ПВНЗ "Буковинський університет"
Факультет інформаційних технологій та економіки
Кафедра комп'ютерних систем і технологій

Схвалено та затверджено на засіданні
науково-методичної ради факультету
протокол №1 від 30.09.2021

ЗАТВЕРДЖУЮ:
декан факультету
інформаційних технологій та економіки
_____ Тетяна Штерма

СИЛАБУС
навчальної дисципліни

«Інформаційні технології паралельних розрахунків»

обов'язкова навчальна дисципліна

Освітньо-професійна програма комп'ютерні науки

Спеціальність 122 - комп'ютерні науки

Галузь знань 12 – інформаційні технології

Рівень вищої освіти другий магістерський

Факультет інформаційних технологій та економіки

Мова навчання українська

Розробник: Гаць Богдан Миколайович, к.т.н., доцент
Профайл викладача: http://www.bukuniver.edu.ua/fakulteti/ekonom%D1%96chnij-fakultet/kafedra-kompyuternix-sistem-%D1%96-texnolog%D1%96j/
Контактний телефон: (0372) 52-00-12
Email: gatsbn@gmail.com

Консультації: четвер з 10:00 до 16:00

1. **Опис навчальної дисципліни**

<table>
<tr><td rowspan="2">Галузь знань, спеціальність, освітня програма, освітній рівень</td><td colspan="2" rowspan="2">Загальні характеристики</td><td colspan="2">Навчальне навантаження з дисципліни</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td rowspan="2"><u>Галузь знань</u>
12 – інформаційні технології</td><td colspan="2" rowspan="2">Вибіркова</td><td colspan="2">Курс підготовки:</td></tr>
<tr><td>4-й</td><td></td></tr>
<tr><td rowspan="2"><u>Спеціальність</u>

122 – комп'ютерні науки</td><td>Загальна кількість кредитів ЄКТС</td><td>4</td><td colspan="2">Семестр підготовки:</td></tr>
<tr><td>Загальна кількість годин</td><td>120</td><td>9-й</td><td></td></tr>
<tr><td rowspan="4"><u>Освітня програма</u>

«Комп'ютерні науки»</td><td rowspan="2">Кількість аудиторних годин</td><td rowspan="2">14</td><td colspan="2">Лекції</td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="2">Кількість годин самостійної роботи</td><td rowspan="2">106</td><td colspan="2">Практичні, семінарські</td></tr>
<tr><td>14 год.</td><td></td></tr>
<tr><td rowspan="8"><u>Освітній рівень</u>

магістерський</td><td colspan="3" rowspan="8">Мова навчання - українська</td><td colspan="2">Лабораторні</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Самостійна робота</td></tr>
<tr><td>106 год.</td><td></td></tr>
<tr><td colspan="2">Форма підсумкового контролю</td></tr>
<tr><td>Диф. залік</td><td></td></tr>
</table>

2. Мета та завдання навчальної дисципліни

Дисципліна призначена для формування знань про основні засоби побудови паралельних та розподілених програм.

Метою викладання дисципліни є вивчення студентами принципів побудови паралельних та розподілених програмних додатків для різноманітних комп'ютерних систем, а також придбання практичних навичків щодо створення, тестування та експлуатації паралельного програмного продукту з використанням сучасних пакетів та стандартів паралельного програмування.

Предметом дисципліни є стандарти паралельного та розподіленого програмування та їх реалізацій.

Основними задачами дисципліни:

– ознайомлення студентів з основними парадигмами паралельного програмування;
– вивчення стандартів паралельного програмування (таких як MPI та OpenMP) та їх реалізацій;
– придбання практичних навиків використання пакетів паралельного програмування;
– підготовка до виконання дипломних проектів та кваліфікаційних робіт, тематика яких пов'язана з дослідженням та проектуванням паралельних алгоритмів прикладних задач.

Вивчення дисципліни безпосередньо базується на знаннях і вміннях, отриманих студентами при опануванні дисциплін „Введення в сучасні операційні системи і середовища", „Програмування", „Структури даних та алгоритми" .

