МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна
Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра електромеханічних та комп'ютерних систем

До захисту допущено
Завідувач кафедри
_____ Інна НЕФЬОДОВА
(підпис) (ім'я, прізвище)
«07» грудня 2024 року

КВАЛІФІКАЦІЙНА РОБОТА (ПРОЄКТ)

рівень вищої освіти другий (магістерський)

спеціальність 015.39 Професійна освіта (Цифрові технології)

освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

тема «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти»

Виконав(ла)

здобувач(ка) групи БД-К23мг
(шифр групи)

Ілля КОРОБЧУК
(ім'я, прізвище) _____ (підпис)

Керівник роботи
к.т.н., доц. Павло ЧИКУНОВ
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Рецензент роботи
к.пед.н., доц. Наталія ЛОГІНОВА
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Консультант
к.пед.н., доц. Юлія БОБРИКОВА
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Засвідчую, що у цій роботі немає цитат та вилучень з праць інших авторів без відповідних посилань
здобувач (ка) _____ (підпис)

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет/ННІ Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра Електромеханічних та комп'ютерних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 015.39 Професійна освіта (Цифрові технології)
Освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____ Інна НЕФЬОДОВА
(підпис) (ім'я, прізвище)
«08» жовтня 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Коробчук Ілля Петрович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти
керівник роботи Чикунов Павло Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «08» жовтня 2024 року № 5101-5/3236

2. Строк подання здобувачем роботи «02» грудня 2024 р.

3. Перелік питань, які потрібно розробити: Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти. Характеристика об'єктів галузі: стан і стратегії розвитку. Вимоги до кадрового забезпечення об'єкту галузі. Методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти.

4. План роботи

№ з/п	Назви етапів роботи
1	Огляд літературних джерел, нових розробок, опублікованих даних та іншої інформації, пов'язаної з темою роботи.
2	Дослідження теоретичних підходів до актуальності професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти».
3	Характеристика об'єктів галузі: стан і стратегії розвитку.
4	Розробка методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Паралельні обчислення мовою C#» у закладах вищої освіти.
5	Розробка вимог до кадрового забезпечення об'єкту галузі
6	Оформлення першого варіанту тексту, подання його на ознайомлення науковому керівнику
7	Усунення недоліків, написання остаточного варіанту тексту, оформлення дипломної роботи
8	Подання роботи на кафедру, перевірка на плагіат та зовнішнє рецензування роботи
9	Захист дипломної роботи у ЕК

5. Дата видачі завдання «08» жовтня 2024 р.

Здобувач(ка) _____ (підпис) Ілля КОРОБЧУК (ім'я, прізвище)

Керівник роботи _____ (підпис) Павло ЧИКУНОВ (ім'я, прізвище)

РЕФЕРАТ

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу освітніх ресурсів.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Виконано аналіз характеристик об'єктів галузі аналіз та рефакторинг коду. Виконана постановка 3 нових лабораторних робіт. Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі в умовах цифрової трансформації суспільства.

Розроблено дидактичний проект консультативного заняття з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти.

За основними результатами дослідження виконана публікація тези доповіді на VIII міжнародній НПК «Студенти та молодь – для майбутнього країни» (Бахмут-Харків, 14-15 листопада 2024 р.).

Робота складається із вступу, чотирьох розділів, висновків, бібліографічного списку, що містить 56 джерел, два додатка, 6 таблиць, 13 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ЛАБОРАТОРНИЙ ПРАКТИКУМ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

The aim of the study is to theoretically substantiate and partially test the methodology for the professional training of digital technology specialists to teach the educational module "Parallel Computing in C#" in higher education institutions.

The object of the study is the process of professional training of digital technology specialists to teach the educational module "Parallel Computing in C#" in higher education institutions.

The subject of the study is the methodology for professional training of digital technology specialists to develop a set of educational resources.

The system of professional training for digital technology specialists to teach the educational module "Parallel Computing in C#" in higher education institutions has been characterized.

An analysis of the characteristics of industry objects, as well as code analysis and refactoring, has been performed. Three new laboratory assignments have been developed. An analysis of staffing requirements for IT sector objects under the conditions of digital transformation of society has also been conducted.

A didactic design for a consultation session on the topic "Thread Synchronization and Locking" within the discipline "Software Engineering" for higher education students has been developed.

The main research results were published as a conference abstract at the VIII International Scientific and Practical Conference "Students and Youth for the Future of the Country"(Bakhmut-Kharkiv, November 14–15, 2024).

The work consists of an introduction, four chapters, conclusions, a bibliography of 56 sources, two appendices, 6 tables and 13 figures.

PROFESSIONAL TRAINING, LABORATORY PRACTICE, PARALLEL COMPUTING, STAFFING, METHODOLOGICAL DEVELOPMENT.

ЗМІСТ

ВСТУП

Сучасний ринок праці вимагає від фахівців у сфері професійної освіти гнучкості, адаптивності та вміння працювати з новітніми цифровими технологіями. Концепція неперервної підготовки фахівців спеціальності 015 Професійна освіта (Цифрові технології) спрямована на забезпечення високої якості підготовки, відповідної до сучасних вимог і викликів. Перед закладами вищої освіти постає завдання підготовки фахівця, спроможного задовольняти запити нинішнього суспільства. Тож інтенсивність освітнього процесу й активізація пізнавальної діяльності здобувачів вищої освіти потребують інших навчальних форм, що дадуть змогу у стислий термін опанувати значним обсягом навчального матеріалу.

Актуальність теми дослідження тісно пов'язана з післядипломною професійною освітою та розробкою комплексу найважливіших проблем становлення і управління системою професійної підготовки фахівців з цифрових технологій, який працює в екстремальних умовах економічної кризи і потребує висококваліфікованого фахівця.

Аналіз психолого-педагогічних джерел засвідчує, що навчання та концептуальні основи післядипломної професійної освіти досліджуються такими вченими, як В. Арешонков, О. Бабкова, В. Вербець, Л. Гончаренко, С. Загородній, Н. Клокар, В Кремень, Л. Лук’янова, Н. Ничкало, О. Худенко, І. Ящук та ін.; різні аспекти професійної підготовки майбутніх інженерів-педагогів досліджують Е. Абільтарова, І. Васильєва, Н. Волкова, Н. Брюханова, Р. Горбатюк, О. Коваленко, М. Лазарєв, В. Кулешова, В. Мальована, С. Хоменко, Л. Штефан та ін.

Необхідність дослідження й вирішення проблеми та вивчення педагогічного досвіду показали зумовлені наявні суперечності між:

– об’єктивною потребою суспільства й українського освітнього простору в підготовлених фахівцях, які мають високий рівень професійної підготовки, та неспроможністю закладів вищої й післядипломної професійної

освіти нагально задовольнити цю потребу в повному обсязі;

– усвідомленням сучасною наукою та практикою навчальної й педагогічної діяльності та недостатнім рівнем розвитку умінь усіх суб’єктів освітнього простору.

Необхідність подолання виявлених суперечностей й обумовила тему дослідження «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти».

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

У зв'язку з поставленою метою, в роботі вирішуються такі завдання.

1. Визначити ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

2. Теоретично обґрунтувати та розробити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Об'єктом дослідження є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Предметом дослідження є методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

В ході написання роботи використані такі методи дослідження:

– теоретичні: аналіз психолого-педагогічної, методичної та наукової літератури з метою визначення психолого-педагогічних підходів, зіставлення різних поглядів на досліджувану проблему та концептуальних основ післядипломної професійної освіти;

– емпіричні: анкетування, спостереження, співбесіди та самооцінки для діагностування з метою вивчення стану сформованості професійних компетентностей у здобувачів вищої освіти.

Наукова новизна одержаних результатів дослідження полягає:

– в уточненні сутності поняття «професійна підготовка» фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти, яка розуміється як процес, спрямований на набуття фахівцями професійних компетентностей з удосконалення здатності до постійного оновлення знань та самоактуалізації за програмами післядипломної професійної освіти, що дозволяє їм освоїти новий вид професійної діяльності;

– подальшого розвитку набули принципи навчання (органічної єдності й наступності базової та післядипломної професійної освіти; гуманізму; демократизму; іманентності розвитку; вільного розвитку фахівця; свободи вибору).

Теоретичне та практичне значення одержаних результатів полягає в розробці методики професійної підготовки фахівців з цифрових технологій для у закладах вищої освіти, обґрунтуванні принципів навчання (органічної єдності й наступності базової та післядипломної освіти; гуманізму; демократизму; іманентності розвитку; вільного розвитку фахівця; свободи вибору); виділено пріоритетні напрями удосконалення професійних компетентностей для фахівців з цифрових технологій.

Матеріали дослідження використовувались при підготовці здобувачів вищої освіти зі спеціальності 015 Професійна освіта (Цифрові технології) у Бахмутському навчально-науковому професійно-педагогічному інституті ХНУ імені В. Н. Каразіна.

Апробація результатів дослідження: за основними результатами дослідження виконана доповідь на міжнародній науково-практичній конференції здобувачів вищої освіти та молодих учених «Студенти та молодь для майбутнього країни» (Бахмут-Харків, 17 листопада 2024 р.).

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ МОВОЮ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Сучасна ситуація в умовах ринкових відносин вимагає постійного розвитку і саморозвитку людини як суб'єкта власної життєдіяльності. Засобом розв'язання цього завдання є концепція неперервної освіти, яка визначає нові підходи до проектування системи освіти, до змісту педагогічного процесу, наступності його рівнів. Ідея неперервної освіти – одна з найбільш прогресивних ідей початку XXI століття. Її основна суть – постійне творче удосконалення, оновлення і розвиток кожної людини протягом усього життя.

Ніколи ще зростання освіти в цілому, і вищої освіти зокрема, не було таким необхідним суспільству для його нормального функціонування, розвитку і економічного, суспільного, культурного, духовного та політичного прогресу, як в даний час.

Цифрові технології надають великі можливості для розвитку професійних навичок та інтелектуального потенціалу у фахівців з цифрових технологій.

Підвищення попиту на фахівців у галузі цифрових технологій інтенсивно прогресує, і відповідно запити ринку праці стають визначальним орієнтиром для вдосконалення системи підготовки дипломованих фахівців, а відтак задають основні вимоги щодо їхніх знань, умінь та навичок.

Ключовою складовою фахівців з цифрових технологій є компетентний фахівець[47].

У багатьох науково-теоретичних і науково-методичних роботах проаналізовано суть компетентнісного підходу (О. Браславська, А. Вербицький, І. Зимня, О. Коберник, П. Лузан, О. Муковіз, О. Овчарук, О. Пометун, Г. Терещук, А. Хуторський та ін.); проблеми формування ключових компетентностей (В. Байденко, М. Бочарнікова, В. Щербакова та ін.);

застосування цифрових та інформаційних технологій в освіті (О. Авраменко, В. Биков, Т. Бодненко, Т. Вакалюк, І. Войтович, А. Гедзик, Ю. Горошко, А. Гуржій, М. Жалдак, Л. Карташова, В. Лапінський, Л. Макаренко, Н. Морзе, Ю. Рамський, С. Семеріков, О. Спірін, Г. Ткачук, Ю. Триус, В. Франчук, С. Яшанов та ін.); різні аспекти професійної підготовки майбутніх інженерів-педагогів (Е. Абільтарова, І. Васильєва, Н. Волкова, Н. Брюханова, Р. Горбатюк, О. Коваленко, В. Кулешова, В. Мальована, М. Лазарєв, С. Хоменко, Л. Штефан та ін.); процес формування комунікативної компетентності та її складових у здобувачів освіти закладів інженерної та інженерно-педагогічної освіти (Т. Калініченко, К. Ковальова, В. Кручек та ін.).

Незважаючи на значну кількість праць, присвячених проблемам компетентнісного підходу, недостатньо обґрунтованими залишаються ключові поняття підходу саме у професійній освіті.

Аналіз професійних компетентностей фахівців показує, що вони повинні мати сформовані організаторські здібності, вміти брати на себе відповідальність за прийняте рішення, володіти прагненням до вдосконалення своїх комунікативних якостей та практичних навичок, професійних знань, оцінювати соціальні процеси, визначати місце та роль у них своїй професійної діяльності та ін. Разом із цим у педагогічній теорії та практиці недостатньо розроблено проблему яка б враховувала сучасні тенденції розвитку професійних компетентностей фахівців з цифрових технологій [37, 40].

На основі аналізу фахових компетентностей установлено, що недостатньо уваги приділяється розвитку професійних компетентностей фахівців з цифрових технологій, а саме: прагненню до постійного підвищення освітнього та наукового рівня, актуалізації й реалізації власного особистісного потенціалу, прагнення до саморозвитку.

Тому роль процесу формування професійної компетентності фахівців з цифрових технологій у процесі професійної підготовки, що виступає як складова професійної культури, надзвичайно висока, оскільки саме вона

визначає соціальну орієнтацію особистості в контексті професійної діяльності, що відображає всі її компоненти: сукупність знань, умінь, сформовану особисту та професійну позицію, особисту та професійну самооцінку, що визначає активне, творче ставлення до діяльності; сукупність особистісних та професійно-важливих якостей особистості, що визначають успішність професійної діяльності спеціаліста, його самореалізацію [38].

Із позицій компетентнісного підходу рівень освіченості та кваліфікованості визначається здатністю розв'язувати проблеми різної складності на основі наявних знань. Компетентнісний підхід не заперечує значення знань, але він акцентує увагу на здатності використовувати здобуті знання. Тобто, за такого підходу, важливим стає не наявність внутрішньої організації знань, а здатність застосовувати компетентності в професійній діяльності [40].

Педагогічними умовами формування професійно-важливих якостей фахівців у процесі професійної підготовки є поєднання у собі високого професіоналізму, культури, соціальної відповідальності. Основою є особистісний підхід, спрямований на розвиток активності фахівців з цифрових технологій і виявляється у формуванні адекватної особистісної та професійної самооцінки фахівця [39].

Все це обумовлює необхідність професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Педагогічна система формування професійної компетентності фахівця у процесі професійної підготовки включає: підсистему цінностей, підсистему принципів, підсистему умов, підсистему критеріїв та показників відповідного змісту, форми та методи навчання в системі вищої освіти; методи використання самооцінки з вивчення рівня сформованості знань, умінь, досвіду, ставлення до професійної діяльності, що освоюється; технологію корекції процесу формування самооцінки фахівців на основі створення умов для прояву активності у навчальній та практичній діяльності; здійснюється

поєднання навчальної та практичної діяльності фахівців як найважливішого фактору формування адекватної особистісної та професійної самооцінки [43].

Завдання вдосконалення професійної підготовки фахівців з цифрових технологій полягає в корекції та доповненні їхньої професійної освіти, а також у розвитку економічно стабілізованої системи професійної освіти, оснащеної сучасною методологією та реалізує гнучкі форми навчання. Система професійної підготовки, з одного боку, повинна бути орієнтована на потреби людини, зацікавленої у прояві та розвитку своїх здібностей, а з іншого боку, новий вид професійної діяльності служить гарантована мобільність фахівця. Вимоги ринкової економіки до підготовки кваліфікованих та конкурентоспроможних фахівців виявляються у змішанні акценту з формування у процесі навчання вузькопрофесійних умінь та навичок на розвиток фахових та дослідницьких компетентностей здобувачів освіти, вироблення потреби у них до самоосвіти, навчання протягом усього життя [39, 46].

