КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Кафедра теорії та технології програмування

«ЗАТВЕРДЖУЮ»
Заступник декана
з навчальної роботи

_____ Кашпур О.Ф.

«____»_____2017 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ[1]
ОПЕРАЦІЙНІ СИСТЕМИ
для студентів денної форми навчання

галузь знань **11 «Математика та статистика»**
(шифр і назва)
спеціальність **113 «Прикладна математика»**
(шифр і назва спеціальності)
освітній рівень **магістр**
(молодший бакалавр, бакалавр, магістр)
освітня програма **«**Прикладна математика»
(назва освітньої програми)
Спеціалізація:
- Обчислювальна математика;

вид дисципліни **обов'язкова**

Форма навчання	**денна**
Навчальний рік	**2017/2018**
Семестр	**2**
Кількість кредитів ECTS	**2**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**екзамен**

Викладачі: **к.ф.-м.н., доц.Волохов В.М.** (лекції, лабораторні заняття),

Пролонговано: на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

[1] Робоча програма навчальної дисципліни є нормативним документом вищого навчального закладу і містить виклад конкретного змісту навчальної дисципліни, послідовність, організаційні форми її вивчення та їх обсяг, визначає форми та засоби поточного і підсумкового контролів.

КИЇВ – 2017

Розробник: Волохов Віктор Миколайович, к.ф.-м.н., доцент кафедри «Теорії та технології програмування»

ЗАТВЕРДЖЕНО
В.О. Зав. кафедри «Теорії та технології програмування»

_____ (Панченко Т.В.)
(підпис) (прізвище та ініціали)

Протокол № ___ від «____» _____ 20___ р.

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики _____

Протокол від «____» _____ 20___ року №___

Голова науково-методичної комісії _____ (Хусаїнов Д.Я.)
(підпис) (прізвище та ініціали)

«_____» _____ 20___ року

Вступ. Навчальна дисципліна "Операційні системи" є складовою освітньо-професійної програми підготовки фахівців за освітньо-кваліфікаційним рівнем «магістр» галузі знань *"*Системні науки та кібернетика*"* з *напряму підготовки* 8.04030101 „*Прикладна математика* ".

Дана дисципліна за блоком дисциплін, ***спеціальністю "Прикладна математика".***

Викладається у **2** семестрі першого року навчання магістрів **в обсязі – 72 год.**

(2 кредити ECTS[1]) зокрема: *лекції – 34 год., самостійна робота – 38 год.* У курсі передбачено **2** *змістових модулі* та **2** *модульних контрольних роботи*. Завершується дисципліна – **іспитом у 5 семестрі.**

Метою дисципліни „Операційні системи" є опанування теорії, методик та отримання досвіду з проектування та програмування операційних систем, включаючи набуття навичок об'єктно-орієнтованого програмування та оволодіння мовами програмування Java та C#.

Предмет навчальної дисципліни „Операційні системи" включає в себе розгляд теоретичних аспектів проектування та створення операційних систем, вивчення компонент операційних систем, опанування алгоритмів та їх програмування з метою подальшого проектування та програмування операційних систем.

Завдання: набуття компетенцій, знань, умінь та навиків на рівні новітніх досягнень у розробці операційних систем, відповідно до кваліфікації фахівець з інформаційних технологій.

В результаті вивчення навчальної дисципліни студент повинен:

знати: основи теорії побудови операційних систем: основні алгоритми управляння ресурсами операційної системи, методи розробки компонент операційної системи; принципи проектування ядра операційної системи; інструментальні засоби мови програмування; передові технології; мови програмування Java;

вміти: розробляти та реалізувати основні алгоритми управляння ресурсами операційної системи роз'яснювати і представляти проекти / розробки замовникам з використанням сучасних технологій розробки програмних систем;.

Місце дисципліни. Дисципліна за блоком дисциплін „Системне програмування" є складовою циклу професійної підготовки фахівців освітньо-кваліфікаційного рівня „бакалавр".

Зв'язок з іншими дисциплінами. Навчальна дисципліна „Операційні системи" є факультативною для вивчення в подальшому таких спеціальних дисциплін як „Математична графіка", та „Штучний інтелект".

Дисципліна за блоком "Операційні системи" є складовою циклу професійної підготовки фахівців освітньо-кваліфікаційного рівня "бакалавр", є факультативною для засвоєння всіх інших курсів та спецкурсів програміського спрямування, окремих розділів теорії алгоритмів та математичної логіки, теорії та технології баз даних

[1] кредитів ECTS – кредит кратний 36 годинам (Наприклад, 3 кредити ECTS відповідає 108 год.).

Контроль знань і розподіл балів, які отримують студенти.

Контроль здійснюється за модульно-рейтинговою системою.

У змістовий модуль 1 (ЗМ1) входять теми 1- 8, у змістовий модуль 2 (ЗМ2) – теми 9 - 17. Обов'язковим для заліку у 1 семестрі є отримання студентом-магістром протягом 1 семестру не менше 21 балу.

Оцінювання за формами контролю[2]: (як приклад)

	ЗМ1		***ЗМ2***	
	Min. – 10 балів	*Max. – 30 бали*	*Min. – **11** балів*	*Max. – **30** балів*
Модульна контрольна робота	5	10	5	10
Лабораторна робота	5	15	6	15
Активна робота	0	5	0	5
„3" – мінімальна/максимальна оцінку, яку може отримати студент. [1] – мінімальна/максимальна залікова кількість робіт чи завдань.				

Для студентів, які набрали сумарно меншу кількість балів ніж *критично-розрахунковий мінімум – **21** бал* для допуску до заліку обов'язково повинні перескласти контрольні роботи та здати заплановані лабораторні роботи. Студент має право на одне перескладання контрольної роботи із можливістю отримання максимально 5 балів за кожну. Термін перескладання визначається викладачем.

