Притика О., Юрченко А. Формування навичок організації циклічних обчислень на уроках інформатики старшої школи. *Освіта. Інноватика. Практика,* 2022. Том 10, № 2. С. 30-37. DOI: 10.31110/2616-650X-vol10i2-004

Prytyka O., Yurchenko A. Formuvannia navychok orhanizatsii tsyklichnykh obchyslen na urokakh informatyky starshoi shkoly [Formation of cyclic calculation organization skills in it-lessons]. *Osvita. Innovatyka. Praktyka – Education. Innovation. Practice,* 2022. Vol. 10, № 2. S. 30-37. DOI: 10.31110/2616-650X-vol10i2-004

УДК 378.14
DOI: 10.31110/2616-650X-vol10i2-004

Оксана ПРИТИКА[1], Артем ЮРЧЕНКО[2]
Сумський державний педагогічний університет імені А.С. Макаренка, Суми, Україна
[2]https://orcid.org/0000-0002-6770-186X
a.yurchenko@fizmatsspu.sumy.ua

ФОРМУВАННЯ НАВИЧОК ОРГАНІЗАЦІЇ ЦИКЛІЧНИХ ОБЧИСЛЕНЬ НА УРОКАХ ІНФОРМАТИКИ СТАРШОЇ ШКОЛИ

Анотація. *Найбільший потенціал для формування алгоритмічного мислення школярів, крім математики, має інформатика. Багато в чому роль інформатики у розвитку алгоритмічного мислення обумовлена навичками циклічних обчислень, які формуються при вивченні програмування. Проте, не зважаючи на значний напрацьований досвід у цій галузі, через постійний розвиток ІТ, мов програмування та середовищ програмування і відповідно часте оновлення навчальних програм з інформатики, маємо констатувати відсутність ефективних напрацьованих методик формування навичок організації циклічних обчислень та достатнього дидактичного матеріалу. У статті за контент-аналізом матеріалів мережі Інтернет виявлено найбільш популярні мови програмування, які вивчаються в ЗЗСО, а також з'ясовано стан розробленості проблеми формування навичок організації циклічних обчислень на уроках інформатики старшої школи. На основі аналізу навчальних програм з інформатики та чинних підручників з інформатики старшої школи на предмет формування навичок організації циклічних обчислень на уроках інформатики старшої школи виявлено методичні проблеми, які пропонується вирішувати з використанням авторських дидактичних матеріалів. Встановлено, що учням старшої школи притаманний низький рівень зацікавленості вивченням програмування та відсутність мотивації. Вивчення думки вчителів щодо ситуації з небажанням вивчати програмування, зокрема, циклічних обчислень, виявила, що серед основних причин – мала кількість годин на вивчення циклів, відсутність достатньої матеріально-технічної бази, традиційні (лекційно-практичні) методи навчання, відсутність достатньої кількості завдань для опанування циклічних обчислень. Подальшого дослідження потребують: питання методичного супроводу формування навичок організації циклічних обчислень на уроках інформатики старшої школи в умовах дистанційної освіти, а також в позаурочному форматі навчання.*

Ключові слова: *циклічні обчислення; навички організації циклічних обчислень; навчання програмувати; вивчення мов програмування; навчання інформатики; інформатика у профільних класах.*

Oksana PRYTYKA[1], Artem YURCHENKO[2]
Makarenko Sumy State Pedagogical University, Sumy, Ukraine
[2]https://orcid.org/0000-0002-6770-186X
a.yurchenko@fizmatsspu.sumy.ua

FORMATION OF CYCLIC CALCULATION ORGANIZATION SKILLS IN IT-LESSONS

Abstract. *Apart from mathematics, informatics has the greatest potential for forming algorithmic thinking in schoolchildren. In many ways, the role of informatics in the development of algorithmic thinking is due to the skills of cyclic calculations that are formed when learning to program. However, despite the considerable experience gained in this field, due to the constant development of IT, programming languages, and programming environments and, accordingly, the frequent updating of computer science curricula, we have to state the lack of effective methods of developing skills for the organization of cyclic calculations and sufficient didactic material. In the article, based on the content analysis of the materials on the Internet, the most popular programming languages studied in school were identified, as well as the state of development of the problem of forming skills for the organization of cyclic calculations in high school computer science classes was clarified. Based on the analysis of computer science curricula and current high school computer science textbooks for the purpose of forming skills in the organization of cyclic calculations in high school computer science lessons, methodological problems were identified, which are proposed to be solved using the author's didactic materials. It was established that high school students have a low level of interest in learning programming and a lack of motivation. The study of teachers' opinions regarding the situation with reluctance to study programming, in particular, cyclic calculations, revealed that among the main reasons are a small number of hours for studying cycles, the lack of sufficient material and technical base, traditional (lecture-practical) teaching methods, the lack of a sufficient number of tasks for mastering cyclic calculations. Further research is needed: on the issue of methodological support for the formation of skills in the organization of cyclic calculations in high school informatics classes in the conditions of distance education, as well as in the extracurricular format of education.*