Отримані знання й навички студенти можуть використати при написанні випускних і дипломних робіт.

Засвоєння матеріалу дисципліни необхідно для отримання наступних компетенцій:

— створювати та налагоджувати паралельну і розподілену систему;
— створювати паралельні алгоритми для розв'язування різноманітних задач;
— аналізувати ефективність паралельного алгоритму;
— реалізувати паралельний алгоритм для комп'ютерних систем з різними паралельними архітектурами;
— тестування та аналіз роботи паралельного додатку.

Вивчення дисципліни направлено на розв'язання наступних типових задач:

1. Вибір найбільш ефективного паралельного алгоритму розв'язування поставленої задачі для реалізації для заданої паралельної комп'ютерної систем
2. Реалізація паралельного алгоритму за допомогою конкретного пакету паралельного програмування

Курс складається з наступних частин: теоретичної (лекції), практичної (завдання, лабораторні роботи, тести для самоперевірки) та самостійної роботи.

3. Компетентності та результати навчання, формуванню яких сприяє дисципліна (взаємозв'язок з нормативним змістом підготовки здобувачів вищої освіти, сформульованим у термінах результатів навчання):

Інтегральна компетентність. Здатність розв'язувати складні спеціалізовані задачі та практичні проблеми у галузі комп'ютерних наук або у процесі навчання, що передбачає застосування теорій та методів комп'ютерних наук, інформаційних технологій і характеризується комплексністю та невизначеністю умов.

Загальні компетентності (ЗК)

Здатність до пошуку, оброблення та аналізу інформації з різних джерел (ЗК7).

Здатність приймати обґрунтовані рішення (ЗК11).

Здатність працювати в команді (ЗК12).

Здатність спілкуватися з нефахівцями своєї галузі (з експертами з інших галузей) (ЗК13).

Здатність працювати автономно (ЗК14).

Фахові компетентності (ФК)

Знати принципи функціонування та технології віртуалізації серверних систем, архітектури та стандарти комунікаційних засобів розподілених обчислень, протоколи захисту інформації, яка циркулює в інформаційно-комунікаційних системах (ФК1);

Знати класифікацію хмарних обчислень, особливості та характерні ознаки звичайного хостингу веб-ресурсів, оренди віртуальних приватних машин та систем
хмарних обчислень (ФК2);

Знати концепції комп'ютерної реалізації моделей предмету дослідження на основі алгоритмічного, структурного, об'єктно-зорієнтованого, компонентного, аспектно-орієнтованого, сервіс-орієнтованого, мультиагентного та інших сучасних підходів, використовувати концепції паралельної обробки інформації (ФК3);

Вміти виконувати розробку програмного забезпечення окремих функціональних задач для інформаційних управляючих систем (ФК7);

Програмні результати навчання

Здатність демонструвати знання систем хмарних обчислень, архітектури та стандартів комунікаційних засобів розподілених обчислень, концепцій паралельної обробки інформації та здатність до використання отриманих знань у вирішенні практичних завдань (ПРН2);

Здатність до використання алгоритмів управління при проектуванні та подальшій експлуатації інформаційних систем та технологій (ПРН5);

Обізнаність у існуючих інформаційних технологіях для вирішення професійних задач фахівців у ІТ-галузі та здатність до їх обґрунтованого вибору, налаштування та подальшої експлуатації (ПРН8);

Здатність до автономної роботи для вирішення конкретних професійних та дослідницьких завдань (ПРН15).

4. Програма навчальної дисципліни

Модуль 1

Тема 1. Введення до предмету. Класифікація паралельних обчислювальних систем: SMP-системи, кластери, MMP-системи. Системи із загальною й розподіленою пам'яттю. Архітектура NUMA й ccNUMA. Організація когерентності багаторівневої ієрархічної пам'яті в SMP - системах. Моделі паралельних обчислень.

