КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ПРОГРАМНИХ СИСТЕМ

«ЗАТВЕРДЖУЮ»

Заступник декана
з навчальної роботи

_____ Кашпур О.Ф.

«____»_____2019 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

АЛГЕБРО-АВТОМАТНІ МЕТОДИ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

для студентів

галузь знань	**12 Інформаційні технології**
спеціальність	**121 Інженерія програмного забезпечення**
освітній рівень	**магістр**
освітня програма	**Програмне забезпечення систем**
вид дисципліни	**обов'язкова**

Форма навчання	**денна**
Навчальний рік	**2020/2021**
Семестр	**2**
Кількість кредитів ECTS	**5**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**іспит**

Викладачі: **д. ф.-м. н., професор Кривий С.Л.** (лекції, лабораторні заняття).

Пролонговано: на 20__/20__ н. р. _____(_____) «__»___ 20__ р.

на 20__/20__ н. р. _____(_____) «__»___ 20__ р.

КИЇВ – 2019

Розробник: Кривий Сергій Лук’янович д. ф.-м. н., професор, професор кафедри інтелектуальних програмних систем.

ЗАТВЕРДЖЕНО

Завідувач кафедри інтелектуальних програмних систем

_____ О.І. Провотар

Протокол № __ від «___» _____2019 р.

Схвалено науково-методичною комісією факультету комп’ютерних наук та кібернетики

Протокол від «____» _____ 2019 року №___

Голова науково-методичної комісії _____ Л.Л. Омельчук

«_____» _____ 2019 року

Затверджено вченою радою факультету комп’ютерних наук та кібернетики

Протокол від «____» _____ 2019 року №___

Голова вченої ради факультету _____ А.В. Анісімов

1. Метою і завданням навчальної дисципліни «Алгебро-автоматні методи проектування програмного забезпечення» є опанування методів розробки програмного забезпечення на основі методів теорії автоматів, алгебраїчних структур, програмування з обмеженнями та їх застосування до розробки, аналізу і верифікації алгоритмів та програм.

2. Попередні вимоги до опанування навчальної дисципліни. Для успішного вивчення дисципліни «Алгебро-автоматні методи проектування програмного забезпечення» студенти повинні відповідати наступним вимогам:

1. Успішне опанування курсів:
    1. Дискретна математика.
    2. Основи програмування.
    3. Основи об'єктно-орієнтованого програмування.
    4. Об'єктно-орієнтоване програмування.
    5. Системне програмування та операційні системи.
2. Знання:
    1. Основних концепцій процедурного, об'єктно-орієнтованого програмування.
    2. Процедурного та об'єктно-орієнтованого програмування мовою C++.
    3. Об'єктно-орієнтованого програмування мовою Java.
    4. Основних понять теорії компіляторів.
3. Вміти:
    1. Виконувати аналіз проблеми, що виникає.
    2. Будувати математичні моделі відповідних предметних областей.
    3. Створювати специфікації та виконувати верифікацію моделей, що виникли в результаті аналізу проблеми.
    4. Розв'язувати задачі перевірки специфікацій на моделях.
    5. Програмувати у процедурному, функціональному, логічному та об'єктно-орієнтованому стилях.
4. Володіти:
    1. Базовими навичками використання інтегрованих середовищ розробки програмного забезпечення CLion/ IntelliJIDEA/ Eclipse/ NetBeans.
    2. Англійською мовою на рівні не нижче Intermediate.

3. Анотація дисципліни. Навчальна дисципліна «Алгебро-автоматні методи проектування програмного забезпечення» є складовою програми підготовки фахівців за другим (магістерським) рівнем вищої освіти галузі знань 12 Інформаційні технології зі спеціальності 121 Інженерія програмного забезпечення, освітньо-наукової програми «Програмне забезпечення систем».