Keywords: *cyclic calculations; skills of organizing cyclic calculations; learning to program; learning programming languages; learning computer science; computer science in specialized classes.*

Постановка проблеми. Одним з основних завдань закладу освіти є інтелектуальний розвиток учнів, важливою складовою якого є алгоритмічне мислення. Найбільший потенціал для формування алгоритмічного мислення школярів, крім математики, має інформатика. Багато в чому роль

інформатики у розвитку алгоритмічного мислення обумовлена навичками циклічних обчислень, які формуються при вивченні програмування.

Аналіз актуальних досліджень. За аналізом науково-методичних розвідок, присвячених навчанню програмувати та дотичних до цієї теми [1-7], виявлено: концептуальні засади вивчення програмування (М. Жалдак, В. Конюшко, Б. Маккарти, Г. Шилдт та ін.); проблеми освітніх розривів у ланці «школа-університет» під час вивчення мов програмування і способи їх подолання (В. Клочко, Н. Морзе, О. Овчарук та ін.) Так, М. Жалдаком обґрунтовано проблему невідповідності необхідних знань з програмування та часу, який відводиться на їх засвоєння у ЗЗСО. І. Мінтій у навчанні програмувати пропонує використовувати теорію винахідницьких рішень. О.Співаковським пропонується діяльнісний підхід та активні методи навчання. На проектному методичному підході, як дієвому шляху розв'язання проблеми формування алгоритмічного мислення молоді загострюють увагу Т. Вдовичин, І. Дединський, Л. Лазурчак та ін. Натомість, С. Іщеряковим наголошується на дієвості та ефективності задачного підходу вивчення програмування, який ґрунтується на принципі «одна задача – один розв'язок». Отже, проблема змісту, якості і рівня підготовки учнів при формуванні в них навичок програмувати перебуває в центрі уваги педагогів-дослідників.

Процес навчання програмування в ЗЗСО прийнято умовно розбивати на кілька етапів. Перед початком навчання вчитель стикається з проблемою вибору мови програмування для вивчення. Одна група вчителів розглядає тему «Алгоритмізація та програмування» на основі формальних алгоритмів, побудувавши навчання учнів на мові блок-схем. Інша група вчителів інформатики навчає учнів тієї мови програмування, яку сама знає і за допомогою якої сама вміє розв'язувати типові завдання. Таким чином, на початку вивчення програмування вчителі частіше обирають мову програмування з урахуванням власної компетентності і рідше з огляду на інтереси учнів.

Більшість вчителів на початку XXI століття викладали мову Basic. Потім у багатьох школах вивчалася мова Pascal, яка вважалася більш прийнятною з методичної точки зору для вивчення основних принципів програмування. Мова Pascal є навчальною структурною мовою програмування, яка передбачає не тільки вивчення алгоритмічних конструкцій, формування логічного і алгоритмічного мислення в учнів, а й вирішення складних технологічних і виробничих завдань. Наразі уже більшого поширення набувають мови програмування C, C ++, Visual C ++, Delphi, Java, Python та інші.

В якості основних видів організації процесу навчання програмуванню можна виділити традиційні види занять: лекційні, лабораторні та практичні заняття. Для проведення занять необхідно вибрати відповідне навчально-методичне, технічне і програмне забезпечення.

Вивчення програмування в школі проводиться для різних категорій учнів. Головним чином це пов'язано зі спеціалізацією (профільністю) навчання. Одна категорія учнів вивчає програмування у відповідності до обов'язкового змісту, де важливим є вивчення основних алгоритмічних конструкцій з використанням простих типів даних і масивів. Інша категорія учнів вивчає програмування на поглибленому рівні, що відповідає навчанню у профільних класах. Крім базових знань в галузі програмування учні повинні оволодіти однією із сучасних мов програмування і навчитися оперувати основними типами даних та алгоритмізувати обчислювальні процеси.

Втім, не зважаючи на значний напрацьований досвід у цій галузі, через постійний розвиток ІТ, мов програмування та середовищ програмування і відповідно часте оновлення навчальних програм з інформатики маємо констатувати відсутність ефективних напрацьованих методик формування навичок організації циклічних обчислень учнями старшої школи та достатнього дидактичного матеріалу.

