються вправи, у котрих складено неефективний код, який, тим не менш, відповідає поставленому завданню. Такі рішення можна згодом проаналізувати колективно і спільно знайти кращий спосіб пройти лабіринт.

У поданих навчальних матеріалах реалізується кілька інноваційних методів і прийомів навчання. Завдання придатні до використання в умовах **дистанційного і змішаного навчання.** Усі матеріали доступні он-лайн, зі школи чи дому, тож учень може працювати з ними у зручний для нього час, з перервами чи з додатковими спробами.

У формуванні навчального змісту використана ще одна сучасна педагогічна технологія — **обернене навчання,** у якому широко залучаються додаткові ресурси отримання інформації, окрім пояснень учителя. Власне, роль вчителя у такому навчанні полягає не у викладанні нового матеріалу, а у супроводі учнів у процесі його опанування, коментарях та корекції навчального процесу. Поміж вправами на сайті пропонуються відеофрагменти з доступним тлумаченням алгоритмічних конструкцій (на жаль, лише англійською мовою, проте з українськими субтитрами). Ці мініуроки проводять відомі програмісти, наприклад Біл Гейтс пояснює конструкцію галуження, а Марк Цукенберг — цикли з післяумовою.

За успішне виконання завдань учень отримує не оцінки чи бали, а відзнаки, трофеї, що відображає одну з тенденцій, котра називається **цифровими значками,** які служать для візуалізації здобутих знань і навичок. Якщо учень збере достатню кількість трофеїв, їх можна перевести у призи — сертифікати для Skype, Dropbox тощо. Загалом, використання типових для ігрового середовища елементів: яскраві персонажі з попередньою легендою, накопичення трофеїв, поетапне ускладнення рівнів вка-

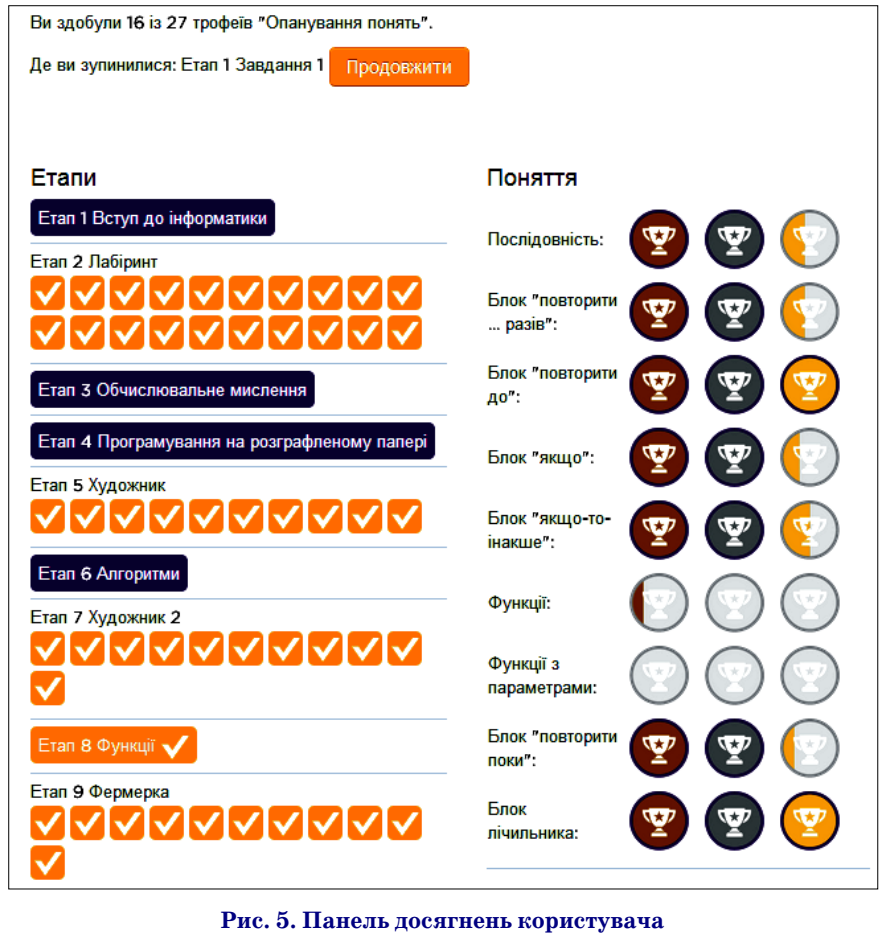



**Рис. 5. Панель досягнень користувача**

зує на використання педагогічної технології **ігрофікації**.