Для підвищення ефективності освітнього процесу в системі професійної підготовки і відповідності сучасним вимогам модернізації освіти, потрібно враховувати сучасні інтерпретації принципу неперервності з концептуальним осмисленням специфічної природи нового виду професійної діяльності.

Отже, професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти потребує вирішення проблеми, яку ми вбачаємо через теоретичне обґрунтування методики професійної підготовки.

Висновки до розділу

Узагальнений матеріал надав можливості зробити такі висновки.

У кваліфікаційній роботі відповідно до мети і завдань розкрито стан наукової проблеми.

У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З’ясовано, що, професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА ОБ'ЄКТІВ ГАЛУЗІ: СТАН І СТРАТЕГІЇ РОЗВИТКУ

2.1 Аналіз силабусів освітніх компонент

Виконано аналіз освітніх компонент (ОК) вітчизняних закладів вищої освіти, присвячених вивченню процесів організації паралельних обчислень, з метою встановлення напрямків подальших досліджень.

Силабус ОК «Технології розподілених систем та паралельних обчислень» Придніпровської державної академії будівництва та архітектури, під авторством Заврак М.В. та Гуркліс І.В. [27], відносить компоненту до переліку обов’язкових навчальних дисциплін, дозволяє розширити цикл дисциплін з програмування для підготовки бакалаврів, а також надати їм додаткові зна.

Мета вивчення навчальної дисципліни «Технології розподілених систем та паралельних обчислень» полягає у забезпеченні студентів необхідними теоретичними знаннями та практичними навичками для ефективного застосування основ паралельних і розподілених обчислень у суміжних дисциплінах та майбутній професійній діяльності. Дисципліна спрямована на розвиток умінь організовувати ефективне розв'язання масштабних задач на комп’ютерах із паралельною архітектурою.

Лабораторний практикум охоплює знайомство з основними методами, алгоритмами та інструментами паралельної й розподіленої обробки даних із використанням технології паралельного програмування MPI.

Силабус ОК «Технології розподілених систем та паралельних обчислень» Військового інституту телекомунікацій та інформатизації, під авторством Редзюка Є.В. [28], відносить обов’язкову компоненту до циклу загальної підготовки.

Мета полягає у вивченні принципів паралельного програмування для багатопроцесорних обчислювальних систем, а також застосуванні «хмарних»

технологій для розв’язання практичних задач в інформаційно-обчислювальних системах військового призначення.

До складу практичних навичок входять:

– технології та інструменти реалізації паралельних і розподілених обчислень;

– паралельне програмування потоків;

– організація багатопоточності (використання бібліотеки concurrent, ForkJoin Framework);

– огляд сучасних платформ для «хмарних» обчислень.

Силабус ОК «Технології розподілених систем та паралельних обчислень» Національного університету водного господарства та природокористування, під авторством Парфенюка О.В. [29], метою вивчення формує здобуття знань про організацію розподілених систем і паралельних обчислень, а також формування практичних навичок застосування цих знань для розробки ефективних паралельних програм із використанням сучасних технологій паралельного та розподіленого програмування.

Практикум спрямований на освоєння здобувачами навичок розробки програмного забезпечення із застосуванням паралельних і розподілених обчислень для різних типів високопродуктивних, паралельних або розподілених обчислювальних систем, зокрема:

– створення паралельних потоків і процесів із використанням Windows API;

– програмування за допомогою технології OLE;

– розробка додатків на основі серверів автоматизації Word і Excel;

– створення COM-об’єктів за допомогою бібліотеки Active Template Library в IDE Visual Studio;

– використання процесів у сучасних мовах і бібліотеках програмування (Java, C#, Win32, MPI).

Силабус вибіркової ОК «Розподілені системи та паралельні обчислення» Національного технічного університету «Київський політехнічний інститут

імені Ігоря Сікорського» під авторством Корочкин О.В. [30], метою ставить підготовку висококваліфікованих фахівців, які володіють базовими поняттями, термінами, структурою та властивостями паралельних обчислювальних систем, знають методи створення й етапи проєктування таких систем, володіють програмними засобами їх реалізації та способами підвищення їх ефективності.

Після виконання завдань лабораторного практикуму здобувачі зможуть:

– створювати програмні засоби для паралельних систем;

– реалізовувати арифметичні операції з використанням OpenMP на різній кількості обчислювальних потоків;

– розподіляти обчислення між потоками у системах із загальною пам’яттю;

– розробляти програми для систем із розподіленою пам’яттю на основі передачі повідомлень;

– організовувати обчислення у системах із розподіленою пам’яттю;

– створювати програми для виконання на графічних процесорах.

Силабус ОК «Технології розподілених систем та паралельне обчислення» Національного університету «Одеська юридична академія» під авторством Чикунова П.О. [31], метою вивчення визначає ознайомлення здобувачів вищої освіти з принципами організації високопродуктивних обчислень за допомогою технологій розподілених комп’ютерних систем, зокрема паралельних і хмарних, які застосовуються для вирішення складних завдань у різних прикладних галузях, що вимагають високої швидкості обробки даних та використання великих обсягів оперативної пам’яті.

Мета лабораторного практикуму полягає у набутті здобувачами навичок побудови паралельних алгоритмів, опануванні багатопоточності в паралельних обчисленнях мовою C# із використанням механізмів синхронізації та блокування потоків.

Аналіз запропонованих у силабусах підходів до паралельного програмування показав, що у першу чергу необхідно враховувати конкретні

потреби розробника прикладного проекту, доступних ресурсів і його знань у галузі паралельного програмування. Деякі запропоновані підходи, зокрема MPI, OpenMP та OpenCL, обмежені використанням мови C або C++, тому далі не розглядаються.

У подальшому матеріалі наведено етапи розробки нового лабораторного практикуму «Паралельні обчислення мовою C#», який акцентує увагу здобувачів на організація паралельних обчислень з використанням класу Thread та оператора lock [1, 6, 9], організації синхронізації потоків засобами Mutex [11, 14-16] та Semaphore [21-23].

2.2 Постановка лабораторної роботи «Організація паралельних обчислень з використанням класу Thread та блокуванням потоків lock»

Мета виконання роботи: отримання навичок реалізації обчислювальних алгоритмів з використанням засобів однопоткового та багатопотокового програмування з ексклюзивним блокуванням потоків Lock.

Постановка завдання лабораторної роботи.

1. Згідно індивідуальному варіанту обрати предметну область для реалізації паралельних обчислень. Реалізувати на мові C# консольну однопотокову програму, що виконує рішення задачі предметної області з виконанням таких умов:

– початкова ініціалізація масивів, якщо це необхідно, виконується генератором випадкових величин один раз, до тестових запусків;

– розмір масивів або діапазонів пошуку підібрати таким чином, щоб середній час виконання був не менш ніж 60 секунд;

– запуск обчислень необхідно виконувати у циклі 3 рази, з подальшим обчисленням середнього часу виконання;

– фіксувати час виконання обчислень та візуалізувати результатів обчислень у консолі.

2. Реалізувати багатопотокову версію програми п.1 з використанням

потоків Thread. Всі попередні умови зберігаються, окрім середнього часу виконання. Передбачити ексклюзивне блокування потоків при доступі до спільних ресурсів за допомогою оператора lock. Виконати окремі запуски програми для різної кількості потоків. Максимальна кількість потоків повинна дорівнювати кількості логічних процесорів на ПК. Порівняти результати, зробити висновки про вплив кількості потоків на продуктивність паралельних обчислень.

3. Виконати порівняння середнього часу виконання багатопотокової та однопотокової версії програм. Зробити висновки про покращення або погіршення продуктивності.

4. У звіті навести блок-схеми, програмні коди та скріншоти роботи однопотокової та багатопотокової версій програми.

Приклад виконання лабораторної роботи. Написати однопотокову та багатопотокову версії програми обчислення квадрату довжини *n*-вимірного вектору із використанням ексклюзивного блокування.

Багатопотокова версія програми відрізняється від однопотокової розбиттям обчислювального процесу на кілька потоків і організацію синхронізації доступу до змінної *result*, яка є спільним ресурсом. Для кожного потоку визначено частину масиву, яку він обробляє. Для запобігання одночасному доступу кількох потоків до змінної result використано оператор lock і об'єкт lockObj. Результати обчислень виводяться у консоль.

На рис. 2.2-2.3 наведено блок-схему однопоточного та багатопоточного алгоритмів обчислення квадрату довжини *n*-вимірного вектору.

Програмний код однопотокової версії програми на мові C#:

```
using System.Diagnostics; // Для використання Stopwatch
using System.Text;
using System.Threading;

class Program
{
    static int n = 1800000000;                // кількість елементів вектора
    static double[] vector = new double[n]; // масив для зберігання значень вектора
    static double result = 0;                // обчислення квадрата довжини вектора

    // головна точка входу у програму
    static void Main()
```

```
    {
        // генератор випадкових величин
        Random random = new Random();
        double totalTime = 0; // Загальний час виконання

        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер
        Console.WriteLine($"Генерація елементів масиву...");

        // Заповнюємо вектор випадковими числами
        for (int i = 0; i < n; i++)
        {
            vector[i] = random.NextDouble() * 10;
        }
        Console.WriteLine($"Масив сформовано... \nТестові запуски розпочато...");
        // Виконуємо цикл для трьох тестових запусків
        for (int run = 1; run <= 3; run++)
        {
            result = 0; // Скидаємо результат перед кожним запуском

            // Створюємо та запускаємо Stopwatch для вимірювання часу
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            CalculateSquareSum(0, n);  // Запуск обчислення

            // Зупиняємо Stopwatch
            stopwatch.Stop();

            // Додаємо час виконання до загальної суми
            totalTime += stopwatch.ElapsedMilliseconds;

            // Виводимо результат обчислень для кожного запуску
            Console.WriteLine($"Запуск {run}:Сума квадратів елементів " +
                $"вектора = {result:F3}; " +
                $"час виконання={(float)stopwatch.ElapsedMilliseconds /1000} сек.");
        }
        // Обчислюємо середній час виконання
        double averageTime = totalTime / 3;
        // Виводимо середній час виконання
        Console.WriteLine($"Середній час виконання: " +
            $"{Math.Round(averageTime / 1000, 3)} сек.");
    }
    // Метод для обчислення суми квадратів елементів вектора
    static void CalculateSquareSum(int start, int end)
    {
        double localResult = 0;

        for (int i = start; i < end; i++)
        {
            localResult += vector[i] * vector[i];
        }
        result += localResult;
    }
}
```

```
Microsoft Visual Studio Debug Console
Генерація елементів масиву...
Масив сформовано...
Тестові запуски розпочато...
Запуск 1:Сума квадратів елементів вектора = 59998153450,013; час виконання = 71,384 сек.
Запуск 2:Сума квадратів елементів вектора = 59998153450,013; час виконання = 40,745 сек.
Запуск 3:Сума квадратів елементів вектора = 59998153450,013; час виконання = 116,709 сек.
Середній час виконання: 76,279 сек.
```