Мета дослідження: виявити особливості формування навичок організації циклічних обчислень на уроках інформатики старшої школи

Для досягнення мети використано низку **методів** дослідження:

теоретичні – систематизація і узагальнення науково-методичних джерел для обґрунтування актуальності проблеми дослідження; контент-аналіз з метою характеристики мов програмування та їх затребуваності на ринку праці; аналіз навчальних програм та чинних підручників з інформатики для визначення вимог до рівня підготовленості учнів з програмування загалом і умінь організації циклічних обчислень, зокрема;

емпіричні – опитування учнів і бесіди з учителями для виявлення практичного стану розробленості проблеми формування навичок організації циклічних обчислень на уроках інформатики старшої школи.

Виклад основного матеріалу. Цифровізація суспільства призвела до потреби активного опанування цифрових технологій вже у закладах загальної середньої освіти. І якщо певний час школи орієнтувалися на підготовку в межах предмету «Інформатика» на користувача програмного засобу чи інформаційної системи, то сьогодні прийшло усвідомлення важливості формування на уроках інформатики теоретико-математичних основ функціонування інформаційних систем, що серед іншого передбачає формування елементарних навичок алгоритмізації і програмування, яке реалізується, як правило, на основі навчання циклічних обчислень (якщо обчислювальний процес містить багаторазові обчислення за однаковими (однотипними) математичними залежностями, але для різних значень

змінних, то такі обчислення називають циклічними), і при цьому не залежить від конкретної мови програмування.

Короткий опис мов програмування, котрі наразі обираються для вивчення у закладах загальної середньої освіти для формування навичок циклічних обчислень, наведено на рис. 1. Серед таких: Scratch, Visual Basic, Pascal, Object Pascal, C++, Python, Java та Ruby.

	Scratch	Visual Basic	Pascal	Object Pascal	C++	Python	Java	Ruby
Тип мови	Графічна, навчальна	Процедурна, об'єктно-орієнтована, подієво-орієнтована	Імперативна, структурована	Об'єктно-орієнтована, загальна, процедурна	Об'єктно-орієнтована, мультипарадигмальносна, процедурна, компільована	Об'єктно-орієнтована, рефлективна, імперативна, функціональна	Мультипарадигмальна, мова JVM	Об'єктно-орієнтована
Тип виконання	Інтерпретуючий	Компілюючий, інтерпретуючий	Компілюючий	Компілюючий	Компілюючий	Інтерпретуючий	Інтерпретуючий	Інтерпретуючий
Рік появи	2007	1991	1970	1986	1983	1991	1995	1995
Останні оновлення	3.0 (02.01.2019)	6.0 (1998)	7.1 (03.1994)	10.3.2 Rio (18.07.2019)	C++17 (12.2017)	3.7.17 (19.10.2019)	Java SE 12 (03.2019)	2.6.5 (01.10.2019)
Автор/ розробник	Мітчелл Резник / MIT Media Lab	Майкрософт	Ніклаус Вірт	Ларрі Теслер	Страуструп Бйорн	Python Software Foundation і Гвідо ван Россум	Джеймс Гослінг і Sun Microsystems	Мацумото, Юкіхіро
Розширення файлів	.sb , .sb2 , .sb3	.bas, .cls, .frm, .vbp, .vbg	.pas, .inc	.p, .pp, .pas	.cc, .cpp, .cxx, .c, .c++, .h, .hpp, .hh, .hxx, .h++	.py, .pyw, .pyc, .pyo, .pyd	.java, .class, jar	.rb чи .rbw
Система типів	Динамічна	Статична, сувора, динамічна	Статична, сильна, безпечна	Сильна, динамічна	Статична	Сильна, динамічна	Сильна, динамічна	Строга, динамічна
Основні реалізації	Scratch	Microsoft Visual Studio	Free Pascal, Turbo Pascal, PascalABC.NET	Lazarus, Delphi, Virtual Pascal	Microsoft Visual C++, Intel C++, Embarcadero C++ Builder, Turbo C++, DevC++	CPython, Jython, IronPython, PyPy, Stackless	C++, C, Ада, Simula, Smalltalk, Object Pascal, Оберон, Eiffel, Модула-3, Mesa	Ruby MRI, JRuby,Rubinius
Підштовхнула до розвитку мови	Snap!, AppInventor, Pocket Code	Visual Basic .NET, REALbasic, Gambas, Xojo, Basic4ppc	Ада, Object Pascal, Java, Oxygene	C#, Java, Nim	C, Сфмула, Алгол 68, Клу, ML, Ада	Ruby, Boo, Groovy, ECMAScript, CoffeeScript, Swift, Nim	-	-

Рис. 1. Порівняльна характеристика мов програмування

Програма, написана будь-якою мовою, складається з певного коду. Зазвичай він виконується послідовно: рядок за рядком, зверху донизу. Але є такі конструкції коду, які змінюють лінійне виконання програми. Їх називають керуючими конструкціями.

Завдяки таким конструкціям, код можна виконувати вибірково. Наприклад, запустити один блок коду замість іншого.

Одним із видів керуючих конструкцій в програмуванні є циклічні конструкції або цикли.

Цикли – це різновид керуючих конструкцій для організації багаторазового виконання однієї й тієї ж ділянки коду.

Код усередині такої конструкції виконується циклічно (повторюється). Кожне виконання коду – це ітерація циклу. Кількість ітерацій регулюється умовою циклу. Код, який виконується усередині циклу, називають тілом циклу.

Відомі такі види циклів (рис. 2).

Рис. 2. Види циклічних конструкцій

Цикли з передумовою (цикли «Поки») – це цикли в яких умова виконання визначається перед першою ітерацією.

Робота циклів з передумовою заключається у наступному. Обчислюється значення виразу (тобто умова), що має бути логічним виразом. Якщо результат обчислення виразу істинний, то виконується тіло циклу (простий або складовий оператор). Потім знову перевіряється умова і т.д. Якщо результат хибний, то відбувається вихід із циклу і керування передається першому оператору, який є наступним за циклом.

Цикли з післяумовою (цикли «До») виконання визначається після першої ітерації. Саме такі види завжди виконуються мінімум один раз. Цикли з умовою актуальні тоді, коли потрібно виконати певну

дію, доки реалізується певна умова: наприклад, зчитувати введені користувачем дані, доки він не введе слово “stop”.

Робота циклу з післяумовою відбувається так. Виконується послідовність операторів, що зазначені на початку циклу (тому тіло циклу виконається хоча б один раз). Далі проводиться перевірка продовження циклу: якщо значення записаного виразу дорівнює false (хибне значення), тіло циклу виконується знову. Якщо значення виразу дорівнює true (істинне), відбувається вихід із циклу.

Цикли з лічильником (цикли «Для») – кількість ітерацій визначається змодельованим лічильником. В умові циклу задається його початкове та кінцеве значення. Кожну ітерацію лічильник збільшує. За допомогою циклів з лічильниками можна наперед визначити кількість ітерацій.

Робота таких циклів базується на виконанні певних дій ту кількість разів, яка записана в умові циклу. Умовою циклу слугую лічильник, якому наперед задано кількість повторів і він тільки контролює цю кількість. Ці цикли бувають корисними, коли потрібно перебрати всі елементи в колекції. Цикли з лічильником називають «циклами для...» – «для кожного елемента деякої колекції здійснити такі дії».

Допустимі випадки, коли виконання циклу можна перервати до досягнення його умови. Наприклад, якщо є деяка колекція зі 100 чисел і необхідно зрозуміти, чи вона містить негативні числа. Можна розпочати перебір всіх чисел, використовуючи цикл «для». Але коли буде знайдено перше негативне число, не обов'язково перебирати інші числа, що залишилися. В такому випадку можна перервати виконання циклу, якщо його подальше виконання не має сенсу. Подібні ситуації називають переривання циклу.

Безумовні цикли – цикли, що виконуються нескінченно. Наприклад: «Поки 1 = 1, друкувати "1 = 1"». Така програма виконуватиметься, доки її вручну не перервуть.

Дані цикли теж бувають корисні, коли використовуються разом із перериванням циклу «зсередини».

Наведемо синтаксис усіх видів циклів на мовах програмування Паскаль, C++, Java та Python. Ці мови найбільше зустрічаються при вивченні інформатики в ЗЗСО.

Таблиця 1

Синтаксис циклічних конструкцій

	Паскаль	**C++**	**Java**	**Python**
Цикл з передумовою	**while** умова **do** тіло циклу**;**	**while (**умова**) {** тіло циклу**;** **}**	**while (**умова**) {** тіло циклу**;** **}**	**while** умова**:** тіло циклу
Цикл з післяумовою	**repeat** тіло циклу **until** умова	**do {** тіло циклу **}** **while (**умова**);**	**do {** тіло циклу **}** **while (**умова**);**	
Цикл з лічильником	**for** параметр **:=** поч_знач **to** кін_знач **do** тіло циклу;	**for (**лічильник **=** значення**;** лічильник **<** значення**;** крок циклу**) {** тіло циклу; **}**	**for (** ініціалізація лічильника**;** **[**умова**];** **[**зміна лічильника**])** **{** тіло циклу **}**	**for** int_var **in** функція_range **:** тіло циклу

Незважаючи на те, що цикли for і while (з лічильником та з умовою) необхідні для повторення певної кількості разів однієї і тієї ж операції, ці цикли відрізняються один від одного і мають свою специфіку. Навіть з урахуванням їхньої формальної взаємозамінності.

While зручний тоді, коли операція, що повторюється, проводиться до того часу, поки умова правильна, тобто повертає значення True. Звідси можлива ситуація, коли цикл не спрацює жодного разу або повторюватиметься нескінченно.

Цикл for застосовується для послідовного маніпулювання елементами ітератора. Іншими словами, він проходить по черзі елементи об'єкта (наприклад, списку) і закінчується після їхнього повного перебору.

Цикл for досить часто використовується для роботи з масивами. Наприклад, програмний код на мові C++ за допомогою циклу з лічильником дозволяє заповнити масив із n елементів нулями:

```
int a[n];
for (int i = 0; i < n; i++) {
   a[i] = 0;
}
```

Таким чином, for зручний для перебору, а while – для перевірки істинності умови перед кожною ітерацією.

З табл.1 бачимо, що синтаксис циклічних конструкцій в більшій мірі однаковий, але кожна мова програмування має свої відмінності. Незважаючи на це принцип використання циклів є однаковим у різних задачах на використання циклів.

Нами досліджувалася проблема формування навичок організації циклічних обчислень на емпіричному рівні. Опитування проводилось серед учнів шкіл м. Суми, а саме КУ Сумська спеціалізована школа I-III ступенів № 2 ім. Д. Косаренка, КУ Сумська загальноосвітня школа I-III ступенів №6, КУ Сумська спеціалізована школа I-III ступенів №7 імені М. Савченка, КУ Сумська спеціалізована школа I-III ступенів №17, КУ Сумська спеціалізована школа I-III ступенів №25.

Запитання анкети

1. В якому класі ти навчаєшся?
2. Чи хочеш вивчати програмування в школі?
3. Чи вивчаєш програмування поза школою?
4. З якими мовами програмування ти знайомий?
5. Постав позначку там, де можна використати циклічні обчислення для визначення результату:

a. 1+2+3+4+…+10
b. 1-2*3-4+5
c. 1, 1, 2, 3, 5, 8, 13, ***x=?***
d. *2x+3=x-5*

За результатами анкетування (в опитуванні брали участь 73 учні різних класів) (рис.3), виявлено, що відношення до програмування дуже різне, причому воно змінюється в бік погіршення із переходом до старшої школи.

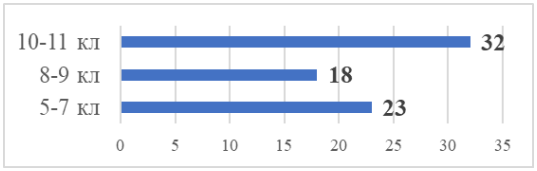


Рис. 3. Розподіл учасників опитування

Нижче наведені думки учнів щодо програмування (рис. 4-5).

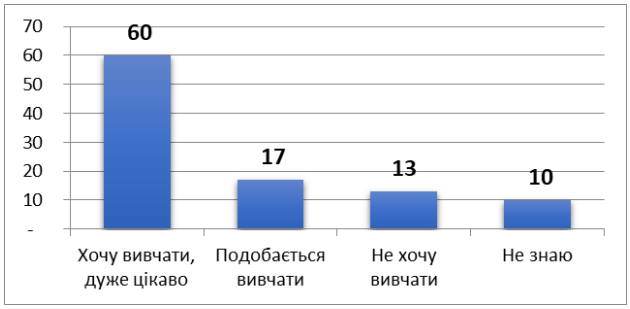


Рис. 4. Вислови учнів 5-6-х класів щодо ставлення до програмування (%)

Рис. 5. Вислови учнів 8-9-х класів щодо ставлення до програмування (%)

На запитання «Чи вивчаєш програмування поза школою?» більшість опитаних відповіли «Ні» (табл.2).

Таблиця 2

Розподіл відповідей на запитання 3

Клас	*5-7 кл*	*8-9 кл*	*10-11 кл*
Кількість відповідей загалом	5	2	6
По відношенню до вікової групи	22%	11%	19%
По відношенню до загальної кількості	7%	3%	8%

На четверте запитання про обізнаність у мовах програмування маємо ситуацію, що найчастіше учнями в школі вивчається Scratch (86,7%), Python (61,3%), Object Pascal (20,0%), Pascal (12,0%), C++ (6,7%), інша (5,3%) мова програмування. Високий відсоток учнів, які знають мову Scratch, пояснюємо тим, що саме ця мова є обов'язковою для вивчення ще в початкових класах.