Розробники наводять кілька рекомендацій з проведення занять на базі їхнього ресурсу. Оскільки у вивченні програмування метою не є набуття практичних навичок роботи з комп'ютером, доцільною є **парна і групова** форми роботи. Так, учні разом можуть швидше знайти розв'язок задачі, обговорити варіанти рішень, навести аргументи на підтримку або заперечення тієї чи іншої ідеї. Важливо, щоб учитель не пропонував готових відповідей на запитання, що виникають, а скеровував учнів до самостійного пошуку. Для цього можна запровадити правило **«Спитай сусідів, а тоді вчителя»**. Це значною мірою суперечить традиційним уявленням про дисципліну в класі, але є надзвичайно корисним прийомом, який заохочує до спільної роботи і взаємодії. Відомо, що найбільшу ефективність має навчання інших, зокрема у вирішенні практичних задач, тож учні, котрі знають відповідь на запитання і допомагають однокласникам, ефективно закріплюють свої знання. Коли ж учні самостійно не можуть знайти рішення, бажано, щоб учитель також не давав прямої відповіді на запитання, а скоріш скеровував до її пошуку, пропонував разом дослідити різні варіанти і шляхи розв'язання. Так в учнів формуються навички самостійної роботи і критичного мислення, впевненість у власних знаннях та вміння їх застосовувати.

Успішне проведення заходів «Години коду» дозволило продовжити роботу з розробки нових навчальних матеріалів і влітку 2014 року випущено нові набори інтерактивних вправ, котрі допомагають ознайомитись з основними поняттями програмування. Окрім згаданої програми, на сторінці проекту [9] можна знайти інші ресурси для вивчення програмування у середніх школах, поділені на категорії: уроки для початківців, безмашинні вправи, вивчення мов програмування, програмування для мобільних пристроїв, робототехніка, онлайнові курси університетів тощо. Наведений перелік різноманітних середовищ програмування, придатних для використання **наймолодшими учнями**: Scratch (рис. 7), Tynker, LightBot тощо. До речі, влітку 2014 р. анонсований вихід версії Scratch Jr, призначеної для учнів, котрі ще не вміють читати: у блоках використовуються інтуїтивно зрозумілі зображення дій, здебільшого пов'язаних із переміщенням персонажів на робочій області.

Набір онлайнових підручників (англійською мовою) навчає програмуванню Java Script з використанням платформ Khan Academy, Codecademy, CodeHS, Code Avengers, CodeCombat. Окрім Java Script пропонуються ресурси для вивчення таких **мов програмування**, як Python, Ruby, RoboMind, ObjectiveC, Java, Processing (рис. 8). Написання коду часто відбувається безпосередньо у браузері, де одразу ж відбувається його перевірка. Отже, не потрібно встановлювати додаткове програмне забезпечення, налаштовувати середовище розробки — потрібен лише комп'ютер з доступом до мережі Інтернет.

Навчання може відбуватись в умовах різних класів і технічного обладнання, тому передбачено **підручники для широкого спектру пристроїв**: LightBot для iOS, Android (також працює у браузері), AppInventor для браузера і платформи Android, Kodable, Hopscotch, Two Lives Left для iPad, Kodu для Windows, xBox, а також KidsRuby для установки на комп'ютер користувача.

Такий широкий перелік середовищ для початкового навчання програмуванню підтверджує тезу про те, що важливо вивчати основні принципи, а не детальні особливості конкретної мови. На платформі Code.Org сформовано навчальні курси трьох рівнів для учнів від 1 до 9 класу, кожен з яких містить набори вправ, завдань та тестових питань. У підсумку вивчення цих матеріалів кожен учень створить інтерактивну історію або гру, посиланням на які можна поділитись зі своїми друзями, що значно підвищує мотивацію до ефективної роботи над завданнями.

Значна практична орієнтованість вправ спрямована на опану-


**Рис. 7. Інтерфейс середовища Scratch Jr**


**Рис. 8. Посилання на ресурси з вивчення мов програмування Python, Java Script, Ruby, Java**

вання **обчислювального мислення**, яке включає в себе розв'язання задач, проектування систем та розуміння людської поведінки, виходячи з базових понять інформатики, а безмашинні вправи вчать розпізнавати відповідні задачі у повсякденному житті. Отже, ресурси проекту Code.Org можна використовувати у процесі вивчення алгоритмізації та програмування на початковому етапі, коли важливо не стільки опанувати певну мову програмування, скільки зрозуміти її основи і сформувати інтерес до більш спеціалізованих занять у майбутньому.

★ ★ ★

**Pasichnyk O. Computational thinking in the computer science lessons**

**Annotation.** The paper examines the notion of computational thinking as a school computer science education component. According to the new curricula, algorithms and programming are taught from the second grade. As content and methods of teaching vary for different age groups, the article presents a range of materials, which can be used to develop computational thinking for the students of 2nd-9th grades.