Рисунок 2.1 – Результати роботи однопотокової програми

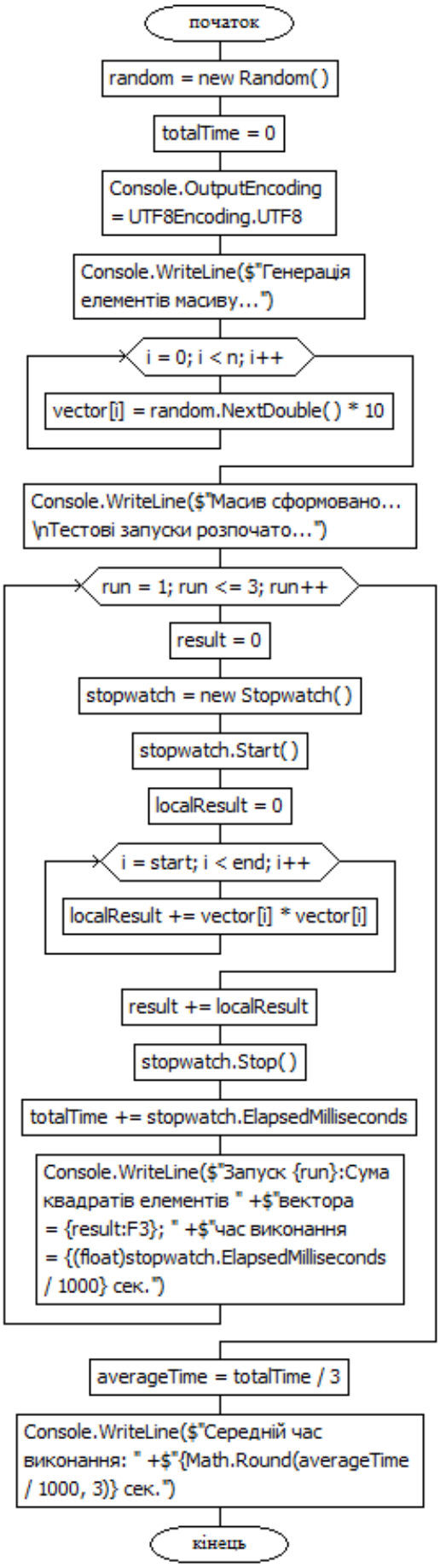

Рисунок 2.2 – Блок-схема однопоточного алгоритму

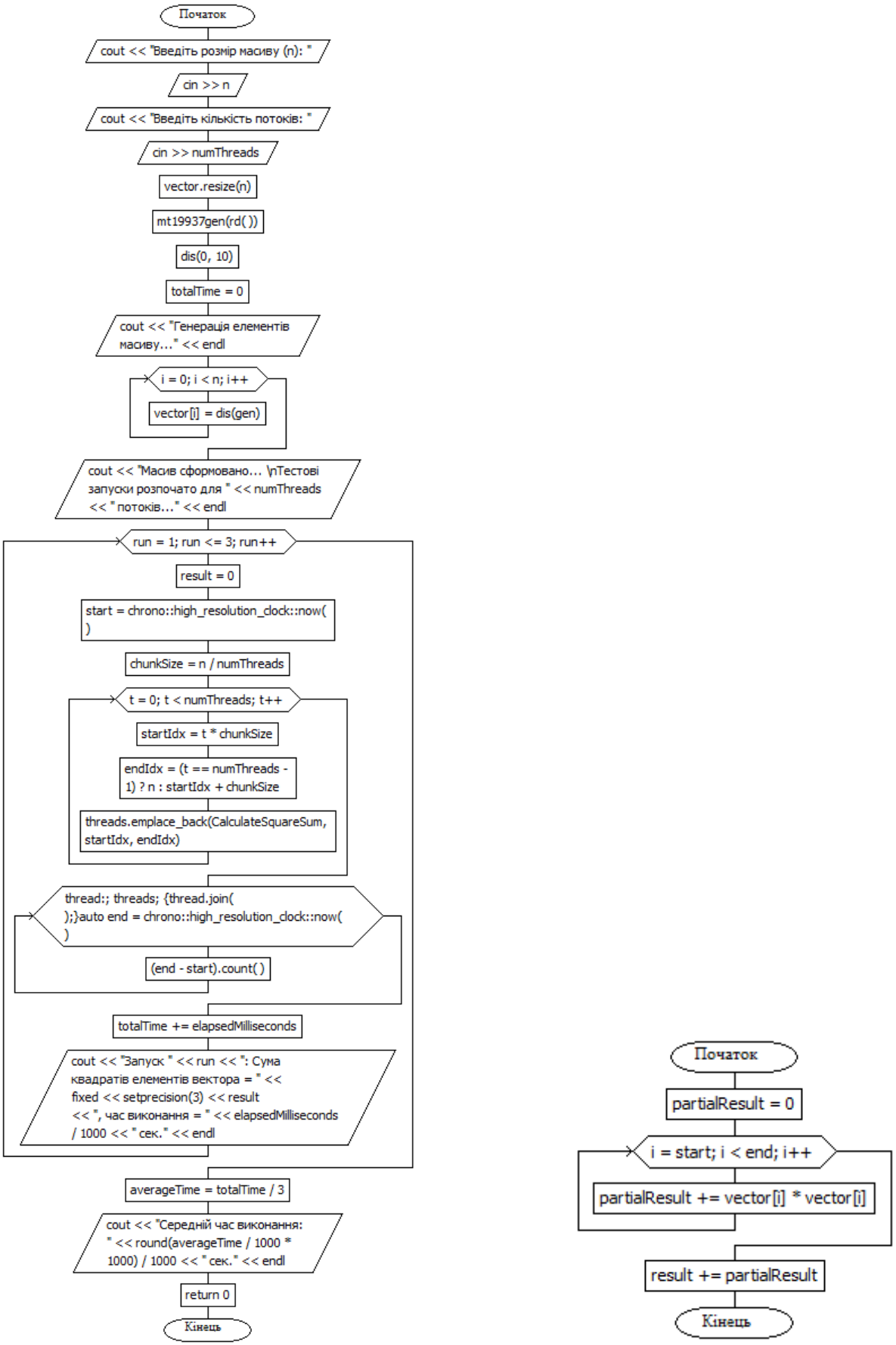

Рисунок 2.3 – Блок-схема багатопоточного алгоритму

Програмний код багатопотокової версії програми з ексклюзивним блокуванням lock() на мові C#:

```
using System.Diagnostics; // Для використання Stopwatch
using System.Text;
using System.Threading;
class Program
{
    static int n = 1800000000;                // кількість елементів вектора
    static double[] vector = new double[n]; // масив для зберігання значень вектора
    static double result = 0;                 // обчислення квадрата довжини вектора
    static int numThreads = 4;                // кількість потоків
    static object lockObj = new object();    // об'єкт для блокування потоків

    // головна точка входу у програму
    static void Main()
    {
        // генератор випадкових величин
        Random random = new Random();
        double totalTime = 0; // Загальний час виконання

        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер
        Console.WriteLine($"Генерація елементів масиву...");

        // Заповнюємо вектор значеннями (випадковими числами)
        for (int i = 0; i < n; i++)
        {
            vector[i] = random.NextDouble() * 10;
        }

        Console.WriteLine($"Масив сформовано... \nТестові запуски розпочато " +
            $"для {numThreads} потоків...");

        // Виконуємо цикл для трьох тестових запусків
        for (int run = 1; run <= 3; run++)
        {
            result = 0; // Скидаємо результат перед кожним запуском

            // Створюємо та запускаємо Stopwatch для вимірювання часу
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            // Розподіл на потоки
            Thread[] threads = new Thread[numThreads];
            int chunkSize = n / numThreads;

            for (int t = 0; t < numThreads; t++)
            {
                int start = t * chunkSize;
                int end = (t == numThreads - 1) ? n : start + chunkSize;

                threads[t] = new Thread(() => CalculateSquareSum(start, end));
                threads[t].Start();
            }
            // Очікуємо завершення роботи всіх потоків
            foreach (var thread in threads)
            {
                thread.Join();
            }
            // Зупиняємо Stopwatch
            stopwatch.Stop();

            // Додаємо час виконання до загальної суми
            totalTime += stopwatch.ElapsedMilliseconds;
```

```
            // Виводимо результат обчислень для кожного запуску
            Console.WriteLine($"Запуск {run}:Сума квадратів елементів вектора = " +
                $"{result:F3}, час виконання = {(float)stopwatch.ElapsedMilliseconds
                / 1000} сек.");
        }
        // Обчислюємо середній час виконання
        double averageTime = totalTime / 3;
        // Виводимо середній час виконання
        Console.WriteLine($"Середній час виконання: {Math.Round(averageTime /
            1000, 3)} сек.");
    }
    // Метод для обчислення суми квадратів елементів вектора
    static void CalculateSquareSum(int start, int end)
    {
        double localResult = 0;
        for (int i = start; i < end; i++)
        {
            localResult += vector[i] * vector[i];
        }
        // Блокування при доступі до спільного ресурсу
        lock (lockObj)
        {
            result += localResult;
        }
    }
}
```

```
Microsoft Visual Studio Debug Console
Генерація елементів масиву...
Масив сформовано...
Тестові запуски розпочато для 4 потоків...
Запуск 1:Сума квадратів елементів вектора = 59999143993,759, час виконання = 19,27 сек.
Запуск 2:Сума квадратів елементів вектора = 59999143993,759, час виконання = 13,772 сек.
Запуск 3:Сума квадратів елементів вектора = 59999143993,759, час виконання = 3,197 сек.
Середній час виконання: 12,08 сек.
```

Рисунок 2.4 – Результат роботи програми для 4 потоків

```
Microsoft Visual Studio Debug Console
Генерація елементів масиву...
Масив сформовано...
Тестові запуски розпочато для 8 потоків...
Запуск 1:Сума квадратів елементів вектора = 59997456582,877, час виконання = 12,632 сек.
Запуск 2:Сума квадратів елементів вектора = 59997456582,877, час виконання = 10,236 сек.
Запуск 3:Сума квадратів елементів вектора = 59997456582,877, час виконання = 7,426 сек.
Середній час виконання: 10,098 сек.
```

Рисунок 2.5 – Результат роботи програми для 8 потоків

```
Microsoft Visual Studio Debug Console
Генерація елементів масиву...
Масив сформовано...
Тестові запуски розпочато для 12 потоків...
Запуск 1:Сума квадратів елементів вектора = 60001347998,498, час виконання = 12,974 сек.
Запуск 2:Сума квадратів елементів вектора = 60001347998,498, час виконання = 9,904 сек.
Запуск 3:Сума квадратів елементів вектора = 60001347998,498, час виконання = 2,139 сек.
Середній час виконання: 8,339 сек.
```

Рисунок 2.6 – Результат роботи програми для 12 потоків

Висновки. Порівняння середнього часу виконання однопотокової версії програми (76,279 сек.) та найкращої багатопотокової версії (8,339 сек.) дозволяє зробити висновки про покращення продуктивності обчислення квадрату довжини n-вимірного вектору. Також встановлено, що збільшення кількості потоків з 4 (12,08 сек.) до 8 потоків (10,098 сек.) та до 12 потоків (8,339 сек.) впливає на зростання продуктивності обчислення, хоч й не значно. Підсумовуючі вищесказане, можна обгрунтовано стверджувати, що використання багатопотокового програмування значно прискорює обчислювальний процес навіть з використанням ексклюзивного блокування потоків.

2.3 Постановка лабораторної роботи «Організація паралельних обчислень з використанням класу Thread та блокуванням потоків Mutex»

Мета виконання роботи: отримання навичок реалізації обчислювальних алгоритмів з використанням засобів багатопотокового програмування з блокуванням потоків Mutex.

Постановка завдання лабораторної роботи.

1. Згідно індивідуальному варіанту обрати предметну область для подальшої реалізації паралельних обчислень. Реалізувати на мові C# консольну багатопотокову програму, що виконує рішення задачі предметної області з виконанням таких умов:

– початкова ініціалізація масивів, якщо це необхідно, виконується генератором випадкових величин один раз, до тестових запусків;

– розмір масивів або діапазонів пошуку повинен співпадати з реалізацією попередньої лабораторної роботи;

– передбачити блокування потоків при доступі до спільних ресурсів за допомогою Mutex;

– запуск обчислень необхідно виконувати у циклі 3 рази, з подальшим обчисленням середнього часу виконання;

– фіксувати час виконання обчислень та візуалізувати результатів обчислень у консолі;

– виконати окремі запуски програми для різної кількості потоків. Максимальна кількість потоків повинна дорівнювати кількості логічних процесорів на ПК. Порівняти результати, зробити висновки про вплив кількості потоків на продуктивність паралельних обчислень.

– виконати порівняння середнього часу виконання програм з результатами попередньої лабораторної роботи;

– зробити висновки про покращення або погіршення продуктивності.

2. У звіті навести програмний код та скріншоти роботи програми.

Приклад виконання лабораторної роботи. Написати багатопотокову версії програми обчислення квадрату довжини *n*-вимірного вектору із використанням блокування потоків Mutex.

У цій програмі використовуємо клас *Thread* для створення пулу окремих потоків *threads[]* для обчислення квадратів компонент вектора. Кількість потоків задана змінною *numThreads*, її можна змінювати за потребою. Синхронізація та блокування потоків здійснюється за допомогою *Mutex*. Вимірювання часу виконання відбувається з використанням *Stopwatch*. Результат обчислення та час виконання виводяться у консоль.

Програмний код багатопотокової версії програми з блокуванням потоків Mutex на мові C#:

```
using System.Diagnostics;
using System.Text;
using System.Threading;

class Program
{
    static int n = 1800000000; // Розмір вектора
    static int numThreads = 4; // Кількість потоків
    static double[] vector; // Вектор для обчислень
    static double result = 0; // Результат обчислень
    // М'ютекс для ексклюзивного доступу до результату
    static Mutex mutex = new Mutex();

    static void Main(string[] args)
    {
        double totalTime = 0; // Загальний час виконання
        vector = new double[n];
        Random random = new Random();
        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер
```

```
        Console.WriteLine($"Кількість потоків: {numThreads}");
        Console.WriteLine("Генерація елементів масиву...");
        // Заповнюємо вектор випадковими значеннями
        for (int i = 0; i < n; i++)
        {
            vector[i] = random.NextDouble();
        }

        Console.WriteLine($"Масив сформовано... \nТестові запуски розпочато...");
        Stopwatch stopwatch = new Stopwatch();

        for (int i = 0; i < 3; i++)
        {
            result = 0;
            stopwatch.Restart();

            Thread[] threads = new Thread[numThreads];

            // Запускаємо потоки для обчислення суми квадратів елементів вектора
            for (int j = 0; j < numThreads; j++)
            {
                threads[j] = new Thread(CalculateSquareLength);
                threads[j].Start(j);
            }

            // Очікуємо завершення всіх потоків
            foreach (var thread in threads)
            {
                thread.Join();
            }

            stopwatch.Stop();

            Console.WriteLine($"Тест {i + 1}: Квадрат довжини вектора: {result}, " +
                $"час виконання:{(float)stopwatch.ElapsedMilliseconds / 1000} сек");
            // Додаємо час виконання до загальної суми
            totalTime += stopwatch.ElapsedMilliseconds;
        }
        // Обчислюємо середній час виконання
        double averageTime = totalTime / 3;
        // Виводимо середній час виконання
        Console.WriteLine($"Середній час виконання: " +
            $"{Math.Round(averageTime / 1000, 3)} сек.");
    }

    static void CalculateSquareLength(object threadIndex)
    {
        int startIndex = (int)threadIndex * (n / numThreads);
        int endIndex = ((int)threadIndex + 1) * (n / numThreads);

        double threadResult = 0;

        // Обчислення суми квадратів частини вектора
        for (int i = startIndex; i < endIndex; i++)
        {
            threadResult += vector[i] * vector[i];
        }

        // Заблоковуємо м'ютекс, щоб оновити загальний результат
        mutex.WaitOne();
        result += threadResult;
        mutex.ReleaseMutex();
    }
}
```

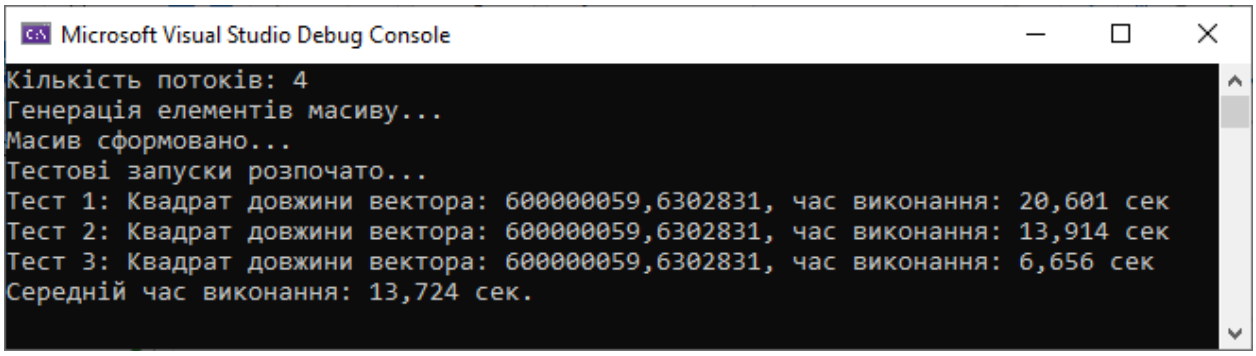

Рисунок 2.7 – Результат роботи програми для 4 потоків

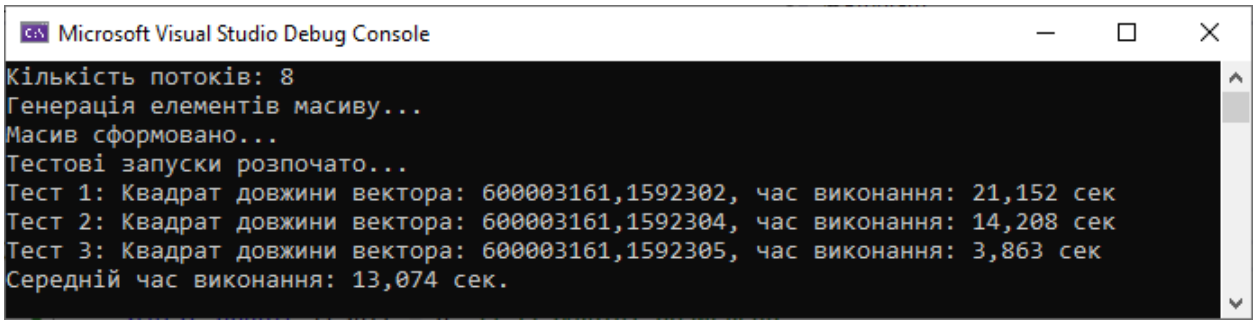

Рисунок 2.8 – Результат роботи програми для 8 потоків

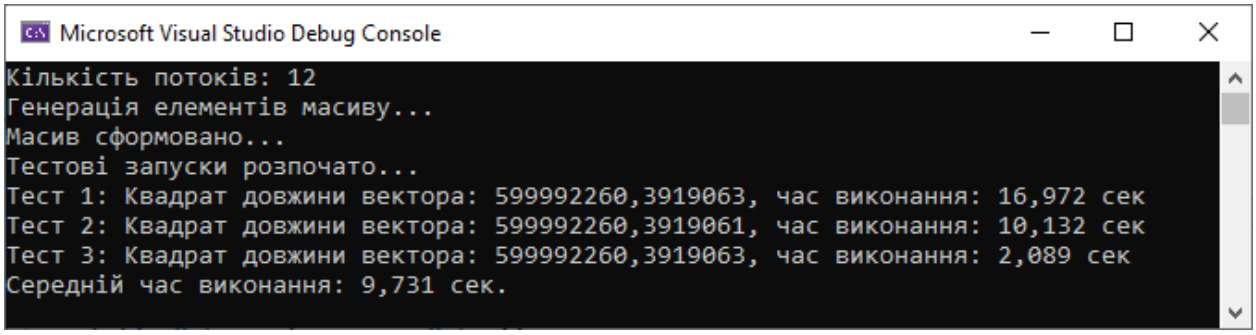

Рисунок 2.9 – Результат роботи програми для 12 потоків

Висновки. Порівняння середнього часу виконання версій програм з блокуванням lock та з блокуванням Mutex для різної кількості потоків дозволяє зробити висновки про те, що блокування Mutex більш гальмує обчислювальний процес у порівнянні з lock. Встановлено, що збільшення кількості потоків з 4 (13,724 сек.) до 8 потоків (13,074 сек.) та до 12 потоків (9,731 сек.) впливає на зростання продуктивності обчислення, хоч й не значно.