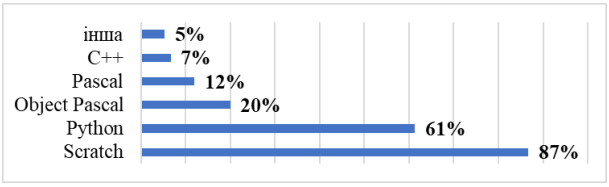


Рис. 6. Обізнаність учнів щодо мов програмування

На п'яте запитання про доцільність використання циклічних обчислень (лише відповіді *a* і *c* були правильними) коректно відповіли лише 9% від загальної кількості респондентів (табл.3).

Таблиця 3

Розподіл відповідей на запитання 5

Клас	*5-7 кл*	*8-9 кл*	*10-11 кл*
Кількість відповідей загалом	1	2	4
По відношенню до вікової групи	4%	11%	13%
По відношенню до загальної кількості	1%	3%	5%

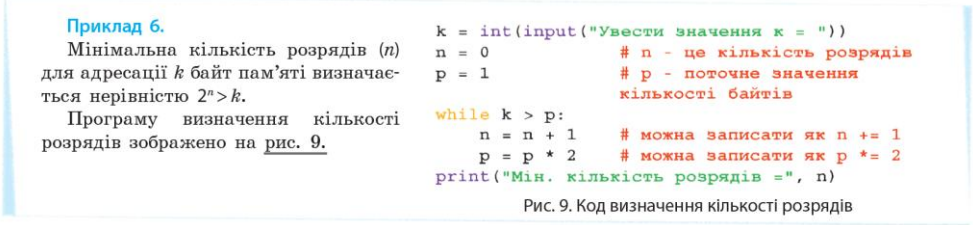
Загалом аналіз відповідей показав, що прослідковується зниження інтересу до програмування з 60% (5-6 класи) до 22% учнів (4% прагнуть і вивчають самостійно, 18% виявили зацікавленість і хочуть вивчати). Зіставлення результатів опитування свідчить про зниження надії навчитися програмувати («я цього ніколи не зрозумію» – 26%, «програмування занадто важке і нудне» – 16%, «хотів би вивчати, але не вірю у свої сили» – 7%).

Також виявлено, що учням старшої школи притаманний низький рівень зацікавленості вивченням програмування та відсутність мотивації.

Розглядаючи актуальні підручники з інформатики для старших класів ЗЗСО на предмет вивчення програмування в них та спираючись на навчальну програму можна стверджувати, що програмування можна знайти тільки у підручнику для профільних класів. Нами детально проаналізовано саме такий підручник авторів В. Д. Руденко, Н. В. Речич, В. О. Потієнко [5].

У даному підручнику для вивчення програмування виділяється розділ «Мова програмування та структури даних». Зміст даного розділу зосереджено у вивченні тем. Із змісту можна бачити, що автори підручника пропонують для вивчення мову програмування Python. Майже весь розділ і присвячено її вивченню.

У підручнику розглядаються цикли з параметром та з умовою (перед і після). Варто зазначити, що весь теоретичний матеріал супроводжується розв'язаними прикладами (рис. 7). Таких прикладів з циклічними обчисленнями – 8.



Приклад 6.

Мінімальна кількість розрядів (n) для адресації k байт пам'яті визначається нерівністю $2^n > k$.

Програму визначення кількості розрядів зображено на рис. 9.

```
k = int(input("Увести значення k = "))
n = 0              # n - це кількість розрядів
p = 1              # p - поточне значення
                   кількості байтів
while k > p:
    n = n + 1      # можна записати як n += 1
    p = p * 2      # можна записати як p *= 2
print("Мін. кількість розрядів =", n)
```

Рис. 9. Код визначення кількості розрядів

Рис. 7. Приклад із підручника

На закріплення знань учням пропонується для розв'язання 8 задач (рис. 8).

Завдання для самостійного виконання

1. Розробіть код обчислення суми для чисел: 2, 7, 21, 9, 33, 13.
2. Розробіть код, обчислення суми непарних чисел, що більші 7, але менші 25.
3. Розробіть код обчислення суми чисел натурального ряду, максимальне значення якого не перевищує 7.
4. Перший член геометричної прогресії 6, а її знаменник — 0.5. Розробіть код обчислення значень членів прогресії, більших 0.6, і визначення номера останнього члена прогресії, що підсумовується.
5. Дано куб, сторони якого набувають п'ять значень: 3, 4.5, 6, 7.5, 9. Розробіть код визначення об'єму для кожного з них.
6. Радіус першої кулі дорівнює 2 см, а радіус кожної наступної збільшується на 0.5 см. Розробіть код для визначення бокових поверхонь перших шести куль.
7. Відомо результати плавання вільним стилем на дистанції 50 м п'яти учнів кожного з трьох 10 класів школи. Розробіть код, за допомогою якого визначається середній час запливу учнями кожного класу.
8. У банк покладено S грн під N відсотків річних. Розробіть код, за допомогою якого визначається кількість років, через які сума вкладу буде не менше N грн.

