Відповідно до таблиці 1 кожний із застосунків має свої переваги, про те слід зауважити, що деякі з них мають і недоречності. Так, Visual Studio Code досить довго завантажується. Notepad++ не має IDE (Integrated development environment). Недоліком Sublime є незручність роботи з менеджером плагінів, так як частина з них працює некоректно. Vim один з найскладніших для вивчення інструментів розробки. Високий поріг входження вимагає від веброзробників значних витрат часу на запам'ятовування його особливостей, команд, плагінів. Застосунок Eclipse має неабияку складність настройки IDE, а також перевантаженість для розробки простих вебсайтів на HTML і CSS.

Таким чином, для навчання у старшій школі доцільно обирати застосунки Visual Studio Code та Notepad++. Оскільки, Visual Studio Code має частину функціоналу IDE, інтегрованого середовища розробки, потужної програми, що містить, окрім коду, ще ряд механізмів, що дозволяють проводити аналіз коду, запуск його і налагодження. Застосунок Notepad++ поширюється як безкоштовне програмне забезпечення, його репозиторій є доступним в GitHub, також, він є перевіреним і усталеним інструментом багатьох веброзробників.

Список використаних джерел

1. «Інформатика». Навчальна програма вибірково-обов'язкового предмету для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту).

2. Публічна платформа Stack Overflow. URL: https://insights.Stackoverflow.com/survey/2020#technology-development-environments-and-tools-web-developers. (дата звернення 12.02.2021).

3. Відео курси по програмуванню. URL: https://itvdn.com/ru. (дата звернення 102.02.2021).

ФОРМУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВНОГО МИСЛЕННЯ У ЗДОБУВАЧІВ ОСВІТИ ЗАКЛАДІВ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ

Карабін Оксана Йосифівна
кандидат педагогічних наук, доцент кафедри інформатики та методики її навчання,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
karabin@tnpu.edu.ua

Халупа Наталя Богданівна
магістрантка спеціальності 014.09 Середня освіта (Інформатика),
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
babij_nb@fizmat.tnpu.edu.ua

Важливість упровадження об'єктно-орієнтовного програмування для здобувачів освіти у закладах загальної середньої освіти є актуальною проблемою дослідження нині.

Сучасний світ розвивається в цифровому середовищі, тому зростає вагомість фахівців. Важливо правильно і доступно пояснити здобувачам освіти даний розділ в їх курсі шкільної інформатики та розпочати формувати в них об'єктно-орієнтовне мислення.

Вперше термін «об'єктно-орієнтовне мислення» вводить автор книги «Об'єктно-орієнтовний підхід» Вайсфельд Метт.

Об’єктно-орієнтоване програмування – це фундамент усіх мов програмування, включаючи C++, Java, C#, Visual Basic, .NET, Ruby і Objective-C. Крім того, об’єкти лежать в основі багатьох вебтехнологій, наприклад, JavaScript, Perl і PHP.

Об’єктно-орієнтоване програмування дозволяє легші методики проєктування, через експорт коду і його повторне використання, проте для того, щоб працювати з цим підходом, необхідно розвивати своє алгоритмічне мислення. Розробники-початківці у сфері об’єктно-орієнтованого програмування, не зобов’язанні працювати з однією визначеною мовою програмування (Objective-C, VB .NET, C++, C#,NET або Java) або моделювати (UML), а навпаки розвивати у собі, те що і автор книги називає об’єктно-орієнтованим мисленням.

Незважаючи на те, що технології програмування змінюються та еволюціонують, об’єктно-орієнтовані концепції залишаються – при цьому неважливо, яка саме є платформа.

Розглянувши об’єктно-орієнтовну парадигму або її ще називають ієрархічною парадигмою, видно, що це допомагає у створенні проєктів багатьом фахівцям. Оскільки, алгоритми, реалізовані в процедурному програмуванні, є конкретними. Будь-яка модифікація – це вже новий алгоритм і таким чином кількість процедур і функцій, що знаходяться у використанні, надмірно зростає.

Модульне програмування групує алгоритми в модулі, одночасно інкапсулюючи структури даних. Наступним кроком є побудова ієрархій модулів або класів. Зазначимо ієрархії модулів або класів:

перша ієрархія – частина чогось. Наприклад, грань є частиною многогранника, ребро – частиною грані, вершина – частиною ребра;

друга ієрархія – узагальненням або конкретизацією. Наприклад, овал і многокутник служать конкретизацією плоскої фігури, коло – конкретизацією овалу, чотирикутник – конкретизацією многокутника, подальшими конкретизаціями чотирикутника можуть служити паралелограм, прямокутник, ромб, квадрат.

Відтак, квадрат, ромб, прямокутник є повноцінними паралелограмами дозволяє їм користуватися усіма програмними засобами, створеними для паралелограма, паралелограм в свою чергу є повноцінним чотирикутником і так далі. Цей принцип, відомий під назвою «знову вживаний» – став одним з найважливіших досягнень об’єктно-орієнтованої парадигми. Знову вживаючи вже існуюче програмне забезпечення в більш конкретизованих умовах, дописується лише та його частина, яка стосується особливостей наявної конкретизації. Цей принцип називається «дописування програм».

Тут дуже важливо, щоб кожен здобувач освіти здатен був продемонструвати: об’єктно-орієнтоване мислення, застосування об’єктно-орієнтованих мов програмування та під час проєктування програмних продуктів, програмної реалізації алгоритмів розв’язання задач.

Важливо виробити об’єктно-орієнтоване мислення і згодом перейти до об’єктно-орієнтованої розробки на визначеній мові програмування.

Об’єктно-орієнтоване мислення, в якому цінність речей і навіть їх здатність діяти самостійно не залежать від зв’язку з розробниками. Це мислення базується на принципах, які лежать набагато більш базовому рівні, ніж шаблони дизайну.

Ресурси, які затрачаються на формування такого мислення у здобувачів освіти вимагає демонстрації як і готових проєктів так і покрокового та самостійних розробок.

Будь-який проєкт ділиться на п’ять основних етапів:

– постановка задачі;

– побудова математичної моделі;

– побудова алгоритму;

– реалізація;

– тестування.

На другому та третьому етапах застосовується об’єктно-орієнтовне мислення. Саме на даних етапах мислення характеризується категоріями та об’єктами. Завдання ділиться на підзавдання. І кожна частина проєкту виконується та тестується окрему. У такому випадку зменшується ймовірність помилок у коді програми. Працюючи із підзавданнями не важливо для програміста, що у середині даного підалгоритму, а лише задаються вхідні дані та знаючи наперед, перевіряються вихідні дані. Власне тут, уже скориставшись готовим модулем чи процедурою, яку експортовано, не шукаються в ньому помилки, що і зменшує витрату зусиль на написання аналогічного алгоритму та масивність коду програми. Використання таких конструкцій будує ієрархію проєктів.

Таким чином, об’єктно-орієнтовне мислення – це спосіб знаходження найпростіших способів розв’язання навіть і найважчих завдань та проєктів у ієрархічному програмуванні. Основними групами даного мислення є – категорії та об’єкти. Та саме тут і формується ієрархія коду та програми.

Список використаних джерел

1. Вайсфельд Метт. Об’єктно-орієнтоване мислення. СПб .: Пітер, 2014. 304 с.

2. Милосавеч О. Вступ до основ програмування. URL: http://wwwsashamilashka.blogspot.com/2011/06/blog-post_05.html (дата звернення 15.03.2021).

3. Карабін О. Й., Шуль М. В. Формування цифрових компетентностей здовувачів освіти в контексті нової української школи. *Інноваційна педагогіка.* Одесса, В. 29. Т. 1. 2020. С. 140–144.