2.4 Постановка лабораторної роботи «Організація паралельних обчислень з використанням класу Thread та блокуванням потоків Semaphore»

Мета виконання роботи: отримання навичок реалізації обчислювальних алгоритмів з використанням засобів багатопотокового програмування з блокуванням потоків Semaphore.

Постановка завдання лабораторної роботи.

1. Згідно індивідуальному варіанту обрати предметну область для подальшої реалізації паралельних обчислень. Реалізувати на мові C# консольну багатопотокову програму, що виконує рішення задачі предметної області з виконанням таких умов:

– початкова ініціалізація масивів, якщо це необхідно, виконується генератором випадкових величин один раз, до тестових запусків;

– розмір масивів або діапазонів пошуку повинен співпадати з реалізацією попередньої лабораторної роботи;

– передбачити блокування потоків при доступі до спільних ресурсів за допомогою Semaphore;

– запуск обчислень необхідно виконувати у циклі 3 рази, зафіксувати час виконання тестових обчислень та середнього часу виконання;

– виконати окремі запуски програми для різної кількості потоків. Максимальна кількість потоків повинна дорівнювати кількості логічних процесорів на ПК. Порівняти результати, зробити висновки про вплив кількості потоків на продуктивність паралельних обчислень.

– виконати порівняння середнього часу виконання програм з різними типами блокування потоків (lock, Mutex та Semaphore) в табличному та графічному вигляді;

– зробити висновки про покращення або погіршення продуктивності.

2. У звіті навести програмний код та скріншоти роботи програми.

Приклад виконання лабораторної роботи. Написати багатопотокову

версії програми обчислення квадрату довжини *n*-вимірного вектору із використанням блокування потоків Semaphore.

У цій програмі використовуємо клас *Thread* для створення пулу окремих потоків *threads[]* для обчислення квадратів компонент вектора. Кількість потоків задана змінною *numThreads*, її можна змінювати за потребою. Синхронізація потоків відбувається за допомогою семафорів (Semaphore). Семафор ініціалізується значенням 1, що дозволяє лише одному потоку одночасно отримувати доступ до обчислень. Вимірювання часу виконання відбувається за допомогою Stopwatch. Результат обчислення та час виконання виводяться у консоль.

Програмний код багатопотокової версії програми з блокуванням потоків Semaphore на мові C#:

```
using System.Diagnostics;
using System.Text;
using System.Threading;

class Program
{
    static int n = 1800000000;  // Розмір вектора
    static double[] vector;     // Масив для обчислень
    static double result = 0;   // Результат обчислень
    // Семафор для ексклюзивного доступу до результату
    static Semaphore semaphore = new Semaphore(1, 1);

    // кількість потоків = кількість процесорів
    static int numThreads = Environment.ProcessorCount;
    // або вказуємо необхідну кількість потоків
    // static int numThreads = 4;

    static void Main(string[] args)
    {
        double totalTime = 0; // Загальний час виконання
        vector = new double[n];
        Random random = new Random();
        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер
        Console.WriteLine("Всього логічних процесорів на ПК: {0}.",
Environment.ProcessorCount);
        Console.WriteLine("Кількість потоків: {0}.", numThreads);
        Console.WriteLine("Генерація елементів масиву...");
        // Заповнюємо вектор випадковими значеннями
        for (int i = 0; i < n; i++)
        {
            vector[i] = random.NextDouble();
        }

        Console.WriteLine($"Масив сформовано... \nТестові запуски розпочато...");
        Stopwatch stopwatch = new Stopwatch();

        for (int i = 0; i < 3; i++)
        {
            result = 0;
```

```
            stopwatch.Restart();
            Thread[] threads = new Thread[numThreads];

            // Запускаємо потоки для обчислення суми квадратів елементів вектора
            for (int j = 0; j < numThreads; j++)
            {
                threads[j] = new Thread(CalculateSquareLength);
                threads[j].Start(j);
            }

            // Очікуємо завершення всіх потоків
            foreach (var thread in threads)
            {
                thread.Join();
            }

            stopwatch.Stop();

            Console.WriteLine($"Тест {i+1}: Квадрат довжини вектора:{result}, "+
                $"час виконання:{(float)stopwatch.ElapsedMilliseconds / 1000} сек");
            // Додаємо час виконання до загальної суми
            totalTime += stopwatch.ElapsedMilliseconds;
        }
        // Обчислюємо середній час виконання
        double averageTime = totalTime / 3;
        // Виводимо середній час виконання
        Console.WriteLine($"Середній час виконання: " +
            $"{Math.Round(averageTime / 1000, 3)} сек.");
    }

    static void CalculateSquareLength(object threadIndex)
    {
        int startIndex = (int)threadIndex * (n / numThreads);
        int endIndex = ((int)threadIndex + 1) * (n / numThreads);

        double threadResult = 0;

        // Обчислення суми квадратів частини вектора
        for (int i = startIndex; i < endIndex; i++)
        {
            threadResult += vector[i] * vector[i];
        }

        // Заблоковуємо семафор, щоб оновити загальний результат
        semaphore.WaitOne();
        result += threadResult;
        semaphore.Release();
    }
}
```

```
Microsoft Visual Studio Debug Console
Всього логічних процесорів на ПК: 12.
Кількість потоків: 4.
Генерація елементів масиву...
Масив сформовано...
Тестові запуски розпочато...
Тест 1: Квадрат довжини вектора:600000963,9857142, час виконання:14,701 сек
Тест 2: Квадрат довжини вектора:600000963,9857142, час виконання:9,747 сек
Тест 3: Квадрат довжини вектора:600000963,9857142, час виконання:5,693 сек
Середній час виконання: 10,047 сек.
```

Рисунок 2.10 – Результат роботи програми для 4 потоків

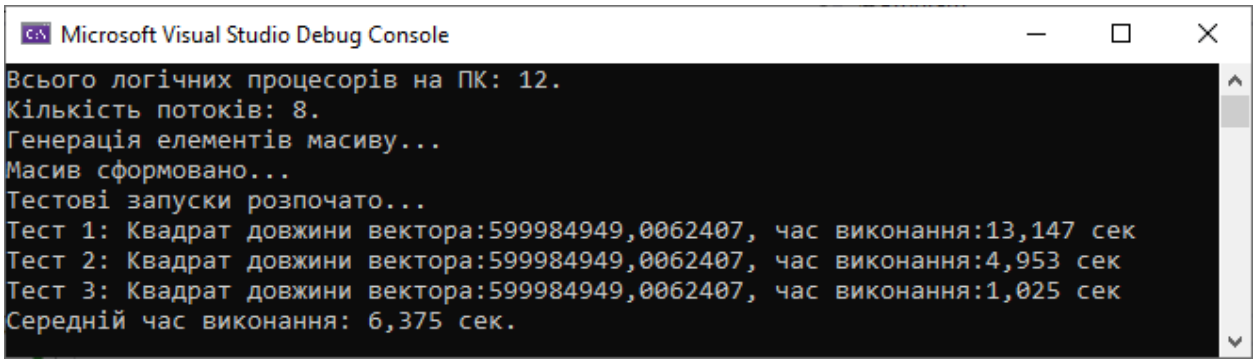

Рисунок 2.11 – Результат роботи програми для 8 потоків

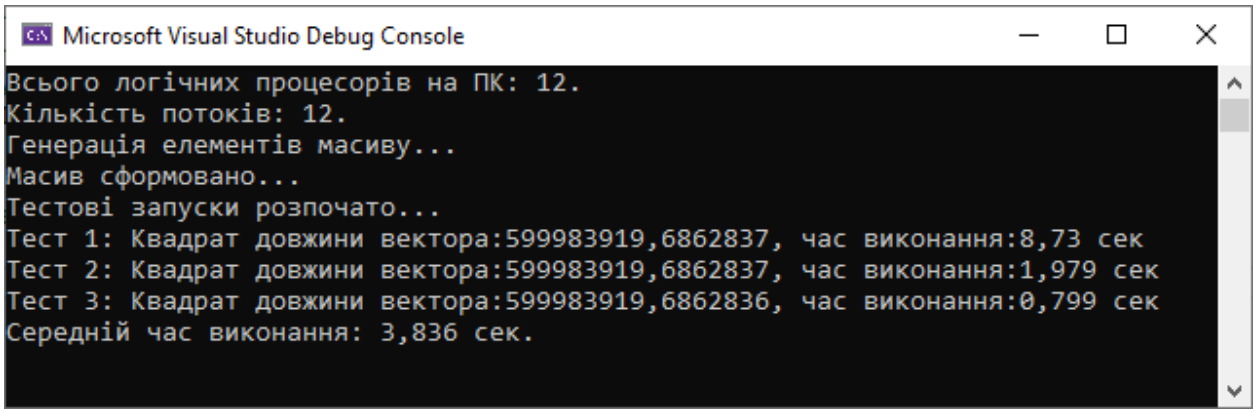

Рисунок 2.12 – Результат роботи програми для 12 потоків

Таблиця 2.1

Порівняння середнього часу виконання програм з різними типами блокування потоків

Тип блокування	Середній час виконання трьох тестових запусків		
	4 потоки	8 потоків	12 потоків
Lock	12,08	10,098	8,339
Mutex	13,724	13,074	9,731
Semaphore	10,047	6,375	3,836
Середнє значення	**11,950**	**9,849**	**7,302**

Висновки. Порівняння середнього часу виконання трьох версій програм з блокуванням потоків lock, Mutex та Semaphor для різної кількості потоків (табл. 2.1) дозволяє зробити висновки про те, що блокування Semaphor дозволяє найменш гальмує обчислювальний процес. Також встановлено (рис.

2.13), що збільшення кількості потоків з 4 до 8 потоків та до 12 потоків впливає на зростання продуктивності обчислення.

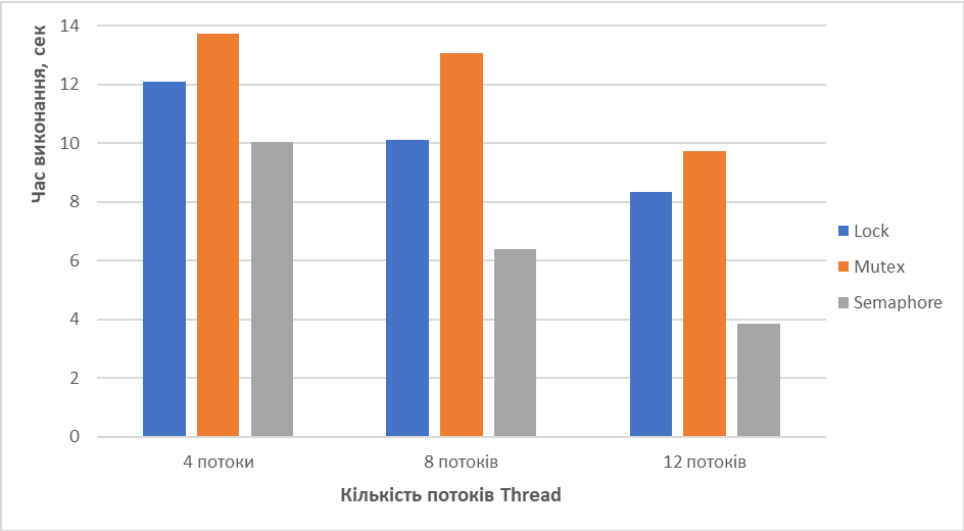

Рисунок 2.13 – Візуалізація середнього часу виконання програм

Висновки до розділу

У розділі виконано аналіз характеристик об'єктів галузі організації паралельних обчислень з використанням багатопотокового програмування мовою C#. Виконано аналіз освітніх компонент вітчизняних закладів вищої освіти, присвячених вивченню процесів організації паралельних обчислень. Аналіз запропонованих підходів до паралельних обчислень показав, що необхідно враховувати конкретні потреби розробника прикладного проекту, доступних ресурсів і його знань у галузі паралельного програмування. За результатами виконаного дослідження виконана постановка завдань нового лабораторного практикуму «Паралельні обчислення мовою C#», який акцентує увагу здобувачів на організація паралельних обчислень з використанням класу Thread та організації синхронізації потоків засобами lock, Mutex та Semaphore:

1. організація паралельних обчислень з використанням класу Thread та блокуванням потоків lock;

2. організація паралельних обчислень з використанням класу Thread та

блокуванням потоків Mutex;

3. організація паралельних обчислень з використанням класу Thread та блокуванням потоків Semaphore.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, зокрема опис предметної області, блок-схеми однопотокових та багатопотокових алгоритмів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio.

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

Для підготовки фахівців цифрових технологій, які викладатимуть освітній модуль «Паралельні обчислення мовою C#» у закладах вищої освіти, важливо мати комплексний підхід.

З кожним роком паралельні обчислення та хмарні технології стають невід'ємною частиною нашого повсякденного життя. Їх вплив поширюється на різні сфери діяльності, починаючи від бізнесу і закінчуючи розвагами, освітніми програмами та медициною. Використання хмарних рішень дозволяє не тільки зберігати великі обсяги даних, але й здійснювати їхню обробку в реальному часі, забезпечуючи доступність інформації незалежно від місця розташування користувача. Хмара стала тим фундаментом, на якому будуються сучасні IT-рішення, що сприяє зростанню попиту на фахівців у цій сфері.

Розробка програмного забезпечення є однією з найбільш затребуваних і перспективних професій сучасності. Розробники займаються створенням додатків і систем, які функціонують у середовищі хмарних обчислень, забезпечуючи надійну роботу різноманітних сервісів. Вони впроваджують нові функції та виправляють помилки, що дозволяє підтримувати стабільність і продуктивність програмних продуктів. Окрім цього, розробники повинні мати високий рівень знань у мовах програмування, таких як Python, Java, C++ та інших, а також бути в курсі останніх тенденцій в API-інтерфейсах та розробці програмних інструментів.

Системні адміністратори займають ключову позицію у сфері хмарних технологій, оскільки саме вони відповідають за безперебійну роботу серверів та всієї інфраструктури підприємства. Їхня робота включає встановлення, налаштування та підтримку хмарних платформ, а також моніторинг їхньої продуктивності. Системні адміністратори також повинні бути експертами у сфері віртуалізації, розуміти основи мережевих протоколів та забезпечення

безпеки даних. Їхні знання дозволяють швидко реагувати на можливі проблеми, підтримуючи стабільність та продуктивність систем.

Архітектори хмарних рішень розробляють і впроваджують складні інфраструктури, здатні витримувати значні навантаження та забезпечувати високу надійність. Вони визначають стратегію розвитку хмарних систем, створюють плани з масштабування ресурсів та інтеграції нових функцій. Основним завданням архітектора є забезпечення максимальної ефективності використання хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure, Google Cloud та інших. Важливим аспектом роботи архітектора є також захист даних та забезпечення відповідності хмарної інфраструктури сучасним стандартам безпеки.

Менеджери у сфері хмарних технологій займаються організацією процесів впровадження нових рішень, плануванням ресурсів та координацією роботи команд розробників. Вони розробляють стратегії розвитку компаній, приймаючи рішення щодо вибору найоптимальніших технологій для задоволення потреб бізнесу. Менеджери відповідають за контроль якості систем, що використовуються, та оптимізацію витрат на хмарні сервіси. Їхні обов'язки також включають аналіз ефективності впроваджених рішень та постійну взаємодію з іншими фахівцями.

Інженери, які спеціалізуються на хмарних технологіях, займаються налаштуванням і підтримкою серверів, мережевих систем та безпеки даних. Вони володіють знаннями про основні платформи, що функціонують у цій сфері, та використовують ці навички для забезпечення стабільної роботи інфраструктури. З розвитком нових технологій інженери постійно вдосконалюють свої знання та впроваджують новітні рішення у хмарних середовищах.

Зі збільшенням обсягів даних аналітики стають невід'ємною частиною процесу прийняття рішень в компаніях. Вони використовують хмарні платформи для обробки великих масивів інформації, будуючи прогностичні моделі та алгоритми машинного навчання. Аналітики мають глибокі знання в

галузі статистики, програмування та аналізу даних, що дозволяє їм знаходити важливі закономірності та витягувати з них корисну інформацію для бізнесу.

У зв'язку зі збільшенням кіберзагроз фахівці з кібербезпеки відіграють ключову роль у захисті даних, що зберігаються в хмарі. Вони розробляють стратегії захисту та реагування на можливі атаки, контролюють вразливості систем та запобігають витоку інформації. Їхня робота вимагає постійного оновлення знань, оскільки кіберзлочинці постійно вдосконалюють свої методи.

IT-сектор стикається з численними викликами, включаючи дефіцит кваліфікованих кадрів. З розвитком нових технологій зростає потреба в професіоналах рівня Middle та Senior, які мають досвід роботи з інноваційними рішеннями та інструментами. Велика конкуренція за таланти на ринку праці змушує компанії підвищувати заробітну плату та розширювати можливості для навчання співробітників.

З впровадженням нейромереж та штучного інтелекту (ШІ) ринок праці зазнає значних змін. Автоматизація рутинних процесів дозволить звільнити час для більш креативних завдань. ШІ не лише спростить виконання багатьох операцій, але й створить нові можливості для розвитку, зокрема відкриття нових професій, таких як інженер-промт або тренер ШІ.

З розвитком технологій важливим аспектом стає оновлення освітніх програм та адаптація їх до сучасних потреб ринку. Освітні установи спільно з бізнесом повинні розробляти курси та програми, які дозволять студентам отримати актуальні знання та навички. Вища освіта стає все більш орієнтованою на підготовку фахівців, здатних працювати з передовими технологіями та захищати інформаційні системи від нових загроз.

Попри автоматизацію багатьох процесів, ШІ та нейромережі не зможуть повністю замінити людську працю, особливо у творчих сферах. Люди, які здатні генерувати унікальні ідеї та підходи, завжди будуть затребувані. Співпраця людини та ШІ відкриває безліч можливостей для розвитку нових продуктів та сервісів, що забезпечить високий рівень конкуренції на ринку

праці.

Оскільки кіберзлочинці стають усе більш винахідливими, необхідність у спеціалістах із захисту даних продовжує зростати. Розробка нових методів захисту та постійний моніторинг систем безпеки стали важливою частиною сучасного IT-сектору. Бізнес інвестує все більше ресурсів у створення надійних кіберзахисних рішень, що дозволяє зменшити ризики та підвищити стабільність роботи компаній.

Висновки до розділу

У цьому розділі проведено детальний аналіз вимог до кадрового забезпечення об'єкта IT-сфери в умовах цифрової трансформації суспільства. Встановлено, що паралельні обчислення та хмарні технології стали незамінною частиною сучасного життя, забезпечуючи можливості для зберігання, обробки й обміну великими обсягами даних, а також дозволяючи організовувати віддалену роботу та глобальне спілкування.

Виявлено, що в останні роки спостерігається зростання попиту на IT-професії, такі як розробники, системні адміністратори, архітектори, менеджери хмарних технологій, інженери хмарних технологій, аналітики даних та фахівці з кібербезпеки. Встановлено, що є необхідність модернізації освітніх програм, оскільки швидкий розвиток технологій випереджає можливості існуючих навчальних планів. Для вирішення цього завдання потрібно посилити співпрацю між навчальними закладами та бізнесом.

Особливу увагу приділяють питанням кібербезпеки, у зв’язку з чим компанії покращують свої заходи для захисту від кіберзагроз, що веде до збільшення інвестицій у цю сферу.

Прогнозується, що протягом найближчих 10 років використання нейронних мереж та штучного інтелекту значно зросте, що може вплинути на ринок праці, зменшуючи попит на певні професії та створюючи нові можливості.

4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ МОВОЮ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ. ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «СИНХРОНІЗАЦІЯ ТА БЛОКУВАННЯ ПОТОКІВ» ДИСЦИПЛІНИ «ПРОГРАМНА ІНЖЕНЕРІЯ» ДЛЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ СПЕЦІАЛЬНОСТІ 015 ПРОФЕСІЙНА ОСВІТА (ЦИФРОВІ ТЕХНОЛОГІЇ)

4.1 Вихідні дані

Навчальний заклад: Бахмутський навчально-науковий професійно-педагогічний інститут Харківського національного університету імені В.Н. Каразіна;

галузь знань: 01 Освіта / Педагогіка;

спеціальність: 015 Професійна освіта (Цифрові технології);

рівень вищої освіти: перший (бакалаврський);

освітній ступінь: бакалавр;

дисципліна: «Програмна інженерія»;

тема: «Синхронізація та блокування потоків».

Дисципліна містить такі характеристики, як:

кількість кредитів – 4.

загальну кількість годин для вивчення дисципліни – 120 навчальних годин з яких 70 годин самостійної роботи та 50 годин аудиторної роботи (20 години лекційних занять та 30 годин лабораторної роботи) для денної форми навчання;

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи становить: для денної форми навчання – 4/120.

Дисципліна «Програмна інженерія» викладається у 5-му семестрі 3-го року професійної підготовки бакалаврів для денної форми навчання.

Форма контролю: Екзамен.

Великий обсяг самостійної роботи, складність навчального матеріалу обумовлюють актуальність проведення консультативних занять з курсу «Програмна інженерія».

Консультативні заняття поглиблюють знання, пояснюють та актуалізують окремі найбільш складні частини навчального матеріалу з теми «Синхронізація та блокування потоків».

4.2 Проектування цілей консультативного заняття

Визначення навчальних цілей заняття є одним з головних питань, від вирішення якого залежить методична організація заняття, вибір його форм, методів, засобів навчання та контролю. В цьому сенсі навчальні цілі є системоутворюючим фактором, бо, відображуючи кінцеві необхідні результати досягнень здобувачів вищої освіти, чітко детермінують принципи побудови системи навчання по всіх основних її параметрах.

Тому методична підготовка здобувачів вищої освіти передбачає перш за все оволодіння вміннями диференційовано визначати навчальні цілі, відповідно до певних рівнів професійної підготовки або рівнів засвоєння.

Проектування цілей консультативного заняття представлені у табл. 4.1 [49].

Таким чином, нами були розроблені цілі консультативного заняття з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

Таблиця 4.1

Цілі консультативного заняття

Цілі консультативного заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій здобувачів освіти
1	2	3	4
1	З переліку визначень впізнавати основні поняття теми «Синхронізація та блокування потоків» такі, як Sleep, Join, вміти називати нелінійні керуючі конструкції мови С та два основні ексклюзивні блокування.	Знати визначення понять «Синхронізація» та «Поток», знання сутності поняття «Паралельні обчислення мовою С#»	Правильно названі з переліку основні поняття теми «Синхронізація та блокування потоків» такі, як Sleep, Join, вміти називати нелінійні керуючі конструкції мови С та два основні ексклюзивні блокування.
2	Уміти характеризувати основні елементи, які включає паралельні обчислення мовою С#, уміти розділити конструкції синхронізації.	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Синхронізація та блокування потоків» такі, як Sleep, Join, вміти називати нелінійні керуючі конструкції мови С та два основні ексклюзивні блокування.	Правильно охарактеризовано властивість ThreadState та управління синхронізацією потоків.
3	Уміти аналізувати нескінченний цикл та робити висновки про доцільність використання в кодах програми операторів переходу return, goto, continue.	Виконання дій першого і другого рівнів: правильно охарактеризований циклічний оператор for, а також умови входу та виходу з циклу, застосування оператора break для виходу в зовнішній та внутрішній цикл, про класифіковані інвестиції оператори за призначенням	Правильно проаналізований нескінченний цикл та зроблені висновки, щодо доцільність використання в кодах програми операторів переходу return, goto, continue.
4	Уміти розробляти програму з використанням паралельних обчислень мовою С#.	Виконання дій першого, другого і третього рівнів: правильно проаналізований нескінченний цикл та зроблені висновки, щодо доцільність використання в кодах програми операторів переходу return, goto, con	Правильно розроблена програма з використанням паралельних обчислень мовою С#.

4.3 Перелік джерел інформації

Майбутній фахівець має вміти самостійно користуватися джерелами інформації.

Наведемо перелік джерел інформації для підготовки здобувачів вищої освіти до консультації згідно з робочою програмою дисципліни «Програмна інженерія».

Нижче представлено перелік основної та допоміжної літератури, а також інформаційні ресурси для вивчення дисципліни та підготовки до консультативного заняття.

Рекомендована література:

Основна (базова) література

1. Гордєнко Ю., Таран В. Хмарні обчислення: Конспект лекцій. Електронне мережне навчальне видання. КПІ ім. Ігоря Сікорського, 2022. – С. 1024.

2. Гордєнко Ю., Таран В. Хмарні обчислення: лабораторний практикум. Електронне мережне навчальне видання. КПІ ім. Ігоря Сікорського, 2021. – С. 38.

3. Гришанович Т. О. Лабораторний практикум із дисципліни “Паралельні та розподілені обчислення” [Електронний ресурс]; ВНУ імені Лесі Українки. Луцьк : ВНУ імені Лесі Українки, 2022. – С. 50.

4. Зінченко О.В., Іщеряков С.М., Прокопов С.В., Сєрих С.О., Василенко В.В. Хмарні технології. – Навчальний посібник. – К: ФОП Гуляєва В.М., 2020. – С. Минайленко Р. М. Паралельні та розподілені обчислення : навч. посіб. М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2021. – 153 с.

5. Коцовський В.М. Теорія паралельних обчислень. – Ужгород: ПП «АУТДОР–Шарк», 2021. – 188 с.

6. Методичні вказівки до лабораторних робіт з дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 123 «Комп’ютерна

інженерія» всіх форм навчання. Частина 2 / Укл.: Р.К. Кудерметов, В.В. Шкарупило, О.В. Польська. – Запоріжжя: НУ «Запорізька політехніка», 2020.

7. Методичні вказівки до лабораторних робіт з дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 123 «Комп'ютерна інженерія» всіх форм навчання. Частина 2 / уклад.: Р.К. Кудерметов, В.В. Шкарупило, О.В. Польська. – Запоріжжя : НУ «Запорізька політехніка», 2020.

8. Паралельні та розподілені обчислення: Методичні вказівки до виконання самостійних робіт для студентів денної форми навчання за спеціальністю 123 «Комп'ютерна інженерія» / уклад. : В.В. Смірнов, Н.В. Смірнова ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 23 с.

Додаткова (допоміжна) література

1. Chykunov P. Organization of prime number search process using Nvidia CUDA / P. Chykunov, V. Chernetskyi // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 06-07 грудня 2023 р.). – Харків: ННППІ УІПА, 2023. С. 40-41.

2. Chykunov P. Profiling multithreaded programs in the Visualizer Concurrency / P. Chykunov, A. Kotov // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 06-07 грудня 2023 р.). – Харків: ННППІ УІПА, 2023. С. 36-38.

3. Бородкіна І. Інженерія програмного забезпечення: навчальний посібник. Центр учбової літератури, 2021. – 204 с.

4. Кипаренко Д. О. Методи обробки великих даних в розподілених хмарних системах : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 123 Комп'ютерна інженерія / Д. О. Кипаренко ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2024. – 66 с.

5. Климова І. М. Моделі та методи побудови паралельних алгоритмів / І. М. Климова, А. В. Черниш, В. М. Федорченко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доп. 14-ї міжнар. наук.-техн. конф., 25-26 квітня 2024 р. – Харків : Impress., 2024. – С. 129.

6. Сіциліцин Ю. О., Є. С. Пересунько. Візуалізація паралельних обчислень. Українські студії в європейському контексті: зб. наук. пр. № 8. 2024, с. 276.

7. Методичні вказівки до практичних занять і розрахунково-графічної роботи з дисциплін «Паралельні та розподілені обчислення та CKLOUD-технології» / укладачі : М. А. Мірошник, Ю. В. Савін. - Харків, 2023. - 93 с.

8. Розподілені та паралельні обчислення. Лабораторний практикум для студентів спеціальності 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня [Електронний ресурс] / уклад. С. В. Мінухін, Ю. В. Савін; Харківський національний економічний університет ім. С. Кузнеця. — Харків : ХНЕУ ім. С. Кузнеця, 2023. — 170 с.: іл. — Загол. з титул. екрану. — Бібліогр.: с. 169.

9. Чикунов П. О. Можливості мови C# для організації паралельних обчислень / Матеріали Міжнародної науково-практичної конференції «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. – С. 140-143.

10. Чикунов П.О. Розробка програмного середовища візуалізації та аналізу графових моделей / П.О. Чикунов, М.О. Сафонов, Д.А. Неізмайлов // Матеріали III Всеукраїнської науково-практичної інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Бахмут, 29-30 листопада, 2020 р.). – Бахмут: ННППІ УІПА, 2020. – С. 48-49.

Інформаційні ресурси

1. http://cyber.onua.edu.ua/ – робочі матеріали з курсу

2. http://dspace.onua.edu.ua – eNUOLAIR – депозитарій (архів) НУ «ОЮА»

3. Microsoft Azure: From Zero to Hero – The Complete Guide – Онлайн-курс Udemy. URL: https://www.udemy.com/course/parallel-dotnet/

4. Multithreading and Parallel Programming in C# – Онлайн-курс Udemy. URL: https://www.udemy.com/course/parallel-csharp/

5. Parallel programming in .NET: A guide to the documentation – Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/

6. Amazon Web Services: AWS Cloud Practitioner Essentials – Онлайн-курс edX. URL: https://www.edx.org/learn/amazon-web-services-aws-cloud-practitioner

7. Introduction to High-Performance and Parallel Computing – Онлайн-курс Coursera. URL: https://www.coursera.org/learn/scala-parallel-programming

8. Learn Parallel Programming with C# and .NET – Онлайн-курс Udemy. URL: https://www.udemy.com/course/parallel-dotnet/

Основним джерелом для підготовки здобувачів вищої освіти до консультації є конспект лекцій, розроблений викладачем, оскільки він є найбільш адаптованим до робочої програми дисципліни «Програмна інженерія».

4.4 Визначення найбільш складних для розуміння та засвоєння питань

На даному етапі визначимо найбільш складні для розуміння та засвоєння питання (табл. 4.2) [49].

Отже, на даному етапі ми визначили найбільш складні для розуміння та засвоєння питання з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

Таблиця 4.2

Обрання питань для консультування та формулювання відповідей на можливі питання

Теми (або тема) дисципліни	Зміст програми за кожною темою	Найбільш складні питання за темами (темою)	Відповіді на питання
1	2	3	4
«Синхронізація та блокування потоків».	1. Синхронізація потоків. 2. Блокування об’єкту. 3. Блокування класу. 4. Блокування потоків.	1. Назвіть та охарактеризуйте, будь ласка, на які категорії можна розділити конструкції синхронізації?	1. Конструкції синхронізації можна розділити на чотири категорії: 1. Прості методи блокування. Вони чекають на закінчення іншого потоку або протягом певного періоду часу. Методи Sleep, Join є найпростішими методами блокування. 2. Блокуючі конструкції. обмежують кількість потоків, які можуть виконувати активність або виконувати секцію коду за один раз. Ексклюзивні блокуючі конструкції найбільш поширені - вони дозволяють лише один потік за раз і дозволяють конкуруючим потокам отримувати доступ до загальних даних, не заважаючи один одному. Стандартними ексклюзивними замикаючими конструкціями є lock, Mutex, SpinLock. Неексклюзивні блокуючі конструкції – це Semaphore, SemaphoreSlim. 3. Сигналізаційні конструкції. Вони дозволяють потоку зупинятися до отримання повідомлення від іншого, уникаючи необхідності неефективного опитування. Найчастіше використовувані сигнальні пристрої- це дескриптори очікування подій і клас Monitor. 4. Неблокуючі конструкції синхронізації. Вони захищають доступ до загального поля, викликаючи примітив процесора. У CLR і C# надають такі неблокуючі конструкції: Thread.MemoryBarrier, Thread.VolatileRead, Thread.VolatileWrite, за volatile ключовим словом, і клас Interlocked

Продовження табл. 4.2

1	2	3	4
		2. Що таке синхронізація? 3. Скажіть, будь ласка, як відбувається розблокування	2. Синхронізація – це координація процесів потоків для отримання прогнозованого результату. 3. Розблокування відбувається одним із чотирьох способів: 1. За умовою блокування 2. За тайм-аутом операції 3. Шляхом Thread.Interrupt 4. Шляхом Thread.Abort

4.5 Вибір дидактичних методів активізації

Здійснюємо вибір методів активізації навчальної діяльності здобувачів вищої освіти на консультації для спонукання до активної розумової та практичної діяльності в процесі оволодіння навчальним матеріалом (табл. 4.3) [50].

Таблиця 4.3

Методи активізації навчальної діяльності студентів на консультації

Дидактичні методи	Реалізація методів при проведенні консультаційного заняття
Методи підвищення наочності	Використання інтерактивної дошки для демонстрації слайдів з теми «Синхронізація та блокування потоків» та мультимедійного проектора.
Мотиваційні методи	Для реалізації мотивації використаємо: тип: внутрішня мотивація; вид: вступна мотивація; метод: мотивуючий вступ; прийом: віднесення до особистості. Формування позитивного відношення до вивчення даної теми: «Тема «Синхронізація та блокування потоків» – одна з найбільш важливих тем курсу – вміти розробляти та аналізувати алгоритми розв’язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій».
Проблемні методи	Використання проблемного питання. Проблемне питання: «Навіщо синхронізувати потоки?».
Комунікативні методи	Імітація ситуацій з реального життя. Вміти розробляти та аналізувати алгоритми розв’язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій.

Таким чином, ми обрали методи активізації навчальної діяльності здобувачів вищої освіти на консультації.

4.6 Вибір способів організації консультативного заняття

Здійснюємо вибір способів організації консультативного заняття, згідно даних, наведених в таблиці 4.4 [49].

Таблиця 4.4

Варіанти організації консультативного заняття

№ варіанта	Етапи організації заняття	Характеристика варіанта
1	- вступне слово лектора, - відповіді на питання студентів і обговорення їх, - заключне слово викладача	Недоліком цього варіанту проведення лекції-консультації є відсутність послідовності, системи в питаннях, на які доводиться викладачу давати відповіді. Питання поступають хаотично, що знижує якість консультації.
2	- збір питань в письмовій формі до лекції, їх систематизація, - відповіді на питання, що поступили, - відповіді на додаткові питання, - обмін думками, - висновки	Цей варіант, на відміну від попереднього, дозволяє викладачу групувати відповіді, що сприяє кращому засвоєнню навчального матеріалу здобувачами освіти.
3	- видача завдань на самостійне вивчення матеріалу теми. - підготовка питань лектору. - відповіді і їх обговорення	В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу.
4	- повідомлення теми, - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки	Цей варіант лекції-консультації проводиться, як правило, зі спеціальних дисциплін, іноді для цієї мети використовуються наукові семінари. Такі заняття дають можливість зіставити думки різних учених на одну і ту ж проблему і є чудовою школою ведення дискусії.

Консультативне заняття будемо здійснювати згідно першого варіанту організації, на ньому викладач пояснює питання, які здалися складними здобувачами освіти.

4.7 Розробка сценарію проведення консультативного заняття

Проведення консультативного заняття пропонуємо здійснити згідно обраного варіанту його організації (табл. 4.5) [50].

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консультативного заняття	Дії викладача	Дії здобувачів освіти
1	2	3
Організаційний момент	Вітання, фіксація відсутніх, перевірка зовнішньої обстановки в аудиторії	Вітання викладача. Перевірка присутності, формування позитивної мотивації на здійснення навчальної діяльності.
Повідомлення теми і мети заняття	Повідомлення теми заняття: «Синхронізація та блокування потоків». Формулювання його цілей: Сформувати уміння давати визначення поняттям: «Синхронізація», «Синхронізація в комп'ютерних системах», «Класичні задачі синхронізації», «Програмний алгоритм синхронізації».	Сприйняття теми заняття та її мети.
Мотивація мети	Повідомлення важливості вивчення даної теми: «Синхронізація та блокування потоків» дає вам можливість проводити як професійні практичні так наукові дослідження та володіти мовами системного програмування та методами розробки програм, що взаємодіють з компонентами комп’ютерних систем, знати мережні технології, архітектури комп’ютерних мереж, мати практичні навички технології адміністрування комп’ютерних мереж та їх програмного забезпечення.	Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї.

Продовження табл. 4.5

1	2	3
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Назвіть будь ласка основні операції з потоками та способи реалізації потоків? 2. Назвіть будь ласка два основні ексклюзивні блокування 3. Скажіть, будь ласка, що таке потік?	Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання. Передбачувані відповіді: 1. Створення нового потоку відбувається за допомогою функції pthread_create – в UNIX, CreateThread – в Windows. Основним параметром цих функцій є ім'я процедури, яку необхідно запустити в новому потоці. Windows також передбачена функція CreateRemoteThread, що дозволяє створити потік в іншому процесі. Завершити потік можна зсередини самого цього потоку функцією pthread_exit в UNIX або ExitThread – в Windows. Після цього потік зникає і планувальник його не використовує. Завершити потік можна також з іншого потоку за допомогою функції pthread_cancel – в UNIX або TerminateThread – в Windows. Очікування потоком завершення іншого певного потоку виконується за допомогою функції pthread_join – в UNIX або WaitForSingleObject і WaitForMultipleObject -в Windows. Потік може добровільно надати свою чергу іншому потоку за допомогою функцій pthread_yield – в UNIX або SwitchToThread і SuspendThread – в Windows. 2. Два основні ексклюзивні блокування – це оператор lock та Mutex. 3. Потік – це незалежно планований контекст виконання, що розділяє єдиний адресний простір з іншими потоками свого процесу.

Продовження табл. 4.5

1	2	3
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу – пояснення: «Синхронізація», «Синхронізація в комп'ютерних системах», «Класичні задачі синхронізації», «Програмний алгоритм синхронізації».	Слухають пояснення, конспектують.
Визначення проблемних моментів під час вивчення питань теми та формування ВД	Викладач запитує у здобувачів освіти про проблеми, які виникли у них під час самостійного вивчення теми. Викладач відповідає на поставлені запитання: 1. «Синхронізація дозволяє уникнути помилок узгодженості пам'яті, викликаних через непослідовний доступ до спільної пам'яті». 2. Синхронізація в комп'ютерних системах — це координація роботи процесів таким чином, щоб послідовність їх операцій була передбачуваною. Як правило, синхронізація необхідна при спільному доступі до ресурсів, що розділяються. 3. Класичні задачі синхронізації — це модельні задачі, на яких досліджуються різні ситуації, які можуть виникати в системах з розділюємим доступом та конкуренцією за спільні ресурси.	Питання здобувачів освіти: 1. Навіщо синхронізувати потоки?. 2. Що таке синхронізація в комп'ютерних системах?. 3. Що таке класичні задачі синхронізації
Підведення підсумків	Викладач підводить підсумки проведення консультації: «На консультації ми розглянули складні для вас питання теми для самостійного вивчення. Для перевірці засвоєння складного для вас матеріалу дайте мені, будь ласка, відповідь на запитання: «Що таке синхронізація?»	Здобувачі слухають, відповідають: Синхронізація – це координація процесів потоків для отримання прогнозованого результату. Синхронізація особливо важлива, коли потоки звертаються до тих самих даних. Здобувачі освіти прощаються.

Отже, на цьому етапі ми розробили сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

На заключному етапі представлено контурний конспект з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

Конспект – сукупність взаємозв'язаних опорних сигналів теми.

За обсягом інформації, що представляється, конспекти діляться на повні і контурні (опорні), а за способом подання інформації — на плани-конспекти і конспекти-схеми. План-конспект стисло представляє зміст кожного з пунктів плану. Конспект-схема — це ієрархія понять теми, впорядкованих згідно плану і доповнених основними відомостями.

Контурний конспект заняття з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології) представлено у Додатку.

Висновки до розділу

У розділі розроблено дидактичний проект консультативного заняття з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Програмна інженерія». Подано список використаних джерел та відповідні посилання.

ВИСНОВКИ

У кваліфікаційній роботі теоретично обґрунтовано та перевірено методику професійної підготовки фахівців із цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у вищих навчальних закладах, а також виокремлено пріоритетні напрямки удосконалення професійних компетентностей.

У процесі дослідження уточнено та систематизовано ключові поняття, що визначають суть і структуру процесу професійної підготовки фахівців із цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти, а також визначено пріоритетні напрямки вдосконалення професійних компетентностей.

Виявлено, що професійна підготовка фахівців із цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у вищих навчальних закладах є актуальною проблемою в контексті сучасної професійної педагогіки.

Проаналізовано важливість проблеми професійної підготовки фахівців із цифрових технологій для викладання цього модуля в вищих навчальних закладах. Охарактеризовано систему професійної підготовки фахівців із цифрових технологій для викладання освітнього модуля «Паралельні обчислення мовою C#» у закладах вищої освіти.

Виконано аналіз характеристик об'єктів галузі організації паралельних обчислень з використанням багатопотокового програмування мовою C#. Виконано аналіз освітніх компонент вітчизняних закладів вищої освіти, присвячених вивченню процесів організації паралельних обчислень. За результатами виконаного дослідження виконана постановка завдань нового лабораторного практикуму «Паралельні обчислення мовою C#», який акцентує увагу здобувачів на організації паралельних обчислень з використанням класу Thread та організації синхронізації потоків засобами lock, Mutex та Semaphore.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, зокрема опис предметної області, блок-схеми однопотокових та багатопотокових алгоритмів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio.

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

Виконано огляд ключових аспектів кадрового забезпечення фахівців для викладання освітнього модуля «Паралельні обчислення мовою C#», зокрема встановлено, що в останні роки спостерігається зростання попиту на IT-професії, такі як розробники, системні адміністратори, архітектори, менеджери хмарних технологій, інженери хмарних технологій, аналітики даних та фахівці з кібербезпеки. Встановлено, що є необхідність модернізації освітніх програм, оскільки швидкий розвиток технологій випереджає можливості існуючих навчальних планів.

Розроблено дидактичний проект консультативного заняття з теми «Синхронізація та блокування потоків» дисципліни «Програмна інженерія» для здобувачів спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

За основними результатами дослідження виконана публікація тез доповідей на конференції:

– Коробчук І.П. Лабораторний практикум «Паралельні обчислення мовою C#» // Матеріали VIII міжн. наук.-практ. конф. здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (м. Харків, 14-15 листопада 2024 р.) [упоряд. Г. Г. Михальченко]. Харків, 2024.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# : навч. посіб. Тернопіль: ФОП Паляниця В. А., 2020. 320 с.

2. Поліщук А.М., Ніколюк П.К. Технологія програмування та основні етапи її розвитку. Комп'ютерні технології обробки даних, 2022, 89-91.

3. Прокоп Ю. В., Трофименко О. Г., Толокнов А. А., Дубовой Я. В. Комплексний аналіз підходів до викладання курсів CS1 і CS2 в університетах світу та України. Вісник Черкаського державного технологічного університету. Технічні науки. 2021. № 2. С. 18-30.

4. Самчинська Я.Б., Вінник М.О. Просування й розповсюдження педагогічного програмного забезпечення на ринку України. Інформаційні технології в освіті, 2013. № 15. – С. 210-220.

5. Сурков К., Книшук А., Сорокун С. Інноваційні методи навчання при вивченні об'єктно орієнтованих мов програмування для майбутніх IT-фахівців. Перспективи та інновації науки. 2024. № 4 (38).

6. Онлайн-курс Udemy. Learn Parallel Programming with C# and .NET. URL: https://www.udemy.com/course/parallel-dotnet/

7. Трофименко О.Г., Савєльєва О.С., Прокоп Ю.В., Логінова Н.І., Дика А.І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19.

8. Гордєнко Ю., Таран В. Хмарні обчислення: Конспект лекцій. Електронне мережне навчальне видання. КПІ ім. Ігоря Сікорського, 2022. – С. 1024.

9. Microsoft Learn. Parallel programming in .NET: A guide to the documentation. URL: https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/

10. Гордєнко Ю., Таран В. Хмарні обчислення: лабораторний практикум. Електронне мережне навчальне видання. КПІ ім. Ігоря Сікорського, 2021. – С. 38.

11. Гришанович Т. О. Лабораторний практикум із дисципліни “Паралельні та розподілені обчислення” [Електронний ресурс]; ВНУ імені Лесі Українки. Луцьк : ВНУ імені Лесі Українки, 2022. – С. 50.

12. Зінченко О.В., Іщеряков С.М., Прокопов С.В., Сєрих С.О., Василенко В.В. Хмарні технології. – Навчальний посібник. – К: ФОП Гуляєва В.М., 2020. – С. Минайленко Р. М. Паралельні та розподілені обчислення : навч. посіб. М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2021. – 153 с.

13. Коцовський В.М. Теорія паралельних обчислень. – Ужгород: ПП «АУТДОР–Шарк», 2021. – 188 с.

14. Методичні вказівки до лабораторних робіт з дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 123 «Комп’ютерна інженерія» всіх форм навчання. Частина 2 / Укл.: Р.К. Кудерметов, В.В. Шкарупило, О.В. Польська. – Запоріжжя: НУ «Запорізька політехніка», 2020.

15. Методичні вказівки до лабораторних робіт з дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 123 «Комп’ютерна інженерія» всіх форм навчання. Частина 2 / уклад.: Р.К. Кудерметов, В.В. Шкарупило, О.В. Польська. – Запоріжжя : НУ «Запорізька політехніка», 2020.

16. Паралельні та розподілені обчислення: Методичні вказівки до виконання самостійних робіт для студентів денної форми навчання за спеціальністю 123 «Комп’ютерна інженерія» / уклад. : В.В. Смірнов, Н.В. Смірнова ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 23 с.

17. Chykunov P. Organization of prime number search process using Nvidia CUDA / P. Chykunov, V. Chernetskyi // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 06-07 грудня 2023 р.). – Харків: ННППІ УІПА, 2023. С. 40-41.

18. Бородкіна І. Інженерія програмного забезпечення: навчальний посібник. Центр учбової літератури, 2021. – 204 с.

19. Кипаренко Д. О. Методи обробки великих даних в розподілених хмарних системах : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 123 Комп'ютерна інженерія / Д. О. Кипаренко ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2024. – 66 с.

20. Климова І. М. Моделі та методи побудови паралельних алгоритмів / І. М. Климова, А. В. Черниш, В. М. Федорченко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доп. 14-ї міжнар. наук.-техн. конф., 25-26 квітня 2024 р. – Харків : Impress., 2024. – С. 129.

21. Multithreading and Parallel Programming in C#. URL: https://www.udemy.com/course/parallel-csharp/

22. Методичні вказівки до практичних занять і розрахунково-графічної роботи з дисциплін «Паралельні та розподілені обчислення та CKLOUD-технології» / укладачі : М. А. Мірошник, Ю. В. Савін. - Харків, 2023. - 93 с.

23. Розподілені та паралельні обчислення. Лабораторний практикум для студентів спеціальності 121 "Інженерія програмного забезпечення" освітньої програми "Інженерія програмного забезпечення" першого (бакалаврського) рівня [Електронний ресурс] / уклад. С. В. Мінухін, Ю. В. Савін; Харківський національний економічний університет ім. С. Кузнеця. – Харків : ХНЕУ ім. С. Кузнеця, 2023. – 170 с.: іл. – Загол. з титул. екрану. – Бібліогр.: с. 169.

24. Онлайн-курс Coursera. Introduction to High-Performance and Parallel Computing. URL: https://www.coursera.org/learn/scala-parallel-programming

25. Чикунов П. О. Можливості мови C# для організації паралельних обчислень / Матеріали Міжнародної науково-практичної конференції «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. – С. 140-143.

26. Чикунов П.О. Розробка програмного середовища візуалізації та аналізу графових моделей / П.О. Чикунов, М.О. Сафонов, Д.А. Неізмайлов // Матеріали III Всеукраїнської науково-практичної інтернет-конференції

«Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Бахмут, 29-30 листопада, 2020 р.). – Бахмут: ННППІ УІПА, 2020. – С. 48-49.

27. Силабус навчальної дисципліни технології розподілених систем та паралельних обчислень. URL: https://suitt.edu.ua/wp-content/uploads/2024/01/ok-27-tekhnolohii-rozpodilenykh-system-ta-paralelnykh-obchyslen.pdf

28. Технології розподілених систем та паралельних обчислень. URL: https://dls.viti.edu.ua/course/info.php?id=411

29. Парфенюк О.В. Силабус навчальної дисципліни «Технології розподілених систем та паралельних обчислень» для здобувачів вищої освіти першого ступеня «бакалавр», які навчаються за освітньо-професійною програмою «Комп'ютерні технології», спеціальності 015 Професійна освіта. Рівне. НУВГП. 2020. 14 стор. URL: https://ep3.nuwm.edu.ua/19186/

30. Корочкін О.В. Силабус ОК «Паралельні та розподілені обчислення». URL: https://comsys.kpi.ua/paralelni-ta-rozpodileni-obchislennya

31. Чикунов П.О. Силабус навчальної дисципліни «Технології розподілених систем та паралельне обчислення» освітньо-професійної програми «Комп’ютерні науки». URL: https://docs.google.com/document/d/1ZnEaAWrH4d_D_EtroVhV9GclJ9ocSFXN/edit

32. Антонюк Л.Л. Компетентністний підхід у вищій освіті: світовий досвід навч. пос. Київ : КНЕУ, 2016.66 с.

33. Професійна педагогіка : Підручник / Авт. : О. В. Грабовський, Л. В.Коломієць, О. С. Савельева, А. В.Семенова, В. Ф.Яні; за заг. ред. А. В.Семенової. – Одеса : Бондаренко М. О., 2020. – 575 с.

34. Професійна педагогіка: навч. посібник для вищих навч. закладів/ В. І. Жигірь, О. Чернєга ; за ред. М. В. Вачевського. - Київ: К.: Кондор, 2016. – 336 с. 978-966-351-359-1. Код 233704.

35. Зайченко І. В. Теорія і методика професійного навчання: навч.

посібник. 2-е вид., доповн. і переробл. К.: Видавництво Ліра-К, 2016. 580 с.

36. Методика формування пошуково-дослідницьких умінь майбутніх інженерів-педагогів у процесі професійної підготовки: колективна монографія / В.В. Кулешова, В.В. Мальована. – Артемівськ: ННППІ УІПА, 2012. – 264 с. (власний внесок: Р1; Р2 с.86-92; Р3; 9,8 д.а.).

37. Формування професійної компетентності викладачів технічних дисциплін: колективна монографія / В.В.Кулешова, В.В.Мальована, Ю.С.Бобрикова. – Х., 2020. – 206 с. (власний внесок: Р1 с.8-94; Р2 с.95-100; Р3с.146-159; 6,5 д.а.).

38. Кулешова В.В., Мальована В.В. Особливості особистості викладача технічних дисциплін у вищих навчальних закладах/ Проблеми інженерно-педагогічної освіти. Збірник наукових праць. №50-51 Харків: УІПА,– 2016 р., – С.322-329

39. Rулешова В.В., Мальована В.В. Формування професійних методичних умінь у майбутніх інженерів-педагогів економічного профілю / Міжнародний науковий журнал «ІНТЕРНАУКА». №7 (29) Київ:– 2017 р., – С.26-29

40. Кулешова В.В. Формування креативної компетентності майбутніх інженерів у процесі професійної підготовки / Проблеми інженерно-педагогічної освіти. Збірник наукових праць. № 58 Харків: УІПА,– 2018 р., – С.21-26

41. Коваленко А.В. Шляхи забезпечення формування безперервної освіти: школа – ПЗО – Фаховий коледж – Академія. Роль закладів фахової передвищої та 194 професійної освіти в системі безперервної освіти: матеріали VII наук.-практ. конф., м. Одеса, 25 бер.2020 р. Одеса, 2020. С.21-24.

42. Олійник В. В. Відкрита післядипломна педагогічна освіта: нові моделі та форми професійного розвитку / Освіта дорослих у перспективі змін: інновації, технології, прогнози: колективна монографія / За ред.. А. Василюк, А. Стоговського. – Ніжин: Видавець ПП Лисенко М.М., 2017. – 248 с. С. 93–108.

43. Ортинський В. Л. Педагогіка вищої школи: навч. посіб. / Ортинський В. Л. – Центр учбової літератури, 2017. – 472 с

44. Професійна освіта України на шляху до євроінтеграції (1992–2017) / науков. ред. Н.Г. Ничкало; упорядники: Л.В. Горбань, В.П. Тименко. – К.: ДП «Інформ.-аналіт. агенство», 2018. – 358 с.

45. Сисоєва С.О. Теорія і практика вищої освіти: навч. посібник / С.О. Сисоєва, І.В. Соколова. – К., 2016. – 338 с. – Режим доступу: http://lib.iitta.gov.ua/711948/1/Sysoieva_Socolova_2016_pos..pdf

46. Теорія та методика викладання фахових дисциплін у ЗВО: навчально-методичний посібник / укладач І. В. Казанжи – Миколаїв : СПД Румянцева, 2018. – 154 с.

47. Теорія і практика вищої професійної освіти в Україні : навч. посіб. для магістрантів зі спеціальності 011 «Освітні, педагогічні науки» / [авт.-укл.: Т.О.Дороніна]. – Кривий Ріг : КДПУ, 2018. – 250 с.

48. Методика професійного навчання: метод. вказ. по виконанню курсової роботи для здобувачів освіти освітнього ступеня «бакалавр» денної та заочної форми навч. інженерно-педагогічних спеціальностей / ННППІ Укр. інж.-пед. акад. ; упоряд. : В. В. Кулешова, В.В. Мальована, Ю.С. Бобрикова ; за заг. ред. д-ра пед. наук, проф.В. В. Кулешової. – Бахмут, 2022. – 92 с.

49. Методика професійного навчання : конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 1 / О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 200 с.

50. Методика професійного навчання: конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 2/ О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 180 с.

51. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика

професійного навчання: дидактичне проектування: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 204 с.

52. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика професійного навчання: основні технології навчання: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 174 с.

53. Методика професійного навчання: дидактичне проектування: конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут: [б. в.], 2017. - 100 с.

54. Методика професійного навчання: основні технології навчання: конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут: [б. в.], 2017. - 101 с.

55. Теорія і методика професійного навчання : навч. посібник. – 2-е вид., доповн. і переробл. / І.В.Зайченко. – К.: Видавництво Ліра-К, 2016. – 580 с.

56. Методика професійного навчання: навч. посібник для вищих навч. закладів інж.-пед. спец. для традиційної та дистанційної форм навчання. Ч. 1: Дидактичне проектування/ О. Е. Коваленко, Н.О. Брюханова, Н.В. Корольова; Укр. інж.- пед. академія. - 2-ге вид., перероб. та доп.. - Х.: ФОП Шевченко С. О., 2010. - 264 с.

ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ НА ТЕМУ «СИНХРОНІЗАЦІЯ ТА БЛОКУВАННЯ ПОТОКІВ»

Синхронізація – це координація процесів потоків для отримання прогнозованого результату. Синхронізація особливо важлива, коли потоки звертаються до тих самих даних. Конструкції синхронізації можна розділити на чотири категорії:

1. **Прості методи блокування.** Вони чекають на закінчення іншого потоку або протягом певного періоду часу. Методи Sleep, Join є найпростішими методами блокування.

2. **Блокуючі конструкції.** Вони обмежують кількість потоків, які можуть виконувати певну активність або виконувати секцію коду за один раз. Ексклюзивні блокуючі конструкції найбільш поширені - вони дозволяють лише один потік за раз і дозволяють конкуруючим потокам отримувати доступ до загальних даних, не заважаючи один одному. Стандартними ексклюзивними замикаючими конструкціями є lock, Mutex, SpinLock. Неексклюзивні блокуючі конструкції – це Semaphore, SemaphoreSlim.

3. **Сигналізаційні конструкції.** Вони дозволяють потоку зупинятися до отримання повідомлення від іншого, уникаючи необхідності неефективного опитування. Найчастіше використовувані сигнальні пристрої- це дескриптори очікування подій і клас Monitor.

4. **Неблокуючі конструкції синхронізації.** Вони захищають доступ до загального поля, викликаючи примітив процесора. У CLR і C# надають такі неблокуючі конструкції: Thread.MemoryBarrier, Thread.VolatileRead, Thread.VolatileWrite, за volatile ключовим словом, і клас Interlocked .

Блокування важливе для всіх, крім останньої категорії. Потік вважається заблокованим, коли його виконання припиняється з будь-якої причини, наприклад, при виклику методу Sleep або очікуванні завершення іншого через виклик Join або EndInvoke. Блокований потік негайно віддає свій процесорний час, і з цього моменту не споживає процесорний час, поки не буде виконано

його умову блокування. Перевірити, чи заблокований потік можна через його властивість. Коли потік блокується чи розблокується, операційна система перемикає контекст. Розблокування відбувається одним із чотирьох способів:

1. за умовою блокування
2. за тайм-аутом операції (якщо заданий тайм-аут)
3. шляхом переривання через Thread.Interrupt
4. шляхом переривання через Thread.Abort

Статус виконання потоку можна дізнатися через його властивість ThreadState. Ця властивість повертає елемент перерахування типу ThreadState. Більшість значень, однак, є надмірними, невикористовуваними або застарілими.

Властивість ThreadState корисна для діагностичних цілей, але непридатна для синхронізації, тому що стан потоку може змінитися між тестуванням ThreadState і діє на основі цієї інформації.

Виняткове блокування використовується для того, щоб тільки один потік міг вводити певні розділи коду за раз. Два основні ексклюзивні блокування – це оператор lock та Mutex.

Оператор **lock** швидше та зручніше. Розглянемо наступний приклад:

class ThreadUnsafe

```
class ThreadUnsafe
{
    static int _val1 = 1, _val2 = 1;
    static void Go()
    {
        if (_val2 != 0)
            Console.WriteLine(_val1 / _val2);
        _val2 = 0;
    }
}
```

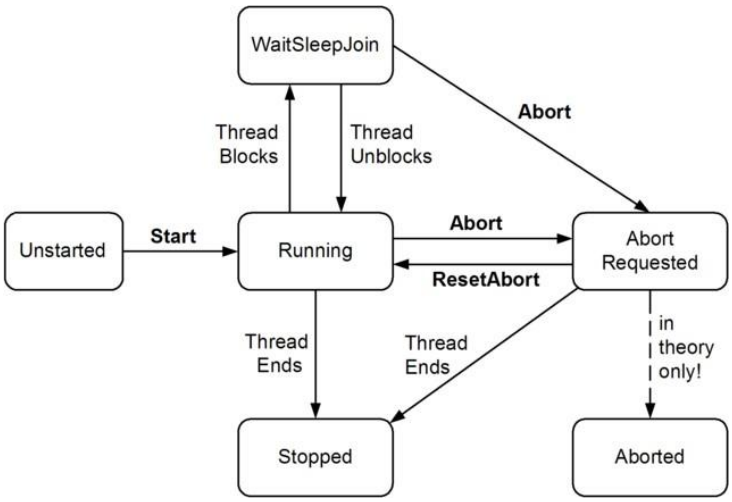

Рис. 1. Діаграма станів потоків та способів зміни стану

Цей клас не є потокобезпечним, якщо метод Go буде викликаний двома потоками одночасно, то можна було б отримати помилку поділу на нуль, тому що _val2 можна було б встановити нуль в одному потоці прямо, оскільки інший потік знаходився між виконанням оператора і Console.WriteLine(). Ось як за допомогою опреатора lock можна вирішити проблему:

```
class ThreadSafe
{
    static readonly object _locker = new object();
    static int _val1, _val2;
    static void Go()
    {
        lock (_locker)
        {
            if (_val2 != 0) Console.WriteLine(_val1 / _val2);
            _val2 = 0;
        }

    }
}
```

Тільки один потік може блокувати синхронізуючий об'єкт (в даному випадку _locker) за раз, і будь-які потоки, що конфліктують, блокуються доти,

поки блокування не буде звільнено. Якщо більш ніж один потік посилається на блокування, вони ставляться в чергу і надають блокування на основі "першим прийшов, першим обслужений". І тут ми захищаємо логіку всередині методу Go, і навіть поля _val1 і _val2.

Оператор lock насправді є синтаксичним ярликом для виклику методів Monitor.Enter та Monitor.Exit за допомогою блоку try/finally. Ось що насправді відбувається в Go метод попереднього прикладу: Monitor.Enter (_locker);

```
try
{
    if (_val2 != 0)
        Console.WriteLine(_val1 / _val2);
    _val2 = 0;
}
finally
{
    Monitor.Exit(_locker);
}
```

Будь-який об'єкт, видимий для кожного із сторонніх потоків, може використовуватися як об'єкт синхронізації, підпорядкований одному жорсткому правилу: він має бути типом посилання. Синхронізуючий об'єкт зазвичай є приватним (оскільки це допомагає інкапсулювати логіку блокування) і є екземпляром або статичним полем.

Недоліком блокування є те, що ви не інкапсулюєте логіку блокування, тому стає важче запобігти надмірному блокуванню. Блокування може також просочуватися через межі домену програми (в рамках одного й того самого процесу).

Ви також можете блокувати локальні змінні, захоплені лямбда-виразами або анонімними методами. Блокування не обмежує доступ до об'єкта, що синхронізує. Іншими словами, x.ToString() він не блокуватиметься, тому що викликав інший потік lock(x); обидва потоки повинні викликати lock(x) для блокування.

Як основне правило, вам необхідно заблокувати доступ до будь-якої доступної для запису області. Навіть у найпростішому випадку – операція присвоєння в одному полі – ви повинні враховувати синхронізацію. У наступному класі ні метод Increment, ні метод Assign не є потокобезпечними:

```
class ThreadUnsafe
{
    static int _x;
    static void Increment()
    { _x++; }
    static void Assign()
    { _x = 123; }
}
```

Ось потокобезпечні версії Increment та Assign:

```
class ThreadSafe
{
    static readonly object _locker = new object();
    static int _x;
    static void Increment()
    { lock (_locker) _x++; }
    static void Assign()
    { lock (_locker) _x = 123; }
}
```

Якщо група змінних завжди читається та записується в межах одного блокування, ви можете сказати, що змінні читаються та записуються атомарно. Припустимо, що поля x та y завжди читаються і призначаються всередині lock об'єкта locker:

```
lock (locker) { if (x! = 0) y / = x; }
```

Можна сказати, що до змінних x і y ми отримуємо доступ атомарно, тому що кодовий блок не може бути розділений або витіснений діями іншого потоку таким чином, щоб він змінив x або y до обчислення результату.

Атомність, що надається блокуванням, порушується, якщо викидається виняток усередині блоку lock. Наприклад, розглянемо наступний приклад:

```
decimal _savingsBalance, _checkBalance;
void Transfer(decimal amount)
{
    lock (_locker)
    {
        _savingsBalance += amount;
        _checkBalance -= amount + GetBankFee();
    }
}
```

Якби виняток був кинутий методом GetBankFee(), банк втратив би гроші. У цьому випадку ми могли б уникнути проблеми, викликавши GetBankFee раніше. Рішенням для складніших випадків є реалізація логіки «відкату» всередині catch або finally блоку.

Ще один інструмент управління синхронізацією потоків представляє клас **Mutex**, що також знаходиться в просторі імен System.Threading. Цей клас є класом-оболонкою над відповідним об'єктом ОС Windows. М'ютекс є взаємно виключаючим синхронізуючим об'єктом. Це означає, що він може бути отриманий потоком лише по черзі. М'ютекс призначений для тих ситуацій, в яких загальний ресурс може одночасно використовуватися тільки в одному потоці. Припустимо, що системний журнал спільно використовується в кількох процесах, але тільки в одному з них дані можуть записуватись у файл цього журналу в будь-який момент часу. Для синхронізації процесів у цій ситуації ідеально підходить м'ютекс. Придбання та звільнення Mutex займає кілька мікросекунд – приблизно в 50 разів повільніше за lock.

У конструкторі класу Mutex вказується, чи повинен мьютексом спочатку володіти потік, що викликає, і його ім'я. Mutex має кілька конструкторів. Нижче наведено два найбільш уживані конструктори:

```
public Mutex()
public Mutex(bool initiallyOwned)
```

У першій формі конструктора створюється м'ютекс, яким спочатку ніхто не володіє. А в другій формі вихідним станом м'ютексу заволодіває

викликаючий потік, якщо параметр спочаткувиконує логічне значення true. Інакше м'ютексом ніхто не володіє.

Щоб отримати м'ютекс, у коді програми слід викликати метод WaitOne() для цього мьютексу. Метод WaitOne() успадковується класом Mutex від класу Thread.WaitHandle. Метод WaitOne() очікує до тих пір, поки не буде отримано м'ютекс потоку, з якого він був викликаний. Отже, цей метод блокує виконання викликаючого потоку до тих пір, поки стане доступним зазначений м'ютекс. Наведемо приклад створення та використання м'ютексу:

```
class OneAtATimePlease
{
    static void Main()
    {
        using (var mutex = new Mutex(false, "Mutex1"))
        {
            if (!mutex.WaitOne(TimeSpan.FromSeconds(3), false))
            {
                Console.WriteLine("Another app instance is
running. Bye!");
                return;
            }
            RunProgram();
        }
        Console.ReadLine();
    }

    static void RunProgram()
    {
        Console.WriteLine("Running. Press Enter to exit");
        Console.ReadLine();
    }
```

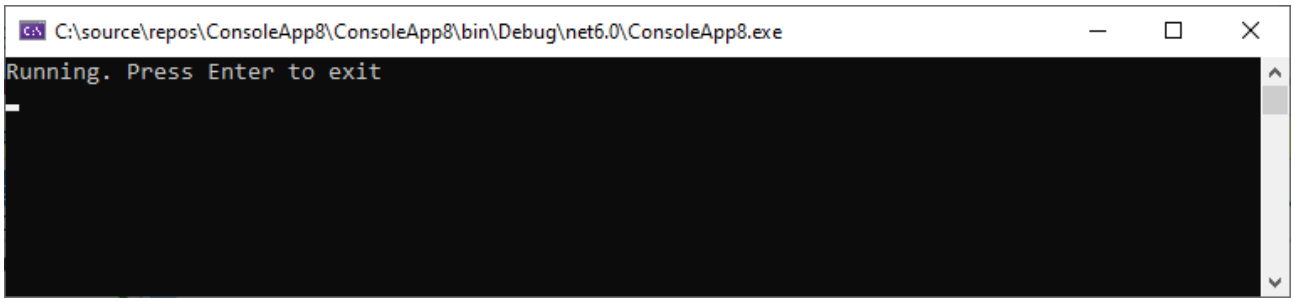

Рис. 2. Результат створення та використання м'ютексу

Ще один інструмент, який пропонує платформа. NET для управління синхронізацією, представляють семафори. Семафори дозволяють обмежити доступ певною кількістю об'єктів. Семафор аналогічний Mutex, за винятком того, що семафор не має «власника». Будь-який потік може викликати метод Release семафору, тоді як з Mutex і Lock, тільки потік, який отримав блокування, може його звільнити. Семафори можуть бути корисні в обмеженні паралелізму - запобігання надто великій кількості потоків від виконання певної частини коду одночасно. У наступному прикладі п'ять потоків намагаються запровадити нічний клуб, який дозволяє одночасно задіяти лише три потоки:

```
class TheClub
{
    static SemaphoreSlim _sem = new SemaphoreSlim(3,3);
    static void Main()
    {
        for (int i = 1; i <= 5; i++) new Thread(Enter).Start(i);
    }
    static void Enter(object id)
    {
        Console.WriteLine(id + "wants to enter");
        _sem.Wait(); Console.WriteLine(id + "is in!");
        Thread.Sleep(1000 * (int)id); _sem.Release();
        Console.WriteLine(id + "is leaving");
    }
}
```

Для створення семафору використовується конструктор класу Semaphore: static Semaphore sem = new Semaphore (3, 3). Його конструктор приймає два параметри: перший вказує, якій кількості об'єктів спочатку буде доступний семафор, а другий параметр вказує, яка максимальна кількість об'єктів буде використовувати цей семафор. В даному випадку у нас тільки три потоки можуть одночасно перебувати в нічному клубі, тому максимальне число дорівнює 3. Основний функціонал зосереджений у методі Enter, який виконується в потоці. На початку для очікування на отримання семафору використовується метод Wait. Після того, як у семафорі звільниться місце, цей потік заповнює вільне місце та починає виконувати всі подальші дії. Після закінчення читання ми звільняємо семафор за допомогою методу Release. Після цього у семафорі звільняється одне місце,

Обробники подій EventWaitHandle використовуються для сигналізації. Сигналізація - це коли один потік чекає, доки він не отримає повідомлення від іншого. Обробники подій - це найпростіші з сигнальних конструкцій, і вони пов'язані з подіями C#. Вони бувають трьох видів: AutoResetEvent, ManualResetEvent. Вони засновані на загальному EventWaitHandle класі, від якого вони отримують всю свою функціональність. AutoResetEvent схоже на турнікет із квитками: вставка квитка дозволяє пройти рівно одній людині. "Auto" в назві класу відноситься до того факту, що відкритий турнікет автоматично закривається або "скидається" після того, як хтось через нього пройде. Ви можете створити об'єкт AutoResetEvent двома способами. Перший – через виклик його конструктора:

```
var auto = new AutoResetEvent(false);
```

Інший спосіб створення AutoResetEvent:

```
var auto = new EventWaitHandle (false, EventResetMode.AutoReset);
```

Застосовуються події дуже просто. Для події типу ManualResetEvent порядок застосування наступний. Потік, що очікує певну подію, викликає метод WaitOne() для подійного об'єкта, що представляє цю подію. Якщо подійний об'єкт перебуває у сигнальному стані, відбувається негайне

повернення з методу WaitOne(). В іншому випадку виконання викликаючого потоку припиняється доти, доки не буде отримано повідомлення про подію. Як тільки подія відбудеться в іншому потоці, цей потік встановить подійний об'єкт сигнальний стан, викликавши метод Set(). Після встановлення подійного об'єкта сигнальний стан відбудеться негайне повернення з методу WaitOne(), і перший потік відновить своє виконання. А в результаті виклику методу Reset() подійний об'єкт повертається до несигнального стану.

Подія AutoResetEvent відрізняється від події типу ManualResetEvent лише способом встановлення у вихідний стан. Якщо для події типу ManualResetEvent подійний об'єкт залишається в сигнальному стані до тих пір, поки не буде викликаний метод Reset(), то для події типу AutoResetEvent подійний об'єкт автоматично переходить у несигнальний стан, як тільки потік, що чекає на цю подію, отримає повідомлення про нього і відновить своє виконання. Тому, якщо застосовується подія типу AutoResetEvent, то викликати метод Reset() необов'язково.

У наступному прикладі починається потік, який просто чекає, поки не буде сигналізований іншим потоком:

```
class BasicWaitHandle
{
    static EventWaitHandle _waitHandle = new
            AutoResetEvent(false);
    static void Main()
    {
        new Thread(Waiter).Start(); Thread.Sleep(1000);
        _waitHandle.Set();
    }
    static void Waiter()
    {
        Console.WriteLine("Waiting...");
        _waitHandle.WaitOne(); Console.WriteLine("Notified");
    }
}
```

ДОДАТОК Б ПУБЛІКАЦІЯ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ЛАБОРАТОРНИИ ПРАКТИКУМ «ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ МОВОЮ C#»

Автор: Коробчук І.П., магістр
Науковий керівник: Чикунов П.О., к.т.н., доц.
Бахмутський навчально-науковий професійно-педагогічний інститут ХНУ імені В. Н. Каразіна

З кожним роком паралельні обчислення стають невід'ємною частиною нашого повсякденного життя. Їх вплив поширюється на різні сфери діяльності, починаючи від бізнесу і закінчуючи розвагами, освітніми програмами та медициною. У сучасних задачах обробки даних часто потрібно працювати з величезними наборами даних. Використання паралельних обчислень дозволяє не тільки зберігати великі обсяги даних, але й здійснювати їхню швидку обробку в режимі реального часу. Розробники повинні мати високий рівень знань у мовах програмування, таких як C#, Java, C++, а також бути в курсі останніх тенденцій в API-інтерфейсах та розробці програмних інструментів [1].

З впровадженням нейромереж та штучного інтелекту зростає можливість автоматизації багатьох рутинних процесів, але сучасні нейронні мережи потребують значної обчислювальної потужності. Паралельні обчислення дозволяють розподілити навантаження між всіма ядрами центрального процесора або графічними процесорами, що значно скорочує час навчання.

Завдяки паралельним обчисленням можна масштабувати моделі на великій кількості обчислювальних ресурсів та одночасно проводити експерименти з різними конфігураціями нейромереж, що сприяє більш швидкому пошуку оптимальних гіперпараметрів. У великих дата-центрах або на клауд-платформах паралельні обчислення забезпечують більш ефективне використання апаратних ресурсів, що знижує вартість навчання.

Виконано аналіз освітніх компонент вітчизняних закладів вищої освіти, присвячених вивченню процесів організації паралельних обчислень. Аналіз запропонованих підходів показав, що необхідно враховувати конкретні потреби розробника прикладного проекту, доступних ресурсів і його знань у галузі паралельного програмування. За результатами виконаного дослідження виконана постановка завдань нового лабораторного практикуму «Паралельні обчислення мовою C#», який акцентує увагу здобувачів на організації паралельних обчислень з використанням класу Thread та синхронізації потоків засобами lock, Mutex та Semaphore. Розроблено завдання лабораторних робіт, приклади виконання робіт (опис предметної області, блок-схеми однопотокових та багатопотокових алгоритмів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio).

На рис. 1 показано результати тестового виконання програми обчислення суми квадратів елементів масиву з 1.8 мільярда цілих чисел із використанням блокування потоків засобами lock, Mutex та Semaphor для різної кількості потоків. Порівняння середнього часу виконання програм дозволяє зробити висновки про те, що блокування потоків засобами Semaphor найменш гальмує обчислювальний процес. Також встановлено, що збільшення кількості потоків з

4 до 8 та до 12 впливає на зростання продуктивності обчислення.

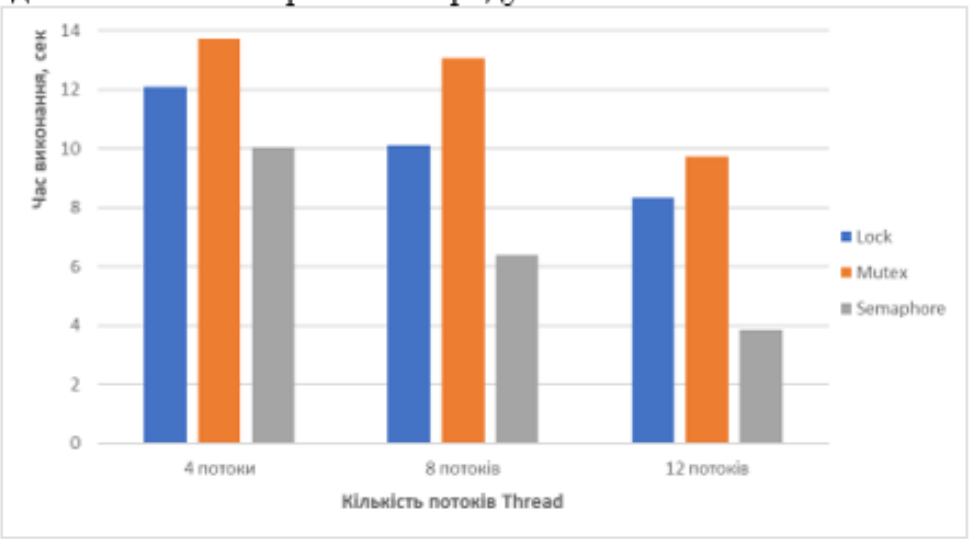

Рисунок 1 – Порівняння середнього часу виконання програм з різними типами блокування потоків

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

Список використаних джерел

1. Чикунов П. О. Можливості мови C# для організації паралельних обчислень / Матеріали Міжнародної науково-практичної конференції «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. – С. 140-143.