**Keywords:** computational thinking, middle school programming, school computer science, Code.Org.

★ ★ ★

**Пасичник О. В. Вычислительное мышление на уроках информатики**

**Аннотация.** В статье рассматривается понятие вычислительного мышления (computational thinking) в качестве составляющей школьного курса информатики. Согласно новым программам, разделы, посвященные алгоритмизации и программированию, изучаются, начиная со второго класса. Поскольку содержание и методы обучения различаются для разных возрастных групп, в статье описано материалы, которые можно использовать для формирования вычислительного мышления у учащихся 2–9 классов.

**Ключевые слова:** вычислительное мышление, программирование в средней школе, информатика в школе, Code.Org.

### Література

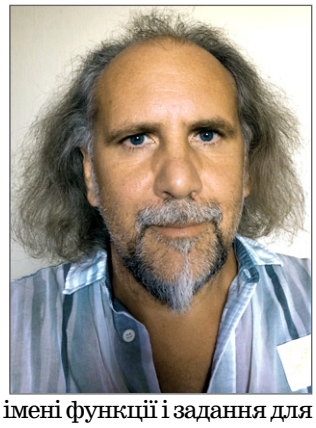
1. Computational Thinking, Jeannette M. Wing — Communications of the ACM, March 2006 [Електронний ресурс]. — Режим доступу: http://www.cs.cmu.edu/~CompThink/papers/Wing06.pdf.
2. Exploring Computational Thinking [Електронний ресурс]. — Режим доступу: http://www.google.com/edu/computational-thinking/what-is-ct.html.
3. We Can Code It! Tasneem Raja [Електронний ресурс]. — Режим доступу: http://goo.gl/E2V6j3.
4. Навчальні програми з інформатики Великобританії [Електронний ресурс]. — Режим доступу: http://www.computingatschool.org.uk/.
5. Навчальні програми з інформатики США [Електронний ресурс]. — Режим доступу: http://csta.acm.org/Curriculum/sub/K12Standards.html.
6. Навчальні програми з інформатики України [Електронний ресурс]. — Режим доступу: http://www.mon.gov.ua/ua/often-requested/educational-programs/.
7. Проект Code.Org [Електронний ресурс]. — Режим доступу: http://code.org/.
8. Відео-звернення про проект Година коду [Електронний ресурс]. — Режим доступу: http://goo.gl/6hspdN.
9. Навчальні матеріали проекту Code.Org [Електронний ресурс]. — Режим доступу: http://learn.code.org/ ; на момент написання статті доступні бета-версії нових курсів http://learn.code.org/beta.

★ ★ ★

# НАВЧАННЯ ОСНОВНИХ ІДЕЙ ФУНКЦІОНАЛЬНОГО ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ МОВИ OBJECT PASCAL

**Горошко Юрій Васильович,**

*завідувач кафедри інформатики і ОТ Чернігівського національного педагогічного університету імені Т. Г. Шевченка, доктор педагогічних наук, доцент.*



Парадигма програмування — це сукупність ідей і понять, через які визначається стиль написання програми. Парадигма, у першу чергу, визначається базовою програмною одиницею. У сучасній індустрії програмування дуже часто парадигма програмування визначається набором інструментів програміста, а точніше, мовою програмування і бібліотеками, що використовуються.

Інколи парадигму програмування ще називають стилем програмування, хоча термін «стиль» щодо програмування може мати й інший сенс. У даній статті під терміном «стиль» матимемо на увазі якраз парадигму програмування.

Відомо кілька основних парадигм програмування, найважливішими з яких на даний момент є парадигми імперативного, функціонального, декларативного, об'єктно-орієнтованого програмування.

Зауважимо, що, мабуть, найбільш незнайомою для студентів педагогічних університетів є парадигма функціонального програмування.

У функціональному програмуванні наголошується на застосуванні функцій, на відміну від імперативного програмування, у якому наголошується на змінах у стані й виконанні послідовностей команд. Іншими словами, функціональне програмування є способом створення програм, у яких єдиною дією є виклик функції, єдиним способом поділу програми є створення нового імені функції і задання для цього імені виразу, за яким обчислюється значення функції, а єдиним правилом композиції є оператор суперпозиції функцій. Жодних комірок пам'яті, операторів надання значень, циклів, чи, тим більше, блок-схем чи передавання управління [1].

Користь в оволодінні функціональною парадигмою програмування полягає в тому, що під час написання програми у такому стилі, потрібно розбити програму на невеликі частини, які є простими й надійними у той же час.

В основу функціонального програмування покладено дві ідеї: