РОЗДІЛ 4. ОПТИМІЗАЦІЯ НАВЧАННЯ ДИСЦИПЛІН ПРИРОДНИЧО-МАТЕМАТИЧНОГО ЦИКЛУ ЗАСОБАМИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 372.8
DOI 10.5281/zenodo.3547793

В. М. Базурін
ORCID ID 0000-0002-6614-4889
Глухівський національний педагогічний
університет імені Олександра Довженка

МЕТОДИКА НАВЧАННЯ ОСНОВ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ УЧНІВ ЗАКЛАДІВ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ

У статті розкриваються особливості навчання учнів загальноосвітніх шкіл основ об'єктно-орієнтовного програмування. У даний час об'єктно-орієнтована парадигма програмування використовується у більшості мов програмування. Методика навчання об'єктно-орієнтованого програмування має певні відмінності від методики навчання структурного програмування. Шляхи вирішення даної проблеми було знайдено вітчизняними науковцями 15-20 років тому, проте за цей час самі об'єктно-орієнтовні мови програмування продовжували розвиватися.

Одним із шляхів інформатизації освіти України визнано посилення змістової лінії програмування. Застосування змістової лінії програмування повинно сприяти не лише формуванню навичок програмування, а й формуванню алгоритмічного мислення, підвищенню комп'ютерної грамотності учнів.

В основі об'єктно-орієнтованої парадигми програмування лежить поняття об'єкта. Всі числові значення визначаються не через змінні, а через властивості відповідних об'єктів. Дії, які виконує програма, задаються не через функції і процедури, а за допомогою методів відповідних об'єктів. У статті розкрито особливості формування поняття об'єкт, клас, властивість, метод, наслідування та інших, які є фундаментальними поняттями об'єктно-орієнтованого програмування. Запропоновані автором прийоми пояснення основних питань об'єктно-орієнтованого програмування супроводжуються відповідними прикладами.

Перспективами подальших наукових досліджень у даному напрямі є визначення педагогічних умов успішного формування основних понять і прийомів об'єктно-орієнтованого програмування.

Ключові слова: *об'єктно-орієнтоване програмування, заклади загальної середньої освіти, учні, математичні здібності, математична освіта, інформатизація освіти, мова програмування, основи програмування.*

Постановка проблеми. У сучасному світі зростає роль засобів інформаційно-комунікаційних технологій. Ці технології бурхливо розвиваються. Для подальшого розвитку їх необхідна наявність фахівців, які розробляють відповідне програмне забезпечення. Проте у даний час існує деяка невідповідність між попитом на ринку праці і наявністю програмістів в Україні. Попит значно перевищує пропозицію. І така ситуація буде зберігатися певний час, адже професія програміста відноситься до таких, що вимагають від своїх представників високого рівня математичних здібностей, знання мов програмування і умінь створювати програмні продукти. На думку фахівців, ця ніша залишиться незаповненою ще значний час.

Причинами ситуації, що склалася, вважають:

1) падіння загального рівня математичної освіти в Україні;

2) відтік кваліфікованих учителів інформатики в інші галузі народного господарства, де рівень заробітної плати вищий;

3) недостатньо ефективну методику навчання учнів основ програмування.

Науковці мають мало можливості вплинути на перші два чинники. Проте ситуацію з третім чинником можна вирішити.

Одним із найважливіших напрямів інформатизації освіти України є «посилення змістової лінії програмування у навчанні інформатики учнів старшої школи, які обрали фізичний, математичний чи інформаційно-технологічний профіль» [1, с. 196].

Аналіз актуальних досліджень. Питання інформатизації освіти знаходяться в центрі уваги В.Ю. Бикова, О.М. Спіріна, О.П. Пінчука [1], В.Д. Руденка [13].

Шляхи розв'язання проблеми навчання учнів основ програмування відображені у працях М.І. Жалдака, Ю.С. Рамського, І.М. Лукаша [8; 9], Н.В. Морзе [7], В.В. Лапінського, З.С. Сейдаметової, Ф.В. Шкарбана [14], С.О. Семерікова [15], І.В. Скляр [16], П.Г. Шевчука [17; 18] та інших науковців.

Питання вибору мови програмування розглянуто у дослідженнях В.Д. Руденка, В.Ю. Бикова, С.С. Жуковського та інших дослідників.

Результати найновіших досліджень знайшли відображення у змісті сучасних підручників з інформатики [10; 11; 12].

Мета статті – розкрити специфіку навчання учнів основ об'єктно-орієнтованого програмування у середніх та старших класах закладів загальної середньої освіти.

Виклад основного матеріалу. Змістова лінія «Основи алгоритмізації та програмування» є однією з найважливіших у шкільному курсі інформатики. Елементи програмування вивчаються у початкових класах. Програмою рекомендовано вивчати середовище програмування Scratch. Під час роботи у даному середовищі учень працює з різними об'єктами, які називаються спрайтами. Можна вважати, що учні вивчають початки об'єктно-орієнтованого програмування, а саме: об'єкт, операції над об'єктом, поведінку об'єкта та ін.

Отже, деякі елементи об'єктно-орієнтованого програмування учні вивчають у початкових класах.

У 7-8 класах вивчення програмування продовжується. Програмою рекомендовано вивчати мови Python або Object Pascal (Lazarus). Обидві мови програмування підтримують об'єктно-орієнтовану парадигму, тому в процесі їх вивчення в учнів формуються і основні поняття об'єктно-орієнтованого програмування. На вивчення змістової лінії «Основи алгоритмізації і програмування» у 7-8 класах програмою відведено до 30% часу.

У 10-11 класах основи алгоритмізації і програмування вивчаються у класах академічного та профільного рівнів. Усі мови програмування, які вивчаються у школі (Scratch, Python, Object Pascal, Free Pascal, Java та інші), підтримують об'єктно-орієнтовну парадигму, а Java базується на цій парадигмі.

Вивчення теми «Основи об'єктно-орієнтованого програмування» супроводжується вивченням значної кількості абстрактних понять, без правильного засвоєння яких вивчення подальших питань теми є складним. Абстрактні поняття для цієї теми відіграють більшу роль, ніж у вивченні змістової лінії «Основи алгоритмізації та програмування». Саме тому вивчення теми доцільно почати з абстрактних понять.

Об'єктно-орієнтоване програмування – це одна з сучасних парадигм програмування. Відповідно до цієї парадигми, програміст працює з об'єктами, їх властивостями і методами.

Пояснення змісту поняття «об'єктно-орієнтоване програмування» слід починати з прикладу. Уявіть собі, коли у програмі треба використати дані про яблука (маса, діаметр, колір, стиглість). Якщо зберігати ці дані у масивах, то потрібно використати 4 масиви. Якщо збільшується кількість яблук, то обсяг даних зростатиме зі швидкістю, вчетверо більшою. У великих програмах обсяг даних займає значне місце в пам'яті комп'ютера, і вони працюють повільніше.

Якщо ж ці дані помістити в об'єкт, то кожен об'єкт буде характеризуватися такими властивостями: маса, діаметр, колір, стиглість. Достатньо створити екземпляр класу і присвоїти значення відповідним властивостям.

Для того, щоб використати у програмі характеристики певного об'єкта, слід звернутися до відповідної властивості.

Об'єктно-орієнтоване програмування базується на таких концепціях: поліморфізм, успадкування, інкапсуляція. Це абстрактні поняття, і пояснити їх на прикладах досить важко.

Інкапсуляція – це комбінування даних з процедурами і функціями, які маніпулюють цими даними. Тобто, усі дані містяться не в масивах і записах, а у властивостях і полях класу. Є конкретний екземпляр, який має певні властивості і поля.

Наприклад, коло описується координатами центра і радіусом. Це властивості. Над колом можна виконати такі дії: обчислити довжину і площу кола, перевіряти, чи знаходиться точка всередині кола. Це методи.

Поля і методи, що входять до класу, називаються членами класу. Для роботи з класом треба створити його екземпляр. Для звернення до властивостей або методів класу слід вказати назву екземпляра і через крапку – назву властивості.

Інкапсуляція дозволяє забезпечити захист даних від зовнішнього втручання і неправильного використання. Це забезпечується розділенням доступу до даних (полів і властивостей) і методів об'єкта. Дані і методи можуть мати різний рівень доступу.

Для створення правильного уявлення про інкапсуляцію слід застосувати метод доцільних прикладів. Усі властивості слід пояснювати за допомогою таких прикладів, які доступні дітям.

З-поміж різних вправ перевагу слід віддавати завданням з відкритою умовою, тобто таким, умову яких треба до визначити. Наприклад: словесно описати клас МЕБЛІ, назвати властивості цього класу і вказати типи даних для цих властивостей.

Успадкування (наслідування) – це перенесення властивостей батьківського класу до дочірнього класу. Це можливість використання готових класів для створення класів, похідних від них. Новий клас можна визначити за допомогою готового (базового) класу. При цьому новий клас зберігає усі властивості старого класу. Новий клас успадковує як дані старого класу, так і методи для їх обробки.

Під час пояснення цього поняття слід навести відповідні приклади.

Наприклад, клас ФРУКТИ має такі властивості: маса, довжина, ширина, товщина. Похідний від нього клас ЯБЛУКА має такі є самі властивості. Інший похідний клас СЛИВИ має такі ж самі властивості.

У процесі вивчення основ програмування доцільно використати спеціальні вправи, у яких за допомогою словесного опису учні повинні вказати похідні класи тощо. Наприклад: описати клас ОВОЧІ, який має певні властивості (визначити самостійно). Клас ОВОЧІ має два дочірні класи КАРТОПЛЯ і МОРКВА. Описати словесно дочірні класи за допомогою батьківського класу.

Поліморфізм – це властивість, яка надає можливість єдиного визначення єдиного імені для дії, яка застосовується одночасно для всіх об'єктів ієрархії спадкоємства. Для кожного об'єкта враховуються особливості реалізації даної дії. Концепція поліморфізму означає, що всередині процедури викликаються методи, які відповідають не типу формальної змінної, а типу реально переданої змінної. Реалізація концепції поліморфізму означає, що можна створити загальний інтерфейс для групи близьких за значенням дій. Перевагою поліморфізму є те, що він допомагає знижувати складність програм, дозволяє використовувати однаковий інтерфейс для єдиного класу дій.

Однією з вправ можна вибрати таку: описати властивості і методи класу ПРЯМОКУТНИК. У процесі виконання цієї вправи учні мають встановити, що властивостями класу ПРЯМОКУТНИК можуть бути довжина і ширина. Методами класу ПРЯМОКУТНИК можуть бути площа і периметр.

Основні поняття об'єктно-орієнтовного програмування (об'єкт, властивості тощо) застосовуються також і в інших темах. Доцільно запропонувати учням згадати відповідні приклади з інших тем шкільного курсу інформатики.

Наприклад, в електронних таблицях MS Excel об'єктом роботи можуть бути комірка, формула, таблиця. Для цих об'єктів можна встановити значення певних властивостей.

Під час роботи з текстовим редактором MS Word об'єктами можуть бути: символ, слово, абзац, сторінка.

Під час роботи у середовищі операційної системи MS Windows об'єктами зазвичай є: файл, папка, ярлик, диск.

Отже, основними поняттями об'єктно-орієнтованого програмування є: об'єкт, клас, метод, властивості, подія.

Об'єкт – це компонент реальної чи віртуальної дійсності.

Клас – це набір об'єктів, що мають однакові властивості.

Екземпляр – це одиничний об'єкт, що належить класу.

Властивості – це ознаки об'єктів.

Метод – це дія, яку може виконувати об'єкт або можна здійснювати над об'єктом.

Подія – це те, що відбувається з об'єктом.

Приступаючи до вивчення основ програмування, слід пояснити учням, що об'єкт – це те, з чим вони працюють. Об'єктом може бути спрайт (у мові Scratch), екранна форма (у Delphi або Lazarus).

Свою розповідь доцільно побудувати так.

Кілька спрайтів (у Scratch) одного типу об'єднуються в клас. Наприклад, у програмі може існувати кілька об'єктів класу Кіт. Вони мають однакові властивості і відрізняються лише іменем (за замовчуванням Scratch іменує їх Кіт1, Кіт2, Кіт3 тощо). Проте для кожного такого об'єкта можна змінити значення властивостей, і зовнішній вигляд цих об'єктів буде відрізнятися.

Доцільно провести аналогію з життям. Так, клас Яблуко може налічувати багато екземплярів. Цей клас можна охарактеризувати такими властивостями: колір, маса, діаметр.

Для кожного екземпляра класу ЯБЛУКО значення кожної властивості можуть відрізнятися.

У класі знаходяться меблі, які можна віднести до таких класів: СТІЛ, СТІЛЕЦЬ. Ці класи можна об'єднати в один загальний клас під назвою МЕБЛІ. Тобто, клас МЕБЛІ буде батьківським для класів СТІЛ і СТІЛЕЦЬ. Деякі властивості можуть бути успадковані класом СТІЛ від батьківського класу.

Клас СТІЛ можна охарактеризувати такими властивостями: Висота, Ширина, Довжина, Колір.

Клас СТІЛЕЦЬ можна охарактеризувати такими ж самими властивостями. Проте класи СТІЛ і СТІЛЕЦЬ повинні мати відмінності між собою. Для цього слід ввести властивість Форма. Для класу СТІЛЕЦЬ значення цієї властивості може бути одне, а для класу СТІЛ – інше.

Далі на цьому ж прикладі слід пояснити такі поняття, як наслідування. Так, дочірні класи СТІЛ і СТІЛЕЦЬ наслідують усі властивості батьківського класу МЕБЛІ. Проте вони мають відмінну властивість: Форма. Іншу властивість (Тип меблів) дочірні класи не наслідують.

Під час вивчення основ програмування у середовищі Delphi слід запропонувати учням запустити Delphi. На екрані учні побачать форму. Це і є один з основних об'єктів, з якими працюватимуть учні. Форма має певні ознаки (властивості): довжину, висоту, колір тощо. Учням слід запропонувати визначити властивості форми за допомогою Інспектора об'єктів. Таким чином учнів слід підвести до розуміння поняття властивості. Підсумовуючи приклади, учням слід задати питання: Отже, що таке властивість? Властивість – це певна ознака об'єкта.

Окрім властивості, визначають ще й її значення. Тобто, властивість – це ознака, а значення властивості – це числове або символьне значення ознаки. Наприклад, ширина форми – це властивість, а 640 пікселів – це значення властивості.

В об'єктно-орієнтованих мовах програмування назву властивості звичайно записують через крапку після імені об'єкта. Наприклад:

Form1.Width – це ширина у пікселях форми, яка має назву Form1.

Формувати в учнів поняття методу слід у такій послідовності. Спочатку слід запропонувати учням пригадати, які дії може виконувати користувач з вікном програми. Учні пригадують з попереднього курсу інформатики, що вікно програми можна згорнути, розгорнути, закрити. Тоді слід зазначити, що поведінку об'єкта задають за допомогою методів. Наприклад, для закриття форми слід використати код:

Application.Exit();

Назва методу зазначається після імені об'єкта або назви властивості через крапку. Наприклад:

Form1.Close();

4. Навчання основ програмування на мові Java

Мова програмування Java належить до об'єктно-орієнтованих мов програмування. Спеціалісти фірми Sun Microsystems розробили мову Java в 90-х роках XX ст. На даний час актуальною є 7-а версія цієї мови.

Вибір мови програмування Java зумовлений тим, що це одна з найбільш сучасних мов, яка продовжує розвиватися. Програму, створену на мові Java, можна перетворити у виконуваний файл і продемонструвати друзям або батькам. Процес створення екранної форми може відбуватися у режимі «що бачу – те й отримую». Тобто, створити екранну форму з відповідними елементами можна в редакторі форм. Це значно спрощує процес створення програми.

Починати навчання основам програмування слід після того, як в учнів вже були сформовані основні поняття об'єктно-орієнтованого програмування. Під час створення програми учень працює з класами, їх властивостями і методами.

Після того, як учні засвоїли основні поняття об'єктно-орієнтованого програмування, починають навчання програмування. Доцільно виділити окремий етап уроку на ознайомлення із середовищем програмування.

З учнями проводиться бесіда, у процесі якої слід з'ясувати відповідь на такі питання: Як запустити виконуваний файл? Що таке файл? Що таке програма? У якому вигляді зберігається текст програми на комп'ютері? Як перетворити текст програми у виконуваний файл?

Таким чином учні підводяться до розуміння поняття «інтегроване середовище програмування» (IDE), яке включає в себе текстовий редактор, програму перевірки синтаксису, компілятор, редактор зв'язків.

Далі учням слід запропонувати згадати, які середовища програмування вони вже знають (Scratch, IDLE та ін.). У процесі бесіди учні мають з'ясувати, що для кожної мови програмування існує своє середовище програмування. Деякі середовища програмування допускають створення програм на різних мовах (MS Visual Studio, SharpDevelop та ін.).

Далі на прикладі середовища NetBeans 8.0 учнів ознайомлюють з інтерфейсом та функціональними можливостями середовища програмування. Середовище NetBeans є, на нашу думку, оптимальним для навчання програмістів-початківців. Слід запропонувати учням виконати вправи для засвоєння основних дій з документом і фрагментами тексту програми.

Лише після цього приступають до створення програми. Слід нагадати учням, що таке клас і що таке головний метод класу, а потім пояснити їм, у якому місці програми слід вводити виконуваний код.

Перша створювана учнями програма повинна лише виводити привітання. Після введення тексту програми слід запропонувати учням запустити програму на виконання і виправити помилки (якщо вони є).

Навчання об'єктно-орієнтованого програмування слід здійснювати за принципом «від простого до складного». Починаємо з простих задач і поступово ускладнюємо їх.

Існують два варіанти вивчення об'єктно-орієнтованого програмування. За першим варіантом, спочатку вивчається синтаксис мови програмування, а вже потім – створення екранних форм і програм, які використовують об'єктно-орієнтовану парадигму.

За другим варіантом, учні відразу приступають до роботи над екранними формами.

Перший варіант вимагає більших витрат часу, проте учні міцніше засвоюють основні алгоритмічні конструкції. Недолік – для деяких учнів порівняно важко перейти від операцій з консоллю до операцій з формою.

Другий варіант економить час, проте синтаксис основних алгоритмічних конструкцій на мові Java учні вивчають разом з основами об'єктно-орієнтованого програмування, елементами екранних форм. Це призводить до плутанини.

На нашу думку, перший варіант є оптимальнішим. Проте нові шкільні підручники з інформатики орієнтовані, як правило, на другий варіант.

Розглянемо основні методи навчання об'єктно-орієнтованого програмування.

Під час вивчення програмування слід використовувати метод доцільно дібраних задач і метод пошуку помилок у програмах.

Перший метод передбачає добір таких задач, які відповідають темі заняття. Специфікою теми «Основи об'єктно-орієнтованого програмування» є те, що існує лише кілька збірників задач, які передбачають створення програм з застосуванням об'єктно-орієнтованої парадигми. Тому до уроків слід підібрати такі задачі або скласти їх власноруч.

Другий метод передбачає надання учням умови задачі і програми, яка містить помилки. У процесі налагодження програми учні набувають навичок пошуку помилок і їх виправлення, повторюють структуру програми.

На перших заняттях теми слід використовувати фронтальну форму роботи. Тобто, всі учні працюють над абсолютно однаковим завданням. У подальшому, по мірі набуття учнями навичок програмування, слід індивідуалізувати роботу учнів. Більш здібним учням слід задавати складніші задачі, слабшим учням – простіші задачі. Доцільно також підібрати цікаві задачі для здібних учнів, щоб учні створювали програму вдома.

Учитель повинен мати коди програм – розв'язків задач. Це значно прискорює процес підказки учням (якщо вони працюють індивідуально). Наявність готового коду програми прискорює перевірку програми.

Завершується вивчення теми «Основи об'єктно-орієнтованого програмування» створенням проекту – досить складної програми, яка має практичний зміст. Це значною мірою мотивує учнів до створення програми.

Тему проекту учні можуть запропонувати самостійно. Тема проекту повинна бути пов'язана з життям або різними галузями професійної діяльності людини. І в цьому полягає складність. З одного боку, учні можуть знайти проблему, на яку вчитель просто не звернув уваги. З іншого боку – учні можуть запропонувати тривіальну програму (наприклад, архіватор), значна кількість яких поширюється за безкоштовними і платними ліцензіями. Створити програму, яка могла б конкурувати з продуктами, створеними професійними програмістами, важко.

Якщо тему проекту пропонує вчитель, слід сформулювати її так, щоб учням було цікаво працювати над програмою. Доцільно обговорити з учнями тему проекту. Слід уникати нав'язування теми вчителем учневі.

Висновки та перспективи подальших наукових розвідок. Об'єктно-орієнтоване програмування – це сучасна парадигма програмування, яка застосовується у більшості сучасних мов програмування, фахівці з яких користуються найбільшим попитом на ринку праці. Навчання учнів основ об'єктно-орієнтованого програмування має певну специфіку і вимагає дещо інших підходів. Проте успіх у вивченні даної парадигми програмування призводить до того, що учням спрощується вивчення інших мов програмування, які базуються на об'єктно-орієнтованій парадигмі.

Перспективами подальших досліджень у даному напрямі є визначення педагогічних умов успішного формування основних понять і прийомів об’єктно-орієнтованого програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Биков, В.Ю., Спірін, О.М., Пінчук О.П. (2017). Проблеми та завдання сучасного етапу інформатизації освіти. Наукове забезпечення розвитку освіти в Україні: актуальні проблеми теорії і практики (до 25-річчя НАПН України), 191-198. Режим доступу: http://lib.iitta.gov.ua/id/eprint/ 709026
2. Вапнічний, С.Д., Зубик, В.В., Ребрина, В. А. (2010). Факультативний курс з програмування мовою С++ 7-9 класи. Хмельницький: видавничий відділ Хмельницького ОІППО.
3. Жалдак, М.І., Рамський Ю.С. (1976). Елементи програмування. Посібник для вчителів. Київ: Радянська школа.
4. Караванова, Т. П. (2001). Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. Посібник. Київ: ТОВ «Форум».
5. Лапінський, В.В. (2014). Проблема вибору першої мови програмування – сьогоднішнє бачення. Комп'ютер у школі та сім'ї, 1, 14-17.
6. Лисенко, Т. І., Ривкінд, Й.Я., Шакотько, В.В., Чернікова, Л.А. (2011). Інформатика 11 : підруч. Київ: Генеза.
7. Морзе, Н. В. (2004). Методика навчання інформатики: Навч. посіб.: У 4 ч. Жалдак М. І. (ред.). Київ: Навчальна Книга. Ч. IV: Методика навчання основ алгоритмізації та програмування.
8. Рамський, Ю. С., Лукаш І.М. (2003). Методика навчання основ об’єктно-орієнтованого програмування. Об’єктно-орієнтований аналіз. Комп’ютер у школі та сім’ї, 1, 3-9.
9. Рамський, Ю. С., Лукаш, І. М. (2002). Методика навчання основ об’єктно-орієнтованого програмування. Комп’ютер у школі та сім’ї, 1, 3-7; 2, 3-8; 3, 7-13; 4, 17-22; 5, 10-17; 6, 16-21.
10. Ривкінд, Й. Я., Лисенко, Т. І., Чернікова, Л. А., Шакотько, В. В. (2009). Інформатика : 9 кл. : підруч. для заг. навч. закл. Згуровський М. З. (ред.). Київ: Генеза.
11. Ривкінд, Й. Я., Лисенко, Т. І., Чернікова, Л. А., Шакотько, В. В. (2010). Інформатика : 10 кл. : підруч. для заг. навч. закл. ; станд. Згуровський М. З. (ред.). Київ: Генеза.
12. Ривкінд, Й. Я., Лисенко, Т. І., Чернікова, Л. А., Шакотько, В. В. (2011). Інформатика : 11 кл. : підруч. для заг. навч. закл. ; профільний. Згуровський М. З. (ред.). Київ: Генеза.
13. Руденко, В.Д. (2009). Шкільна інформатика: сучасні проблеми та погляд у майбутнє. Комп'ютер у школі та сім'ї, 5, 3-7.
14. Сейдаметова, З. С., Шкарбан, Ф.В. (2011). Сценарний підхід в навчанні інформатики: об’єктність, наочність, креативність. Науковий часопис НПУ ім. Драгоманова, серія № 2. Комп’ютерно-орієнтовані системи навчання: зб. наук. праць. К.: НПУ ім. Драгоманова. 11 (16), 3-11.
15. Семеріков, С. О. (2000). Активізація пізнавальної діяльності студентів при вивченні чисельних методів у об’єктно-орієнтованій технології програмування (дис. ... канд. пед. наук : 13.00.02). Кривий Ріг.
16. Скляр, І. В. (2010). Розвиток алгоритмічного мислення – основна задача курсу інформатики. Комп'ютер у школі та сім'ї, 2, 11-13.
17. Шевчук, П. Г. (2011). Навчання об’єктно-орієнтованому програмуванню в загальноосвітніх навчальних закладах засобами мови С#. Теорія та методика навчання математики, фізики, інформатики: збірник наукових праць. Випуск IX. Кривий Ріг: Видавничий відділ НМетАУ, 595–601
18. Шевчук, П. Г. (2012). Навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів на основі використання мови С# : Методичні рекомендації для вчителів інформатики. Київ. Режим доступу: http://lib.iitta.gov.ua/896/4/Metodrekom_Shevchuk.pdf.

REFERENCES

1. Bykov, V.Yu, Spirin, O.M., Pichuk, O.P. (2017) Problems and tasks of the modern stage of education informatization In Naukove zabezpechennja rozvitku osviti v Ukraïni: aktual'ni problemi teoriï i praktiki (do 25-richchja NAPN Ukraïni). pp. 191-198. Retrieved from: http://lib.iitta.gov.ua/id/eprint/709026
2. Vapnichnij, S.D., Zubik, V.V., Rebrina, V. A. (2010) Optional C ++ Programming Classes 7-9. Khmelnytskyi: vydavnychyi viddil Khmelnytskoho OIPPO.
3. Zhaldak, M.I., Rams'kij Ju.S. (1976) Elements of programming. Teacher's Guide. Kyiv: Radianska shkola.
4. Karavanova, T. P. (2001) Fundamentals of Algorithmization and Programming. 750 tasks with recommendations and examples. Manual. Kyiv: TOV «Forum».
5. Lapins'kij, V.V. (2014) The problem with choosing the first programming language is today's vision. Kompiuter u shkoli ta simi, 1, 14-17.
6. Lisenko, T. I., Rivkind, J.Ja., Shakot'ko, V.V., Chernikova, L.A. (2011) Informatics 11: textbook. K.: Heneza.
7. Morze, N. V. (2004) Methods of teaching computer science: Educ. guide: In 4 parts. K.: Navchalna Knyha. Part IV: Methods of Learning the Basics of Algorithmization and Programming.
8. Rams'kij, Ju. S., Lukash I.M. (2003) Methods of learning the basics of object-oriented programming. Object-oriented analysis. Kompiuter u shkoli ta simi, 1, 3–9.
9. Rams'kij, Ju. S., Lukash I.M. (2002) Methods of learning the basics of object-oriented programming. Kompiuter u shkoli ta simi, 1, 3–7; 2, 3–8; 3, 7–13; 4, 17–22; 5, 10–17; 6, 16–21.
10. Rivkind, J. Ja., Lisenko, T.I., Chernikova, L.A., Shakot'ko, V.V. (2009) Informatics: 9 class: textbook. (edited by M. Zgurovsky). Kyiv: Heneza.
11. Rivkind, J. Ja., Lisenko, T.I., Chernikova, L.A., Shakot'ko, V.V. (2010) Informatics: 10 class: textbook. (edited by M. Zgurovsky). Kyiv: Heneza.
12. Rivkind, J. Ja., Lisenko, T.I., Chernikova, L.A., Shakot'ko, V.V. (2011) Informatics: 11 class: textbook. (edited by M. Zgurovsky). Kyiv: Heneza.
13. Rudenko V.D. (2009) School Informatics: Current Issues and Looking to the Future. Kompiuter u shkoli ta simi, 5, 3-7.
14. Sejdametova, Z. S., Shkarban, F.V. (2011) Scenario approach in teaching computer science: objectivity, clarity, creativity. Naukovyi chasopys NPU im. Drahomanova, seriia №2. Kompiuterno-oriientovani systemy navchannia: zb. nauk. prats. K.: NPU im. Drahomanova. 11 (16), 3–11.
15. Semerikov, S. O. (2000) Enhancement of students' cognitive activity in the study of numerical methods in object-oriented programming technology. (PhD Thesis: : 13.00.02). Kryvyi Rih.
16. Skljar, I. V. (2010) Development of algorithmic – main task of informatics' course. Kompiuter u shkoli ta simi, 2, 11–13.
17. Shevchuk, P. G. (2011) Teaching to object-oriented programming in general education institutions by means of C# language. In Teoriia ta metodyka navchannia matematyky, fizyky, informatyky: zbirnyk naukovykh prats. Vypusk IX. (p. 595–601). Kryvyi Rih: Vydavnychyi viddil NMetAU.
18. Shevchuk, P. G. (2012) Teaching to programming in technological profile grades of general education institutions on basis of C# language: methodical recommendations for teachers of informatics. Kyiv. Retrieved from: http://lib.iitta.gov.ua/896/4/Metodrekom_Shevchuk_.pdf.

Базурин В. Н. Методика обучения основам объектно-ориентированного программирования учеников заведений общего среднего образования.

В статье раскрываются особенности обучения учащихся общеобразовательных школ основ объектно-ориентированного программирования. В настоящее время объектно-ориентированная парадигма программирования используется в большинстве языков

программирования. Методика обучения объектно-ориентированного программирования имеет определенные отличия от методики обучения структурного программирования. Пути решения данной проблемы было найдено отечественными учеными 15-20 лет назад, однако за это время сами объектно-ориентированные языки программирования продолжали развиваться.

Одним из путей информатизации образования Украины признано усиление содержательной линии программирования. Применение содержательной линии программирования должно способствовать не только формированию навыков программирования, но и формированию алгоритмического мышления, повышению компьютерной грамотности учащихся.

В основе объектно-ориентированной парадигмы программирования лежит понятие объекта. Все числовые значения определяются не через переменные, а через свойства соответствующих объектов. Действия, которые выполняет программа, задаются не через функции и процедуры, а с помощью методов соответствующих объектов. В статье раскрыты особенности формирования понятия объект, класс, свойство, метод, наследование и других, которые являются фундаментальными понятиями объектно-ориентированного программирования. Предложенные автором приемы объяснения основных вопросов объектно-ориентированного программирования сопровождаются соответствующими примерами.

Перспективами дальнейших научных исследований в данном направлении является определение педагогических условий успешного формирования основных понятий и приемов объектно-ориентированного программирования.

Ключевые слова: *объектно-ориентированное программирование, учреждения общего среднего образования, ученики, математические способности, математическое образование, информатизация образования, язык программирования, основы программирования.*

Bazurin V. M. Methods of teaching the basics of object-oriented programming for students of secondary schools.

The article reveals the peculiarities of teaching students of the elementary schools of the basics of object-oriented programming. Currently, the object-oriented programming paradigm is used in most programming languages. The methodology of object-oriented programming has some differences from the teaching of structural programming. Ways to solve this problem were found by domestic scientists 15-20 years ago, but during this time the object-oriented programming languages themselves continued to develop.

One of the ways of informatization of the education of Ukraine is acknowledged strengthening of the content line of programming. The use of a content line of programming should contribute not only to the formation of programming skills, but also to the formation of algorithmic thinking, to the increase of computer literacy of students.

The object-oriented programming paradigm is based on the concept of the object. All numeric values are determined not by the variables but by the properties of the corresponding objects. The actions performed by the program are not determined by the methods of the corresponding objects, not through functions and procedures. The article describes the peculiarities of forming the concept of object, class, property, method, inheritance and others, which are fundamental concepts of object-oriented programming. The techniques offered by the author for explaining the basic issues of object-oriented programming are accompanied by relevant examples.

Prospects for further scientific research in this area are to determine the pedagogical conditions for the successful formation of basic concepts and techniques of object-oriented programming.

Key words: *object-oriented programming, general secondary education institutions, students, mathematical abilities, mathematical education, informatization of education, programming language, basics of programming.*