Тема 2. Механізми керування процесами. Типові задачі синхронізації паралельних процесів: задача взаємного виключення, «виробник-споживач», «читачі-письменники», « філософи, що обідають». Семафори, множинні семафори, рахункові семафори. Механізм синхронізації паралельних процесів монітор. Приклад використання монітора. Реалізація моніторів за допомогою семафорів.

Тема 3. Взаємодія між процесами ОС Unix. Процеси й потоки в сучасних ОС. Створення процесів в Unix. Системні виклики fork(), wait(), exec(). Анонімні канали. Канали FIFO. Ідентифікатори й імена в IPC. Повідомлення процесів в ОС Unix. Семафори System V. Поділювана пам'ять в Unix.

Тема 4. Потоки ОС Unix стандарту POSIX. Створення потоку в Unix. Атрибути потоку. Очікувані й від'єднанні потоки. Скасування потоку в Unix: асинхронне скасовування, синхронне скасовування, потоки, які не можна скасувати. Потокові дані. Оброблювач очищення. Очищення потокових даних у C++. Потокові семафори. Мютекси. Умовні змінні POSIX. Приклад використання мютексів і потокових семафорів.

Модуль 2

Тема 5. Технологія MPI. Призначення MPI. Модель MPI-додатка. Комунікатори. Функції ініціалізації й завершення роботи. Етапи передачі повідомлень між паралельними процесами MPI. Функції передачі повідомлень між процесами типу «один-одному». Типи даних MPI. Колективні комунікації. Розподілені операції в MPI. Створення нових типів даних MPI. Створення розподілених операцій. Топології процесів. Створення декартової топології процесів в MPI-додатках. Приклад використання декартової топології процесів.

Тема 6. Технологія OpenMP. Призначення OpenMP. Модель OpenMP-додатка. Директива паралельної обрабки parallel. Директива розподілення роботи for. Директиви розподілення роботи sections та section. Директиви single та master. Директиви tasks та taskwait. Директиви

синхронізації barrier, ordered, critical, atomic. Спільні та приватні змінні. Функції середовища виконання. Функції блокування та синхронізації. Змінні оточення. Алгоритми планування паралельного виконання циклів (static, dynamic, guided, runtime scheduling).

Тема 7. Моделювання та аналіз параллельних обчислень. Граф інформаційних залежностей послідовного алгоритму. Характеристики часу виконання паралельного алгоритма. Теореми про оцінки часу виконання паралельного алгоритма. Показники ефективності паралельного алгоритма. Каскадна схема сумування. Модифікована каскадна схема сумування. Оцінка максимально допустимого паралелизму(закон Амдаля, ефект Амдаля, закон Густавсона – Барсиса).

5. Структура навчальної дисципліни.

Т Е М А	Кількість годин									
	Денна форма									
	Всього	У тому числі								
		л	пр	інд	сам					
1	2	3	4	5	6					
Модуль 1										
Тема 1. Введення до предмету.			2		16					
Тема 2. Механізми керування процесами.			2		16					
Тема 3. Взаємодія між процесами ОС Unix.			2		16					
Тема 4. Потоки ОС Unix стандарту POSIX.			2		16					
Модуль 2										
Тема 5. Технологія MPI.			2		16					
Тема 6. Технологія OpenMP.			2		16					
Тема 7. Моделювання та аналіз параллельних обчислень.			2		10					
Всього	**120**	**0**	**14**		**106**					

6. Теми практичних занять

№ з/п	Назва теми	Кількість годин
Модуль 1		
1	Створення паралельних програм з використанням каналів (FIFO) ОС Unix.	2
2	Створення паралельних програм з використанням механізмів IPC ОС Unix.	2
3	Створення потоків стандарту POSIX.	2
4	Використання семафорів, мютексів та умовних змінних для синхронізації потоків стандарту POSIX.	2
Модуль 2		
5	Створення паралельних програм з використанням потоків стандарту POSIX.	2
6	Створення паралельних програм за технологією MPI.	2
7	Використання розподілених операцій стандарту MPI.	2
	Всього	**14**