Оцінювання за формами контролю: (як приклад)

	ЗМ1		***ЗМ2***	
	Min. – 10 балів	*Max. –30 бали*	*Min. – **11** балів*	*Max. – **30** балів*
Модульна контрольна робота	5	10	5	10
Лабораторна робота	5	15	6	15
Активна робота	0	5	0	5
„3" – мінімальна/максимальна оцінку, яку може отримати студент. [1] – мінімальна/максимальна залікова кількість робіт чи завдань.				

Для студентів, які набрали сумарно меншу кількість балів ніж *критично-розрахунковий мінімум – **21** балів* для допуску до заліку/іспиту обов'язково повинні перескласти контрольні роботи та здати заплановані лабораторні роботи. Студент має право на одне перескладання контрольної роботи із можливістю отримання максимально 5 балів за кожну. Термін перескладання визначається викладачем.

У випадку відсутності студента з поважних причин відпрацювання та перездачі МКР здійснюються у відповідності до „Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу" від 1 жовтня 2010 року.

При простому розрахунку отримаємо:

	Змістовий модуль1	Змістовий модуль2	іспит	Підсумкова оцінка
Мінімум	15	14	31	60
Максимум	**30**	**30**	**40**	**100**

[2] Див. Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу від 1 жовтня 2010 року, а також Розпорядження ректора «Про методику розрахунку підсумкової оцінки дисциплін, які читаються два і більше семестри» від 29 вересня 2010 року

Контроль знань студентів здійснюється за модульно-рейтинговою системою. Результати навчальної діяльності студентів оцінюються за 100-бальною шкалою. Робота в семестрі поділяється на два змістових модуля. При виставленні балів за змістовий модуль враховується: оцінка за модульну контрольну роботу - 10 балів, виконання студентом модульних лабораторних робіт – 10 балів/одна робота, робота на самостійній роботі - 5 балів. Підсумковий контроль проводиться у формі письмового заліку – 30 балів. Підсумкова оцінка 100=2*(10+2*10+5)+30.

Якщо студент з поважних причин, які підтверджено документально, був відсутній при написанні модульної контрольної роботи, він має право на одне перескладання з можливістю отримання максимальної кількості балів. Термін перескладання визначається викладачем.

За активну роботу студента на практичних заняттях та на самостійній роботі до семестрової оцінки може бути додано до 5 балів.

Якщо впродовж семестру студент не з'являвся на заняття (не залежно від причин), не має модульних оцінок, у відповідних графах „Відомості обліку успішності КМСОНП" виставляються „0", а у графі іспиту – відмітка про недопуск.

Студент допускається до складання заліку, якщо кількість набраних ним балів за семестр становить не менше 30 балів.

Залік вважається не зданим, якщо сумарна кількість балів з дисципліни у семестрі складає менше 60 балів.

При цьому, кількість балів:

- **1-34** відповідає оцінці «незадовільно» з обов'язковим повторним вивченням дисципліни;
- **35-59** відповідає оцінці «незадовільно» з можливістю повторного складання;
- **60-64** відповідає оцінці «задовільно» («достатньо»);
  - відповідає оцінці «задовільно»;
- **75 - 84** відповідає оцінці «добре»;
- **85 - 89** відповідає оцінці «добре» («дуже добре»);
- **90 - 100** відповідає оцінці «відмінно».

Шкала відповідності *(за умови іспиту)*

За 100 – бальною шкалою	За національною шкалою	
90 – 100	5	відмінно
85 – 89	4	добре
75 – 84		
65 – 74	3	задовільно
60 – 64		
35 – 59	2	незадовільно
1 – 34		

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Змістовий модуль 1. Загальна теорія операційних систем.

Тема 1. Покоління архітектур обчислювальних машин. Структура сучасної ОС. Програмні та мікропрограмі рівні.. – **4 год.**

Тема 2. Поняття процесу та ресурсу. Типи ресурсів. Структура процесу. Потоки. Структура потоку, відображення засобами ядра ОС.. Приклади .– **4 год.**

Тема 3. Управління ресурсами. Умови взаємо виключення. Критичні сегменти. Алгоритм Деккера. Взаємовиключення на рівні апаратних засобів. Приклади .– **4 год.**

Тема 4. Семафори. P-, V- операції Дейкстра. Двійкові семафори. Задача "Постачальник-Споживач". Приклади .– **4 год.**

Тема 5. Монітори взаємодії. Операції Signal та Wait. Задача "Читачі-Письменники". Приклади .– **4 год.**

Тема 6. Тупики в середовищі ОС. Алгоритм Банкіра. Управління тупиковими ситуаціями. Приклади .– **4 год.**

Змістовий модуль 2. Управління ресурсами.

Тема 7. Планування ресурсу "Центральний процесор". Черги на центральний процесор. Контекст виконання задачі. Приклади .– **4 год.**

Тема 8. Задачі та потоки в середовищі операційної системи. Планування потоків у контексті задачі. Ресурси задачі та ресурси потоку. Взаємовиключення.. Приклади .– **2год.**

Тема 9. Планування ресурсу "Оперативна пам'ять". Однозадачний та мультизадачний режими ОС. Фрагментація оперативної пам'яті. Статичне та динамічне завантаження модулів. Завантаження модулів з перекриттям.. Приклади .– **4 год.**

Тема 10. Моделі віртуальної пам'яті Апаратні та програмні засоби управління віртуальною пам'яттю. Сегментна та сторінкова стратегії управління ОП. Приклади .– **4 год.**

Тема 11. Управляння віртуальною пам'яттю: сторінкова стратегія. Алгоритми підвантаження та вивантаження сторінок з оперативної пам'яті.. Приклади **.– 4 год.**

Тема 12. Сумісне використання віртуального адресного простору ядром ОС та задачею. Фіксація модулів задач в оперативній пам'яті. Задачі Real Time. Приклади .– **4 год.**

Тема 13. Поняття фізичного та логічного вводу-виводу. Об'єкти управління логічної системи вводу-виводу. Буферизований в/вивод. Приклади .– **4 год.**

Тема 14. Поняття програмного кеш для файлового вводу-виводу. Алгоритми управління програмним кеш.. Приклади .– **4 год.**

Тема 15. Оптимізація обміну даними з НЖМД. Стандарти на RAID-контролери. Рівні RAID. Приклади .– **4 год.**

Лекція 16. Операційні системи та безпека інформації. Напрями безпеки та рівні гарантій. Профілі захищеності. **. - 4 год.**

Тема 17. Доповіді та обговорення рефератів. **- 4 год.**

ТЕМАТИЧНИЙ ПЛАН ЛЕКЦІЙ І СЕМІНАРСЬКИХ ЗАНЯТЬ СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

№ лекції	Назва лекції (теми)	Лекції	Самост. робота
1	2	3	4
Змістовий модуль 1. **Загальна теорія операційних систем.**			
1	Покоління архітектур обчислювальних машин. Структура сучасної ОС. Програмні та мікропрограмі рівні.	2	2
2	Поняття процесу та ресурсу. Типи ресурсів. Структура процесу. Потоки. Структура потоку, відображення засобами ядра ОС.	2	2
3	Управління ресурсами. Умови взаємо виключення. Критичні сегменти. Алгоритм Деккера. Взаємовиключення на рівні апаратних засобів.	2	2
4	Семафори. P-, V-операції Дейкстра. Двійкові семафори. Задача "Постачальник-Споживач".	2	2
5	Монітори взаємодії. Операції Signal та Wait. Задача "Читачі-Письменники".	2	2
6	Тупики в середовищі ОС. Алгоритм Банкіра. Управління тупиковими ситуаціями.	2	2
Модульна контрольна робота 1		1	
Змістовий модуль 2. **Управління ресурсами.**			
7	Планування ресурсу "Центральний процесор". Черги на центральний процесор. Контекст виконання задачі.	2	2
8	Задачі та потоки в середовищі операційної системи. Планування потоків у контексті задачі. Ресурси задачі та ресурси потоку. Взаємовиключення.	2	2
9	Планування ресурсу "Оперативна пам'ять". Однозадачний та мультизадачний режими ОС. Фрагментація оперативної пам'яті. Статичне та динамічне завантаження модулів. Завантаження модулів з перекриттям.	2	2
10	Моделі віртуальної пам'яті Апаратні та програмні засоби управління віртуальною пам'яттю. Сегментна та сторінкова стратегії управління ОП.	2	2
11	Управляння віртуальною пам'яттю: сторінкова стратегія. Алгоритми підвантаження та вивантаження сторінок з оперативної пам'яті.	2	2
12	Сумісне використання віртуального адресного простору ядром ОС та задачею. Фіксація модулів задач в оперативній пам'яті. Задачі Real Time.	2	2
13	Поняття фізичного та логічного вводу-виводу. Об'єкти управління логічної системи вводу-виводу. Буферизований в/вивод.	2	2
14	Поняття програмного кеш для файлового вводу-виводу. Алгоритми управління програмним кеш.	2	2
15	Оптимізація обміну даними з НЖМД. Стандарти на RAID-	2	2

1	2	3	4
	контролери. Рівні RAID.		
16	Операційні системи та безпека інформації. Напрями безпеки та рівні гарантій. Профілі захищеності.	2	4
17	Доповіді та обговорення рефератів.	2	4
	Модульна контрольна робота 2	1	

Загальний обсяг 72 годин, в тому числі:

- Лекцій – 34 год.,
- Самостійна робота – 38 год.

Змістовий модуль 1. Загальна теорія операційних систем.

Лекція1. Покоління архітектур обчислювальних машин. Структура сучасної ОС. Програмні та мікропрограмі рівні.. – **2 год.**

Покоління архітектур ЕОМ. Перші операційні системи. Однозадачні та мультизадачні операційні системи. Інтеграція апаратних та програмних засобів. Програмний та мікропрограмний рівні управління. Структура сучасної операційної системи. [1], [2], [3].

Завдання для самостійної роботи. Ознайомлення з історією створення операційних систем. Операційні системи від компанії IBM. **– 2 год**. [1], [2], [3].

Лекція 2. Поняття процесу та ресурсу. Типи ресурсів. Структура процесу. Потоки. Структура потоку, відображення засобами ядра ОС.. Приклади .– **2год.**

Означення процесу. Трактування процесу на рівні ОС та прикладної задачі. Типи процесів. Кругообіг процесів відносно CPU. Структура процесу. Дескриптор процесу та потоку. Контекст задачі та контекст потоку. Зміна контексту. [1], [2], [3].

Завдання для самостійної роботи. Процеси та потоки у середовищі сучасних операційних системи MS Windows, SCO Unix.**– 2 год.** [1], [2], [3].

Лекція 3. Управління ресурсами. Умови взаємо виключення. Критичні сегменти. Алгоритм Деккера. Взаємовиключення на рівні апаратних засобів. Приклади .– **2год.**

Поняття ресурсу. Типи ресурсів. Умови взаємовиключення. Поняття критичного сегменту у програмі. Перші алгоритми взаємовиключення на рівні програмного коду – Алгоритм Деккура. Взаємовиключення на апаратному рівні. Приклади. [1], [2], [3].

Завдання для самостійної роботи. Інтерфейси операційних системи MS Windows, SCO Unix для реалізації взаємовиключення. **– 2 год.** [1], [2], [3].

Лекція 4. Семафори. P-, V- операції Дейкстра. Двійкові семафори. Задача "Постачальник-Споживач". Приклади .– **2год.**

Означення семафора. Три базових методи роботи з семафорами. Семафори Дейкста. Двійкові семафори. Приклади використання. Постановка задачі „Постачальник - Споживач". Структури даних. Алгоритм дії Постачальника. Алгоритм дії Споживача. Реалізація. [1], [2], [3].

Завдання для самостійної роботи. Інтерфейси операційних системи MS Windows, SCO Unix для реалізації взаємовиключення. **– 2 год.** [1], [2], [3].

Лекція 5. Монітори взаємодії. Операції Signal та Wait. Задача "Читачі-Письменники". Приклади .– **2год.**

Означення монітора. Умови виконання методу (функції) в моніторі. Операції Signal та Wait. Постановка задачі „Читачі-Письменники". Структури даних. Алгоритми дії Читача. Алгоритм дії Письменника. Реалізація. [1], [2], [3].

Завдання для самостійної роботи. Задача „Філософи за столом”. Структури даних. Алгоритми реалізації засобами ядра MS Windows**. - 2 год.** [1], [2], [3].

Лекція 6. Тупики в середовищі ОС. Алгоритм Банкіра. Управління тупиковими ситуаціями. Приклади .– **2год.**

Означення тупика. Умови виникнення тупика. Алгоритми передбачення (прогнозування) тупиків. Алгоритми боротьби з тупиками. Алгоритм банкіра. Наслідки. Боротьба з тупиками на рівні операційної системи та на рівні адміністратора системи: алгоритми, засоби, прийоми. [1], [2], [3].

Завдання для самостійної роботи. Задача „Філософи за столом”. Структури даних. Алгоритми реалізації засобами ядра MS Windows. Реалізація**. - 2 год.** [1], [2], [3].

Контрольні запитання до змістового модуля 1.

1. Структура сучасної операційної системи.
2. Кругообіг задач щодо центрального процесора. Переходи з черги в чергу.
3. Означення семафора. Методи семафора.
4. Означення монітора ресурсу. Умови виконання методів у моніторі.
5. Означення тупика. Умови виникнення тупика. Алгоритм банкіра.

Типове завдання модульної контрольної роботи №1.

1. Кругообіг задач щодо центрального процесора. Переходи з черги в чергу.
2. Задача: Дано три процеси: Постачальник, Обробник, Друкар. Організуйте взаємодію трьох процесid через оперативну пам’ять. Скористайтеся механізмом семафорів.

Змістовий модуль 2. Управління ресурсами.

Лекція 7. Планування ресурсу “Центральний процесор”. Черги на центральний процесор. Контекст виконання задачі. Приклади .– **2год.**

Планування ресурсу центрального процесора через наступні механізми: апарат преривань, квантування часу, пріоритет, динамічний пріоритет. Планування центрального процесора для задач та потоків. Багаторівневі стратегії планування. Ресурси часу для зміни контексту. [1], [4].

Завдання для самостійної роботи. Задача „Філософи за столом”. Структури даних. Алгоритми реалізації засобами ядра MS Windows. Реалізація через механізм задач та механізм потоків**. - 2 год**. [1], [4].

Лекція 8. Задачі та потоки в середовищі операційної системи. Планування потоків у контексті задачі. Ресурси задачі та ресурси потоку. Взаємовиключення.. Приклади .– **2год.**

Планування ресурсу центрального процесора для потоків задач. Організація взаємовиключення. Спільні: оперативна пам’ять, файли та інші ресурси. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 9. Планування ресурсу “Оперативна пам’ять”. Однозадачний та мультизадачний режими ОС. Фрагментація оперативної пам’яті. Статичне та динамічне завантаження модулів. Завантаження модулів з перекриттям.. Приклади .– **2 год.**

Планування оперативної пам’яті в режимі однозначної операційної системи. Структура Дескриптора вільної пам’яті. Планівання по стратегії Стек та по стратегії Куча. Планування ресурсу центрального процесора для потоків задач. Організація взаємовиключення. Спільні: оперативна пам’ять, файли та інші ресурси. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 10. Моделі віртуальної пам'яті Апаратні та програмні засоби управління віртуальною пам'яттю. Сегментна та сторінкова стратегії управління ОП. Приклади .– **2год.**

Поняття віртуальної пам'яті як адресного простору виконання задачі. Стратегії реалізації моделі віртуальної пам'яті. Сторінкова стратегія. Транстяція віртуального адреса в адрес оперативної пам'яті. Структура таблиці трансляції. Обробка апаратного преривання „Сторінка відсутня в ОП". Проблеми управління таблицею трансляції. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 11. Управляння віртуальною пам'яттю: сторінкова стратегія. Алгоритми підвантаження та вивантаження сторінок з оперативної пам'яті.. Приклади .– **2год.**

Алгоритми підвантаження сторінок в оперативну пам'ять. Алгоритми вивантаження сторінок з оперативної пам'яті. Поняття карти пам'яті. Статистика звернення до сторінок оперативної пам'яті. „Робоча" множина сторінок. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 12. Сумісне використання віртуального адресного простору ядром ОС та задачею. Фіксація модулів задач в оперативній пам'яті. Задачі Real Time. Приклади .– **2год.**

Структура ядра в оперативній пам'яті. Вектор програмних преривань. Обробники програмних преривань. Опис ресурса Задача в ядрі ОС. Поняття фіксованих сторінок як в ядрі ОС, так і в прикладній програмі. Задачі Real Time. Ресурси задачі. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 13. Поняття фізичного та логічного вводу-виводу. Об'єкти управління логічної системи вводу-виводу. Буферизований в/вивод. Приклади .– **2 год.**

Поняття логічного та фізичного рівнів вводу – виводу. Об'єкти логічного управління. Буферизований ввод – вивод. Планування оперативної пам'яті під операції вводу-виводу. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 14. Поняття програмного кеш для файлового вводу-виводу. Алгоритми управління програмним кеш.. Приклади .– **2 год.**

Поняття програмного кеш. Структура програмного кеш. Атрибути блоків програмного кеш. Алгоритми ефективного управління програмним кеш. Програмний кеш та СУБД. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 15. Оптимізація обміну даними з НЖМД. Стандарти на RAID-контролери. Рівні RAID. Приклади .– **2 год.**

Критерії оптимізації доступу до НЖМД. Апаратний кеш НЖМД. Стандарти на RAID-контролери. Рівні RAID. Алгоритми RAID. Приклади. [1], [4].

Завдання для самостійної роботи. Робота над рефератом**. - 2 год.** [1], [4].

Лекція 16. Операційні системи та безпека інформації. Напрями безпеки та рівні гарантій. Профілі захищеності. **. - 2 год.**

Системи класу А та Б. Вимоги до систем безпеки. Приклади. Термінологія в області захисту інформації. Напрямки безпеки інформації. Рівні безпеки інформації. Профілі безпеки інформації. Приклади. [1], [4].

Завдання для самостійної роботи. Ознайомлення з термінологією згідно державних стандартів безпеки інформації України на сайті www.dstszi.gov.ua**. - 4 год.** [1], [4].

Лекція 17. Доповіді та обговорення рефератів. **- 2 год.**

Завдання для самостійної роботи. Ознайомлення з термінологією згідно державних стандартів безпеки інформації України на сайті www.dstszi.gov.ua**. - 4 год.** [1], [4].

Контрольні запитання до змістового модуля 2.

1. Алгоритми планування ресурсу CPU
2. Контекст задачі. Зміна контексту. Потоки та контекст задачі.
3. Планування ресурсу оперативної пам'яті в однозадачній ОС.
4. Планування ресурсу оперативної пам'яті в мультизадачній ОС.
5. Модель віртуальної пам'яті. Трансляція віртуального адреса в сторінковій стратегії.
6. Алгоритми вивантаження та підвантаження сторінок.
7. Стандарт RAID. RAID 0, 1, 2, 3.
8. Програмний кеш. Алгоритми ефективного використання програмного кеш.
9. Стандарти безпеки інформації.

Типове завдання модульної контрольної роботи №2.

1. Стандарти безпеки інформації.
2. Алгоритми вивантаження та підвантаження сторінок.
3. Контекст задачі. Зміна контексту. Потоки та контекст задачі.

Перелік питань на залік.

1. Структура сучасної операційної системи.
2. Кругообіг задач щодо центрального процесора. Переходи з черги в чергу.
3. Означення семафора. Методи семафора.
4. Означення монітора ресурсу. Умови виконання методів у моніторі.
5. Означення тупика. Умови виникнення тупика. Алгоритм банкіра.
6. . Алгоритми планування ресурсу CPU
7. Контекст задачі. Зміна контексту. Потоки та контекст задачі.
8. Планування ресурсу оперативної пам'яті в однозадачній ОС.
9. Планування ресурсу оперативної пам'яті в мультизадачній ОС.
10. Модель віртуальної пам'яті. Трансляція віртуального адреса в сторінковій стратегії.
11. Алгоритми вивантаження та підвантаження сторінок.
12. Стандарт RAID. RAID 0, 1, 2, 3.
13. Програмний кеш. Алгоритми ефективного використання програмного кеш.
14. Стандарти безпеки інформації.

ПЕРЕЛІК ТЕМ ДЛЯ НАПИСАННЯ РЕФЕРАТІВ

1. Операційна система Qubes. Технічна документація. Переклад на українську мову. Стор 1-15
2. Операційна система Qubes. Технічна документація. Переклад на українську мову. Стор 16-27.
3. Операційна система Qubes. Технічна документація. Переклад на українську мову. Стор 28-44
4. Операційна система Unix. Служби моніторингу.
5. Операційна система Unix. Аварійні ситуації. Підготовка до відновлення.
6. Операційна система Unix. Забезпечення високої надійності.
7. Операційна система Unix. Забезпечення безпеки системи.
8. Операційна система Solaris. Управління задачами. Потоки та легкі потоки.

9. Операційна система Unix. Набір стандартних утиліт (команд).
10. Операційна система Unix. Утиліти (команди) досвідченого програміста.
11. Операційна система Unix. Інтерпретатори: Bourn shell та C shall.
12. Операційна система Unix. Засоби організації мережі.
13. Операційна система Unix. Засоби рівня ядра ОС: процеси, файли, віртуальна пам'ять.
14. Операційні системи. Протокол TCP/IP. Маршрутизація, доменні імена тощо.
15. Операційна система UNIX: стандартна файлова система: структура дескриптора файла.
16. ОС Windows 2000. Файлова система NTFS, її характеристики.
17. Протокол TCP/IP. Налаштування в середовищі Windows-систем.
18. Протокол TCP/IP. Налаштування в середовищі Unix-систем.
19. Управління задачами та потоками в середовищі Windows та Unix-систем.
20. Безпека та захист інформації в середовищі Windows-систем.
21. Безпека та захист інформації в середовищі Unix-систем.
22. Міжнародний стандарт ISO: термінологія в області безпеки інформації.
23. Міжнародний стандарт ISO: напрямки безпеки та рівні гарантій.
24. Міжнародний стандарт ISO: профілі захищеності систем.

Основна література

1. В. Столлингс. Операционные системы. Внутреннее устройство и принципы проектирования. М. Вільямс. 2002.
2. Д. Цикритзис, Фє Бернстайн. Операционные системы. М. Мир. 1977.
3. Г. Дейл. Введение в операционные системы. Т. 1, 2. М. Мир. 1988.
4. К.Стинсон, К. Зихерт. Windows 200 Professional. Питер. 2003.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Кафедра теорії та технології програмування

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ (ІДИВІДУАЛЬНОЇ) РОБОТИ

у період січень-лютий 2018 року

ТЕОРЕТИЧНА ЧАСТИНА

№ лекції	Назва лекції (теми)	Лекції	Самост. робота
1	2	3	4
Змістовий модуль 1. **Загальна теорія операційних систем.**			
1	Покоління архітектур обчислювальних машин. Структура сучасної ОС. Програмні та мікропрограмі рівні.	2	2
2	Поняття процесу та ресурсу. Типи ресурсів. Структура процесу. Потоки. Структура потоку, відображення засобами ядра ОС.	2	2
3	Управління ресурсами. Умови взаємо виключення. Критичні сегменти. Алгоритм Деккера. Взаємовиключення на рівні апаратних засобів.	2	2
4	Семафори. P-, V-операції Дейкстра. Двійкові семафори. Задача "Постачальник-Споживач".	2	2
5	Монітори взаємодії. Операції Signal та Wait. Задача "Читачі-Письменники".	2	2
6	Тупики в середовищі ОС. Алгоритм Банкіра. Управління тупиковими ситуаціями.	2	2

Основна література

5. В. Столлингс. Операционные системы. Внутреннее устройство и принципы проектирования. М. Вільямс. 2002.
6. Д. Цикритзис, Фє Бернстайн. Операционные системы. М. Мир. 1977.
7. Г. Дейл. Введение в операционные системы. Т. 1, 2. М. Мир. 1988.

К.Стинсон, К. Зихерт. Windows 200 Professional. Питер. 2003.

Контрольні запитання до змістового модуля 1.

1. Структура сучасної операційної системи.
2. Кругообіг задач щодо центрального процесора. Переходи з черги в чергу.
3. Означення семафора. Методи семафора.
4. Означення монітора ресурсу. Умови виконання методів у моніторі.
5. Означення тупика. Умови виникнення тупика. Алгоритм банкіра.

Типове завдання модульної контрольної роботи №1.

1. Кругообіг задач щодо центрального процесора. Переходи з черги в чергу.
2. Задача: Дано три процеси: Постачальник, Обробник, Друкар. Організуйте взаємодію трьох процесів через оперативну пам'ять. Скористайтеся механізмом семафорів.

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Студент повинен виконати лабораторну роботу №1 з нижче наведеного переліку лабораторних робіт. У кінці переліку лабораторних робіт опублікована матриця «Студент-номер роботи та мова програмування»

ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

ЛАБОРАТОРНА РОБОТА 1

1. **Задача «про філософів».** За круглим столом сидять 8 філософів. Кожному філософу подали по тарілці спагеті. Біля кожного філософа зліва та справа поклали по одні виделці, тобто між сусідніми філософами лежить по одній виделці.

Філософи або розмовляють (виделки покладені на стіл), або їдять спагеті (звичайно двома виделками).

Реалізуйте філософів як потоки (під задачі). В стандартний вивод направте повідомлення філософів:

- Філософ xx їсть спагеті;
- Філософ xx очікує на вільні виделки;
- Філософ xx розмовляє.

Коректно реалізована задача не приведе до тупика, стану, коли всі філософи чекають на виделки (нових повідомлень не буде).

2. **Задача «про перукаря».** В перукарні працює 1 перукар (один стілець). В перукарні є кімната на uu стільців, де клієнти очікують на обслуговування. Якщо в кімнаті на очікування вся стільці зайняті, то новий клієнт стоїть на вулиці в черзі на очікування.

Реалізуйте взаємодію клієнтів та перукаря. Головний модуль створює потік клієнтів через випадкове значення часу. Перукар обслуговує клієнтів через фіксований квант часу.

Коректно реалізована задача не призведе до тупика. В стандартний вивод направте повідомлення:

- працюю над клієнтом;
- очікую нового клієнта;
- клієнт в кімнаті для очікування;
- клієнт очікує в черзі на вулиці.

3. **Задача «про перукарів».** В перукарні працює xx перукарів (xx стільців). В перукарні є кімната на uu стільців, де клієнти очікують на обслуговування. Якщо в кімнаті на очікування вся стільці зайняті, то новий клієнт в перукарню не заходить.

Реалізуйте взаємодію клієнтів та перукарів. Головний модуль створює xx перукарів та потік клієнтів через випадкове значення часу. Перукар обслуговує клієнтів через фіксований квант часу.

Коректно реалізована задача не призведе до тупика. В стандартний вивод направте повідомлення:

- перукар uu працює над клієнтом;
- перукар uu очікує нового клієнта;
- клієнт в кімнаті для очікування;
- кімната очікування переповнена, клієнт в перукарню не зайшов.

4. **Задача «Постачальник – обробник - споживач».** Реалізуйте задачу по обробці інформації через 1 буфер обміну, коли:

- перший потік (постачальник) наповнює буфер обміну даними;
- другий потік (обробник) перекодовує «великі» літери тексту на маленькі;
- третій потік (споживач) записує оброблені записи даних у вихідний файл.

Коректно реалізована задача не призведе до зміни тексту у вихідному файлі (по довжині, по значенню).

5. **Задача «Постачальник – обробник - споживач».** Реалізуйте задачу по обробці інформації через список з хх буферів обміну (наприклад, циклічний список), коли:

- перший потік (постачальник) наповнює буфери обміну даними;
- другий потік (обробник) перекодовує «великі» літери тексту на маленькі;
- третій потік (споживач) записує оброблені записи даних у вихідний файл.

Коректно реалізована задача не призведе до змани тексту у вихідному файлі (по довжині, по значенню).

6. **Задача: «Моделювання управління ОП».** Створити потік, який буде моделювати запити на виділення ОП та її утилізацію. Потік очікує запити решти N – потоків. При цьому кожний запит має формат (трійку): <ім’я потоку> <тип запиту> <об’єм в КБ>.

- <тип запиту>: 0 – на виділення ОП, -1 – на утилізацію.

Головний потік має обмеження на ресурс ОП. Промоделювати взаємодію потоків. Головний потік утилізує лише запити, що були надіслані І-м потоком

ЛАБОРАТОРНА РОБОТА 2

Розв’язки задач повинні бути запрограмовані мовою C/C++, Java, C# та містити повний комплект модулів (файлів, проектів тощо), а також тестових файлів – за потреби, для демонстрації всіх запитуваних можливостей.

Спільна умова для варіантів 1 – 21.

Програми (процеси, потоки – згідно варіанту задачі), що реалізують функції f(x) і g(x), займаються тільки обчисленням значення над вхідним аргументом, вони не обробляють ніяких інших запитів (у тому числі – про завершення обчислень) і не взаємодіють з іншими процесами та потоками ні в який інший спосіб, окрім викликів обчислень f(x) і g(x) (тобто запуску функції на обчислення) та повернення результату (коли обчислення результату завершено) – див. малюнок нижче.

Зауважити, що функції f та g – можуть бути частково визначені (тобто «зациклюватись» і ніколи не повертати результат). Потрібно *коректно опрацювати* таку ситуацію і *запитати користувача*: «продовжити обчислення, припинити або продовжити, не запитуючи більше» наприклад, кожні 10 секунд.

Увага! Скористатись правилами булевих обчислень:

x && false == false && x == false
та
x || true == true || x == true
де x ∈ {true, false}.
А також врахувати, що 0 * x == x * 0 == 0 для довільного числа x.

Варіанти.

1. **Взаємодія потоків. Паралелізм**. Обчислити f(x) * g(x), використовуючи 2 допоміжні потоки (threads): один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *.

2. **Взаємодія потоків. Паралелізм**. Обчислити f(x) && g(x), використовуючи 2 допоміжні потоки (threads): один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&.

3. **Взаємодія потоків. Паралелізм**. Обчислити f(x) || g(x), використовуючи 2 допоміжні потоки (threads): один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||.

4. **Взаємодія процесів. Паралелізм. Управління стандартним вводом-виводом**. Обчислити f(x) * g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Не використовувати обмін повідомленнями між процесами та порти. Процеси f та g читають дані з stdin, а результати пишуть в stdout, але не безпосередньо – вводом і виводом керує основна програма, вона посилає на вхід f і g дані та отримує від них результати. Забороняється використовувати допоміжні файли для обміну інформацією між процесами.

5. **Взаємодія процесів. Паралелізм. Управління стандартним вводом-виводом**. Обчислити f(x) && g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Не використовувати обмін повідомленнями між процесами та порти. Процеси f та g читають дані з stdin, а результати пишуть в stdout, але не безпосередньо – вводом і виводом керує основна програма, вона посилає на вхід f і g дані та отримує від них результати. Забороняється використовувати допоміжні файли для обміну інформацією між процесами.

6. **Взаємодія процесів. Паралелізм. Управління стандартним вводом-виводом**. Обчислити f(x) || g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Не використовувати обмін повідомленнями між процесами та порти. Процеси f та g читають дані з stdin, а результати пишуть в stdout, але не безпосередньо – вводом і виводом керує основна програма, вона посилає на вхід f і g дані та отримує від них результати. Забороняється використовувати допоміжні файли для обміну інформацією між процесами.

7. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) * g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант блокуючих операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і “зависанням” процесу на цей час). Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

8. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) && g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант блокуючих операцій обміну повідомленнями,

тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і “зависанням” процесу на цей час). Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

9. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) || g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант блокуючих операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і “зависанням” процесу на цей час). Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

10. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) * g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з “перериванням” обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень. Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

11. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) && g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з “перериванням” обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень.Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

12. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) || g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з “перериванням” обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень. Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

13. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) * g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант блокуючих операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і “зависанням” процесу на цей час). Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.

14. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) && g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант блокуючих

операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і "зависанням" процесу на цей час). Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати між собою.

15. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) || g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант блокуючих операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і "зависанням" процесу на цей час). Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати між собою.

16. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) * g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з "перериванням" обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати між собою.

17. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) && g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з "перериванням" обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати між собою.

18. **Взаємодія процесів. Паралелізм. Обмін повідомленнями через порт**. Обчислити f(x) || g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Використати обмін повідомленнями між процесами через порт (Socket). Реалізувати варіант неблокуючих операцій обміну повідомленнями, тобто з "перериванням" обчислень і їх продовженням (відновленням) після отримання повідомлень з результатами ініційованих допоміжних обчислень. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати між собою.

ЛАБОРАТОРНА РОБОТА 3

1. **Взаємодія потоків. Паралелізм. Спільна пам'ять**. Обчислити f(x) * g(x), використовуючи 2 допоміжні потоки: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію *. Використати спільну пам'ять (shared memory) для взаємодії. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати ні між собою.

2. **Взаємодія потоків. Паралелізм. Спільна пам'ять**. Обчислити f(x) && g(x), використовуючи 2 допоміжні потоки: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію &&. Використати спільну пам'ять (shared memory) для взаємодії. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати ні між собою.