Рис. 8. Завдання на циклічні обчислення

Незважаючи на достатню кількість прикладів завдань на використання циклів, не вистачає типових задач з програмування та прикладів саме їх розв'язання.

Вивчення думки вчителів щодо ситуації з небажанням вивчати програмування, зокрема, циклічних обчислень, виявила, що серед основних причин – мала кількість годин на вивчення циклів, відсутність достатньої матеріально-технічної бази традиційні (лекційно-практичні) методи навчання, відсутність достатньої кількості завдань для опанування циклічних обчислень.

З наведених причин лише третя і четверта можуть піддаватися корекції на рівні «вчитель-учень», а тому нами на основі спілкування з учителями та за аналізом науково-методичних джерел визначено аспекти, які потенційно можуть змінити психологічні установки учнів при вивченні програмування в школі.

Мотивація – використання соціальних (розуміння соціальної значущості, потреби, орієнтація на майбутнє) і пізнавальних (інтереси, емоції, ідеали, прагнення) мотивів.

Оскільки мотивація сприяє прагненню до навчання, спонукає діяти з максимальною енергією, то доцільними є сугестивні прийоми (це тип прямого повідомлення, через який людина впливає на рішення, думки та поведінку іншої людини чи групи людей, не звертаючись до раціональної аргументації та не використовуючи фізичного примусу), комунікативні атаки, доведення та переконування.

Стимулювання і психологічне налаштування. Щоб зняти блок, що гальмує розуміння суті програмування, доцільним є наведення прикладів і статистичних даних про те, скільки людей з нематематичним складом мислення досягають успіхів у програмуванні, який відсоток людей став програмістами вже у зрілому віці. Так, упереджене ставлення дівчат до програмування можна змінювати, зазначаючи, що першим програмістом у світі була жінка – Ада Лавлейс.

Ігрові методи. Підтримка вмотивованості учнів і їх налаштованість на вивчення програмування можливі із застосуванням ігрових методик. Вибір такої стратегії ґрунтується на дослідженнях С. Іщерякова [2]. На думку науковця, новачки починають програмувати з двох причин – навчання і розвага, причому у першому випадку інтерес згасає через нудність процесу і часту плутанину і нерозуміння логіки алгоритмів, а от у другому випадку, де програмування розцінюється як гра, прослідковується зацікавленість і захоплення учнів програмуванням, що і зумовлює подальші успіхи.

Ставлення до програмування як до гри зумовлює активне використання рольових ігор при моделюванні ситуацій, змагань та інших інтерактивних методів, що можуть наочно і зрозуміло пояснювати складні елементи програмування, які, як правило викликають найбільші труднощі і втрату інтересу.

Командна робота. Сучасна ІТ-індустрія переважно спрямована на командну діяльність над проектами, тому учні здебільшого навчаються роботі у команді над спільними проектами. Для залучення до командної роботи можна застосовувати практику перехресного обміну кодом із завданням розібратися у чужому коді та доповнити його, написання рецензії на чужий код. Заохочується обмін думками, знаннями, взаємна допомога, спільна генерація ідей. Заняття у команді сприяють соціалізації учнів та розвивають їх комунікативні навички.

Самостійна робота і рефлексія. Самостійна робота бачиться доцільною через можливість мережної взаємодії, зокрема, через використання різних інтерактивних веб-сервісів типу learningapps.org, www.pythontutor.com, www.e-olymp.com, http://cppstudio.com. Одним із варіантів взаємодії з учнями може стати пропозиція вчителя виконувати певні завдання, імпортувати їх єдину мережеву презентацію за допомогою сервісу Google Docs. Перевагою даного сервісу є наявність вбудованого чату, що дає можливість онлайн обговорення результатів.

Висновки. Аналіз практичного стану розробленості проблеми виявив тенденцію зниження бажання вивчати програмування із переходом до старших класів. При цьому серед опитаних різняться

думки щодо причин зниження такого бажання. Учням старшої школи притаманний низький рівень зацікавленості вивченням програмування та відсутність мотивації. Вивчення думки вчителів щодо ситуації з небажанням вивчати програмування, зокрема, циклічних обчислень, виявила, що серед основних причин – мала кількість годин на вивчення циклів, відсутність достатньої матеріально-технічної бази, традиційні (лекційно-практичні) методи навчання, відсутність достатньої кількості завдань для опанування циклічних обчислень.