Завдання до практичної роботи на тему «Технологія MPI»

1. Дано матрицю дійсних чисел $m \times n$. Кожний із процесів містить k її рядків. Написати процедуру множення цієї матриці на довільний вектор. Використати колективні комунікації.
2. Дано матрицю дійсних чисел $m \times n$. Кожний із процесів містить k її стовпців. Написати процедуру множення цієї матриці на довільний вектор. Використати колективні комунікації.
3. Дано матрицю дійсних чисел $m \times n$. Кожний із процесів містить k її рядків. Написати процедуру множення цієї матриці на довільний вектор. Використати колективні функції.
4. Дано матрицю дійсних чисел $m \times n$. Кожний із процесів містить k її стовпців. Написати процедуру множення цієї матриці на довільний вектор. Використати розподілені операції.
5. Створити розподілену операцію для знаходження максимального за модулем комплексного числа. Кожен паралельний процес читає зі свого файлу масив комплексних чисел. Розміри всіх масивів однакові. Знайти найбільше за модулем комплексне число із цих масивів.
6. Поле клітинного автомата (КА) являє собою двовимірний масив кліток. Кожна внутрішня клітка $C_{i,j}$ (у випадку періодичного КА й гранична) має 8 сусідів: $C_{i-1,j-1}$, $C_{i-1,j}$, $C_{i-1,j+1}$, $C_{i,j-1}$, $C_{i,j+1}$, $C_{i+1,j-1}$, $C_{i+1,j}$, $C_{i+1,j+1}$. Клітка КА може перебуває в одному із двох станів: клітка вільна (0), клітка зайнята (1). Робота КА визначається наступними правилами:
   - Якщо в зайнятий клітки більше трьох сусідніх кліток зайняті, клітка звільняється.
   - Якщо у вільної клітки хоча б дві сусідні клітки зайняті, клітка стає зайнятою.

   Для моделювання роботи КА використати наступну топологію процесів:
   a. Двовимірний масив процесорів.
   b. Двовимірний масив процесорів, періодичний по одній координаті.
   c. Двовимірний масив процесорів, періодичний по обох координатах.
   d. Кільце із процесорів.
7. Цілочисловий масив розподілений між 2^n паралельними процесами. Процеси утворять топологію гіперкуб. Спочатку всі процеси сортують свої частини масиву. Потім деякі з них передають відсортовані частини іншим процесам і завершують роботу. Процес, що одержав відсортований масив, поєднує його зі своїм локальним масивом, зберігаючи впорядкованість. Наприкінці роботи який-небудь один процес одержує повністю відсортований масив.
8. Для обчислення визначеного інтеграла від заданої функції по формулі середніх прямокутників використати n паралельних процесів.
9. Дійсна матриця A розміру $n \times n$ розподілена по стовпцях між p $(n = rp)$ паралельними процесами. Знайти QR-розкладання матриці A, використовуючи обертання Якобі.
10. Матриця розмірності $m \times n$ $(m < n)$ розподілена по рядках між p паралельними процесами ($m = rp$, r – ціле). Кожен процес зчитує свою частину матриці зі свого файлу.
    a. Знайти найбільший елемент у матриці, використовуючи розподілені операції.
    b. Знайти номери нульових стовпців матриці.
    c. Знайти найбільший елемент у кожному стовпці матриці.
    d. Матриця комплексна. Створити розподілену операцію знаходження максимального за модулем комплексного числа. Використовуючи створену розподілену операцію, знайти максимальне за модулем комплексне число матриці.

Завдання до практичної роботи на тему «Взаємодія між процесами ОС Unix»

1. Створити клас, що дозволяє використати «множинні семафори». Використовуючи цей клас, написати наступний програмний додаток мовою C++. Файли `f1,f2,f3,f4,f5` містять цілі числа. П'ять однакових процесів обробляють ці файли. Кожному процесу для роботи необхідно два файли. Результат роботи процес записує у файл `out_i`.