3. **Взаємодія потоків. Паралелізм. Спільна пам'ять**. Обчислити f(x) || g(x), використовуючи 2 допоміжні потоки: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію ||. Використати спільну пам'ять (shared memory) для взаємодії. Функції f(x) та g(x) "нічого не знають друг про друга" і не можуть комунікувати ні між собою.

4. **Потоки. Паралелізм**. Помножити матриці A[n x m] та B[m x k]. Для обчислення створити (динамічно, автоматично) n*k [однотипних] потоків для обчислень (множення векторів). Передбачити введення матриці з клавіатури або авто-генерування та виведення результату обчислень "по ходу обчислень". Продемонструвати непослідовність обчислень елементів матриці-добутку.

5. **Взаємодія потоків, критичний сегмент**. Промоделювати паралельну роботу двох потоків (threads) з загальною коміркою пам'яті: а) з використанням критичного сегменту, б) без використання критичного сегменту. Продемонструвати різницю. (Наприклад, збільшувати значення спільної комірки на 1 1000 разів в кожному потоці.)

6. **Взаємодія процесів, критичний сегмент**. Промоделювати паралельну роботу двох процесів з загальною коміркою пам'яті, що належить третьому процесу, який надає 2 функції – прочитати і записати цю комірку. Наприклад, збільшувати значення спільної комірки на 1 1000 разів в кожному процесі. Побудувати та реалізувати "правильну" (в результаті значення комірки є 2000) і "неправильну" (значення може бути менше 2000) моделі взаємодії.

7. **Взаємодія процесів. Монітор стану процесу**. Промоделювати взаємодію двох процесів на наступному прикладі: один процес виводить повідомлення (в свій ListBox або в Memo) про запуск і зупинку (закриття вікна) другого процесу, а також його стан на початку роботи і при виході з першого (у вигляді повідомлення Message на модальному вікні першого процесу). В якості другого процесу треба взяти а) процес з фіксованим іменем запускного файлу процесу (File Name [.exe]), б) процес з фіксованим іменем вікна (Window Caption). Другий процес нічого не знає про перший і не робить ніяких специфічних дій щодо повідомлення про свій стан.

8. **Взаємодія процесів. Монітор стану процесу**. Промоделювати взаємодію двох процесів на наступному прикладі: один процес виводить повідомлення (в свій ListBox або в Memo) про стан другого процесу (запущений, закритий, не відповідає) при натисканні кнопки опитування або кожні N секунд. В якості другого процесу треба взяти а) процес з фіксованим іменем запускного файлу процесу (File Name [.exe]), б) процес з фіксованим іменем вікна (Window Caption). Другий процес нічого не знає про перший і не робить ніяких специфічних дій щодо повідомлення про свій стан.

9. **Семафор**. Запрограмувати Resource Server – сервер деякого ресурсу, який "видає" вільні ресурси процесам (реалізує операції P і V), що їх запитують і проводить індикацію стану ресурсу, а також програму – Client, яка запитує ресурси і відображає успішність операцій з ресурсами (за допомогою неї можна запитати і віддати ресурс). Програма Client повинна виводити інформацію про стан з великою деталізацією і бути запущеною кілька раз. Реалізувати моделювання управління а) іменованими ресурсами (наприклад, файли файлової системи), б) ресурсами без імен або з неважливими іменами (оперативна пам'ять, процесорний час).

10. **Тупики. Виявлення тупиків**. Реалізувати Resource Server (див. умову 5 варіанту), який управляє кількома ресурсами та Client, який може запитувати будь-який з ресурсів (Client повинен бути множинним, тобто запущений кілька

разів). Resource Server повинен виявляти тупикові ситуації (перехресний запит ресурсів клієнтами) та повідомляти про них (наприклад, через Message Box або в деякому журналі подій ListBox).

11. **Тупики. Виключення типиків**. Те ж саме, що і в попередньому варіанті, але Resource Server повинен не допускати виникнення тупиків і впливати на ситуацію, якщо виникає типик. Промоделювати на прикладі СУБД.

Мови програмування: C++ (C), JAVA (J), C#. Третя лабораторна робота – мова на вибір

№	Прізвище І.Б.	Лаб. 1	Лаб. 2	Лаб. 3
1	Васін Павло Олексійович	1 C	10 J	1
2	Гливинська Яна Сергіївна	2 C	11 J	2
3	Гирила Андрій Ігорович	3 C	12 J	3
4	Кушнір Антон Павлович	4 C	13 J	4
5	Кацара Аліна Ігорівна	5 C	14 J	5
6	Косінський Андрій Леонідович	6 C	15 J	6
7	Рудницький Андрій Валентинович	1 J	16 C	7
8	Тертерян Сергій Серопович	2 J	17 C	8
9	Харьков Олег Сергійович	3 J	18 C	9
10	Тимчишин Ігор Богданович	4 J	1 C	10
11	Мельников В'ячеслав-Євгеній Ігорович	5 J	2 C	11
12	Рижков Володимир Володимирович	6 J	3 C	1
13	Сірик Дмитро Сергійович	1 C#	4 C	2
14	Садовенко Богдан Володимирович	2 C#	5 C	3
15	Солдатенко Наталія Сергіївна	3 C#	6 C	4
16	Ярошевський Андрій Юрійович	4 C#	7C	5
17	Чан Ха Ву	5 C#	8 C	6
18	Шершньов Євгеній Вячеславович	6 C#	9 C	7
19	Швець Софія Вадимівна	1 C	10 J	8
20	Шаріфов Ількін Фірдовсі огли	2 C	11 J	9

Лектор _____ доц. В.М. Волохов