Робота не вирішує повністю аналізовану проблему. Подальшого дослідження потребують: питання методичного супроводу формування навичок організації циклічних обчислень на уроках інформатики старшої школи в умовах дистанційної освіти, а також в позаурочному форматі навчання.

Список використаних джерел

1. Ворожбит А.В., Рибак О.С. Огляд курсу за вибором «основи верстки та веб-програмування». *Фізико-математична освіта*, 2018. Випуск 1(15). С. 20-27.
2. Іщеряков С. Вчити програмування треба в школі чи університеті? *Освітня політика. Портал громадських експертів*. URL: http://osvita.ua/school/54063/
3. Павленко Л.В., Павленко М.П., Хоменко В.Г., Хоменко С.В., Скурська М.М. Інноваційні підходи до вивчення статистики майбутніми ІТ-фахівцями на основі використання мови програмування R. *Фізико-математична освіта*, 2020. Випуск 1(23). С. 97-105.
4. Притика О. В., Юрченко А. О. Про особливості мови програмування С+. *Інформаційні технології в професійній діяльності* : матеріали XIV Всеукраїнської науково-практичної конференції. Рівне : РВВ РДГУ. 2021. С. 155-156.
5. Руденко В. Д., Речич Н. В., Потієнко В. О. *Інформатика (профільний рівень)* : підруч. для 10 кл. закл. загал. серед. освіти. Харків : Вид-во «Ранок», 2019. 256 с.
6. Семеніхіна О.В., Руденко Ю.О. Проблеми навчання програмувати учнів старших класів та шляхи їх подолання. *Інформаційні технології і засоби навчання*, 2018. №66(4), С. 54–64. https://doi.org/10.33407/itlt.v66i4.2149
7. Юрченко А. А., Хворостина Ю. В., Острога М. М., Пунько В. В. Изучение программирования: анализ программ по информатике для старшей школы в Украине. *The 3rd International scientific and practical conference "Perspectives of world science and education"*: Conference proceedings, (November 27-29, 2019) CPN Publishing Group, Osaka, Japan. 2019. P. 587-595.
8. Юрченко А.О., Семеніхіна О.В., Хворостіна Ю.В., Удовиченко О.М. Навчання програмувати в старшій школі крізь призму чинних навчальних програм. *Фізико-математична освіта*, 2019. Випуск 2(20). Ч.2. С. 47-54.

References

1. Vorozhbyt A.V., Rybak O.S. Ohliad kursu za vyborom «osnovy verstky ta veb-prohramuvannia». *Fizyko-matematychna osvita*, 2018. Vypusk 1(15). S. 20-27.
2. Ishcheriakov S. Vchyty prohramuvannia treba v shkoli chy universyteti? *Osvitnia polityka. Portal hromadskykh ekspertiv*. URL: http://osvita.ua/school/54063/
3. Pavlenko L.V., Pavlenko M.P., Khomenko V.H., Khomenko S.V., Skurska M.M. Innovatsiini pidkhody do vyvchennia statystyky maibutnimy IT-fakhivtsiamy na osnovi vykorystannia movy prohramuvannia R. *Fizyko-matematychna osvita*, 2020. Vypusk 1(23). S. 97-105.
4. Prytyka O. V., Yurchenko A. O. Pro osoblyvosti movy prohramuvannia S+. *Informatsiini tekhnolohii v profesiinii diialnosti* : materialy XIV Vseukrainskoi naukovo-praktychnoi konferentsii. Rivne : RVV RDHU. 2021. S. 155-156.
5. Rudenko V. D., Rechych N. V., Potiienko V. O. *Informatyka (profilnyi riven)* : pidruch. dlia 10 kl. zakl. zahal. sered. osvity. Kharkiv : Vyd-vo «Ranok», 2019. 256 c.
6. Semenikhina O.V., Rudenko Yu.O. Problemy navchannia prohramuvaty uchniv starshykh klasiv ta shliakhy yikh podolannia. *Informatsiini tekhnolohii i zasoby navchannia*, 2018. №66(4), S. 54–64.
7. Yurchenko A. A., Khvorostyna Yu. V., Ostroha M. M., Punko V. V. Yzuchenye prohrammyrovanyia: analyz prohramm po ynformatyke dlia starshei shkolы v Ukrayne. *The 3rd International scientific and practical conference "Perspectives of world science and education":* Conference proceedings, (November 27-29, 2019) CPN Publishing Group, Osaka, Japan. 2019. P. 587-595.
8. Yurchenko A.O., Semenikhina O.V., Khvorostina Yu.V., Udovychenko O.M. Navchannia prohramuvaty v starshii shkoli kriz pryzmu chynnykh navchalnykh prohram. *Fizyko-matematychna osvita*, 2019. Vypusk 2(20). Ch.2. S. 47-54.