Процес	Файл 1	Файл 2
1	5	1
2	1	2
3	2	3
4	3	4
5	4	5

Процес виконує одне з наступних дій (визначається варіантом завдання):

a. Знаходить у файлах будь-яку пару парних чисел (по одному числу у файлі), заміняє їхніми нулями. Знайдені числа виводить у файл `out_i`. Якщо хоча б в одному файлі потрібних чисел немає - закінчує роботу.

b. Знаходить у файлах будь-яку пару непарних чисел (по одному числу у файлі), заміняє їхніми нулями. Знайдені числа виводить у файл `out_i`. Якщо хоча б в одному файлі потрібних чисел немає - закінчує роботу.

c. Знаходить у файлах будь-яку пару натуральних чисел (по одному числу у файлі), заміняє їхніми нулями. Знайдені числа виводить у файл `out_i`. Якщо хоча б в одному файлі потрібних чисел немає - закінчує роботу.

d. Знаходить у файлах будь-яку пару простих чисел (по одному числу у файлі), заміняє їхніми нулями. Знайдені числа виводить у файл `out_i`. Якщо хоча б в одному файлі потрібних чисел немає - закінчує роботу.

2. Для рішення задачі спроектувати монітор Хоара. Описати алгоритм рішення задачі з використанням цього монітора. Написати програму мовою С++, що реалізує цей алгоритм (монітор реалізувати з використанням класу семафора).

   a) Процеси `PW`, `PR1` й `PR2` спільно обробляють файл `f.pub`. `PW` читає з файлу `f.in` довільні натуральні числа й записує їх у файл `f.pub`. `PR1` читає з файлу `f.pub` парні числа й записує у файл `f1.out` їхні прості дільники. `PR2` читає з файлу `f.pub` непарні числа, перевіряє їх на простоту. Результат записує у файл `f2.out`. `PW` пише числа у файл, якщо файл порожній або число прочитано. Процеси `PR1`, `PR2` читають числа по одному разі.

   b) Процеси `PW`, `PR1` й `PR2` взаємодіють через загальну пам'ять. `PW` читає з файлу `f.in` і записує загальну пам'ять довільні натуральні числа. `PR1` читає із загальної пам'яті парні числа й записує у файл `f1.out` їхні прості дільники. `PR2` читає із загальної пам'яті непарні числа, перевіряє їх на простоту. Результат записує у файл `f2.out`. `PW` пише числа в загальну пам'ять, якщо загальна пам'ять порожня або число прочитане. Процеси `PR1`, `PR2` читають числа по одному разі.

3. Процес `P1` читає з файлу `f1.in` цілі числа й записує їх у файл `f.pub`. Процес `P2` читає числа з файлу `f.pub` і записує їхні прості дільники у файл `f2.out`. Процеси звертаються до файлу `f.pub` по черзі.

4. Процес `P1` читає з файлу `f1.in` цілі числа й записує їх у загальну пам'ять. Процес `P2` читає числа із загальної пам'яті й записує їхні прості дільники у файл `f2.out`. Процеси звертаються до загальної пам'яті по черзі.

5. Процес `P1` читає з файлу `f1.in` рядок символів, шифрує її й записує у файл `f.pub`. Процес `P2` читає зашифрований рядок з файлу `f.pub`, розшифровує її й записує у файл `f2.out`. Процеси звертаються до файлу `f.pub` по черзі. Рядки шифруються алгоритмом Цезаря.

6. Процес `P1` читає з файлу `f1.in` рядок символів, шифрує її й записує в загальну пам'ять. Процес `P2` читає зашифрований рядок із загальної пам'яті, розшифровує її й записує у файл `f2.out`. Процеси звертаються до загальної пам'яті по черзі. Рядки шифруються алгоритмом Цезаря.

7. Описати клас «рахункові семафори» і вирішити наступну задачу («читачі-письменники»). Процес-письменник `PW` читає з файлу `f1.in` цілі числа й записує у файл `f.pub` (якщо лічено 0 – додаток завершує роботу). Процеси-читачі виконують наступне: `PR1` читає з файлу `f.pub` парні числа й знаходить його найбільший простий дільник. Результат записує у файл `f1.out`. `PR2` читає з файлу `f.pub` непарні числа, перевіряє їх на простоту й

результат виводить у файл `f2.out`. `PR3` читає з файлу `f.pub` негативні числа й виводить у файл `f3.out`.

8. Додаток-сервер створює черга повідомлень. Потім, читаючи із черги повідомлення записує у файл `f1.out` повідомлення типу 1, а у файл `f2.out` – повідомлення типу 2. Одержавши повідомлення типу 3, додаток-сервер видаляє чергу й завершує роботу. Додаток-клієнт читає з файлу `f1.in` числа (тип повідомлення) і рядка символів і записує цю інформацію в чергу повідомлень. Продемонструвати роботу додатка-сервера й мінімум двох клієнтів, що працюють одночасно.
9. Створити конвеєр із трьох процесів `P1`, `P2`, `P3`. Процес `P1` читає з файлу `f1.in` рядок з латинських символів і пробілів і заміняє в ній всі малі літери прописними. Процес `P2` видаляє з рядка зайві пробіли. Процес `P3` заміняє пробіли символом `'_'` і виводить рядок у файл `f1.out`. Використати семафори Unix System Y, описавши на їхній основі клас семафорів. Для взаємодії між собою процеси використають FIFO.
10. Створити конвеєр із трьох процесів `P1`, `P2`, `P3`. Процес `P1` читає з файлу `f1.in` рядок з латинських символів і пробілів і заміняє в ній всі малі літери прописними. Процес `P2` видаляє з рядка зайві пробіли. Процес `P3` заміняє пробіли символом `'_'` і виводить рядок у файл `f1.out`. Використати семафори Unix System Y, описавши на їхній основі клас семафорів. Для взаємодії між собою процеси використають загальну пам'ять.
11. Створити конвеєр із трьох процесів `P1`, `P2`, `P3`. Процес `P1` читає з файлу `f1.in` рядок з латинських символів і пробілів і заміняє в ній всі малі літери прописними. Процес `P2` видаляє з рядка зайві пробіли. Процес `P3` заміняє пробіли символом `'_'` і виводить рядок у файл `f1.out`. Використати семафори Unix System Y, описавши на їхній основі клас семафорів. Для взаємодії між собою процеси використають чергу повідомлень.
12. Створити конвеєр із трьох родинних процесів. Перший процес читає з файлу `f1.in` рядок з латинських символів і пробілів і заміняє в ній всі малі літери прописними. Другий процес видаляє з рядка зайві пробіли. Третій заміняє пробіли символом '_' і виводить рядок у файл `f.out`. Використати семафори Unix System Y, описавши на їхній основі клас семафорів. Для взаємодії між собою процеси використають анонімні канали.

Завдання до практичної роботи на тему «Потоки ОС Unix стандарту POSIX»

А. Додаток **клієнт** посилає додатку **сервер** запит, використовуючи один з механізмів межпроцессного взаємодії. Головний потік додатка **сервер** приймає запити від клієнта (клієнтів) і поміщає їх у чергу запитів. Деяка кількість потоків сервера одночасно обробляють ці запити, витягаючи їх із черги. Сервер завершує роботу, одержавши запит певного зразка. Результати обробки запитів потоки поміщають у файл. Для синхронізації роботи потоків використати мютексы й POSIX-семафори.

Вар.	IPC	Запит
1	FIFO	Розкласти натуральне число на прості множники
2	Черга повідомлень	Знайти n-і простої число
3	FIFO	Зашифрувати рядок шифром Цезаря
4	Черга повідомлень	Перевірити натуральне число на простоту
5	FIFO	Зашифрувати рядок шифром Гронсфельда.

Б. Додаток **клієнт** посилає додатку **сервер** запити двох типів, використовуючи один з механізмів межпроцессного взаємодії. Клієнт читає запити з файлу. Головний потік **сервера** приймає запити від клієнта (клієнтів) і поміщає їх у чергу запитів. Деяка кількість потоків (двох типів) сервера одночасно обробляють ці запити, витягаючи їх із черги. Сервер завершує роботу, одержавши запит певного зразка. Результати обробки запитів потоки поміщають у різні файли залежно від типу запиту. Для синхронізації роботи потоків використати мютексы, POSIX-семафори й умовні змінні.

Вар.	IPC	Запит 1	Запит 2
1	FIFO	Розкласти натуральне число на прості множники	Знайти n-і простої число
2	Черга повідомлень	Знайти n-і простої число	Зашифрувати рядок шифром Цезаря

3	FIFO	Зашифрувати рядок шифром Цезаря	Знайти n-і простої число
4	Черга повідомлень	Перевірити натуральне число на простоту	Зашифрувати рядок шифром Гронсфельда.
5	FIFO	Зашифрувати рядок шифром Гронсфельда.	Знайти n-і простої число

В. Головний потік програми формує черга завдань із вхідного файлу. Для обробки завдань із цієї черги використається конвеєр із трьох потоків. Перший потік витягає завдання із черги, обробляє й передає другому. Другий потік обробляє завдання й передає третьому. Третій потік обробляє завдання й виводить результат у файл. Потоки виконуються одночасно. Для синхронізації роботи потоків використати мютекси й POSIX-семафори.

вар	Потік 1	Потік 2	Потік 3
1	Інвертує рядок	Забирає пробіли	Інвертує рядок
2	Забирає пробіли в рядку	Шифрує шифром Цезаря	Розшифровує
3	Видаляє цифри з рядка	Забирає пробіли	Інвертує рядок
4	Видаляє цифри з рядка	Шифрує шифром Цезаря	Розшифровує
5	Забирає пробіли	Шифрує шифром Цезаря	Розшифровує

Вимоги до звіту: 1. Постановка задачі. 2. Алгоритм рішення задачі. 3. Текст програми. 4. Тестовий приклад. 5. Результати роботи програми на тестовому прикладі.

Методи навчання.

Підчас проведення лекцій використовуються пасивний та активний методи навчання. На практичних заняттях використовуються активний та інтерактивний методи навчання. Під час виконання студентами лабораторних робіт використовується активний методи навчання.

Методи контролю.

На практичних заняттях проводиться усне опитування студентів. За результатами виконання лабораторних та індивідуальних завдань студенти оформляють звіт. Звіт перевіряється викладачем в присутності студента в формі співбесіди.

За результатами усного опитування та співбесід викладач оцінює знання студента по кожній темі.

7. Розподіл балів, які отримують студенти.

Екзамен за 8-й семестр

Поточне тестування та самостійна робота							**Підсумковий тест (залік)**	Сума
Модуль 1				Модуль 2				
Т1	Т2	Т3	Т4	Т5	Т6	Т7	25	100
2	3	15	15	15	15	10		

8. Рекомендована література

Базова

1. Богачёв К.Ю. Основы параллельного программирования. – М.: Бином, 2003. – 342с.
2. Митчелл М., Оулдем Дж., Самьюэл А. Программирование для Linux. Профессиональный подход. – М: Издательский дом “Вильямс”, 2002.
3. Робачевский А. Операционная система Unix. – СПб: БХВ-Петербург, 1999.
4. Грегори Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования. – М.: Издательский дом «Вильямс», 2003. – 512 с.

Інформаційні ресурси

http://www.parallel.ru – сервер Лаборатории Параллельных информационных технологий Научно-исследовательского вычислительного центра Московского государственного университета имени М.В.Ломоносова.

http://www.mcs.anl.gov – Argonne National Laboratory, Center for Computational Science and Technology.

http://www.cacr.caltech.edu/ – CACR - Центр Вычислительных Исследований (Center of Advanced Computing Research)
http://netlib.org – Netlib is a collection of mathematical software, papers, and databases.