Дисципліна викладається в 1 семестрі в обсязі **150 годин (5 кредитів ECTS),** зокрема: лекції – 34 год., лабораторні заняття – 14 год., самостійна робота – 100 год., консультації – 2 год. Викладання дисципліни закінчується – **іспитом.**

Дана дисципліна є нормативною навчальною дисципліною в рамках освітньої програми «Програмне забезпечення систем» та є базовою для вивчення дисциплін «Прикладні логіки та елементи квантових обчислень» та «Некласичні логіки та їх застосування в розробці програмного забезпечення».

4. Завдання (навчальні цілі). Основними завданнями дисципліни «Алгебро-автоматні методи проектування програмного забезпечення» є набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень в області проектування програмного забезпечення відповідно до освітньої кваліфікації магістр з інженерії програмного забезпечення. Зокрема, розвивати:

- Здатність до абстрактного мислення, аналізу та синтезу (ЗК01).
- Здатність проведення теоретичних та прикладних досліджень на відповідному рівні (ЗК03).
- Здатність аналізувати предметні області, формувати, аналізувати та моделювати вимоги до програмного забезпечення (СК01).
- Здатність проектувати програмне забезпечення, включаючи проведення моделювання його архітектури, поведінки та процесів функціонування окремих підсистем і модулів (СК03).
- Вміння планувати і проводити наукові дослідження, готувати результати наукових робіт з інженерії програмного забезпечення до оприлюднення (СК09).

5. Результати навчання за дисципліною.

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		**Форми (та/або методи і технології) викладання і навчання**	**Методи оцінювання та пороговий критерій оцінювання (за необхідності)**	**Відсоток у підсумковій оцінці з дисципліни**
Код	**Результат навчання**			
РН1.1	Знати основні алгоритми теорії автоматів, їх складність та варіації (базовий рівень).	Лекції, лабораторні заняття, самостійна робота.	Контрольна робота №1, іспит.	10%
РН1.2	Знати основні поняття та концепції загальної алгебри (напівтрупи, групи, гратки, багатоосновні алгебри тощо).	Лекції, лабораторні заняття, самостійна робота.	Контрольні роботи №1, №2, іспит.	5%
РН1.3	Знати основні етапи повного циклу проектування, їх задачі та методи розв’язання.	Лекції, лабораторні заняття, самостійна робота.	Контрольна робота №2, іспит.	35%
РН2.1	Вміти проектувати та розробляти на основі побудованих математичних моделей алгоритми розв’язання поставлених задач, виконувати аналіз складності побудованих алгоритмів.	Лабораторні заняття, самостійна робота.	Здача лабораторних робіт, іспит.	20%
РН3.1	Консультуватися з викладачем стосовно питань що виникають у ході вивчення теоретичного матеріалу.	Лекції, лабораторні заняття, години консультацій.	Поточне оцінювання, здача лабораторних робіт.	5%
РН3.2	Обговорювати з колегами та викладачем проблемі питання що виникають у ході виконання	Лабораторні заняття.	Здача лабораторних робіт.	5%

	лабораторних робіт.			
РН3.3	Чітко та послідовно обґрунтовувати власні проектні рішення в рамках виконання лабораторних робіт.	Лабораторні заняття.	Здача лабораторних робіт.	5%
РН4.1	Закріплення та поглиблення набутих на лекціях теоретичних знань проектування та розробки програм.	Самостійна робота, опрацювання рекомендованих інформаційних джерел.	Поточне оцінювання, здача лабораторних робіт.	5%
РН4.2	Закріплення та поглиблення набутих на лабораторних заняттях практичних навичок проектування та розробки програм.	Самостійна робота, реалізація рекомендованих завдань.	Поточне оцінювання, здача лабораторних робіт.	5%
РН4.3	Лідерство та повна автономність під час реалізації лабораторних проектів.	Лабораторні заняття.	Здача лабораторних робіт.	5%

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання.

Результати навчання дисципліни / **Програмні результати навчання**	**РН 1.1**	**РН 1.2**	**РН 1.3**	**РН 2.1**	**РН 3.1**	**РН 3.2**	**РН 3.3**	**РН 4.1**	**РН 4.2**	**РН 4.3**
ПРН01. Знати і системно застосовувати методи аналізу та моделювання прикладної області, виявлення інформаційних потреб і збору вихідних даних для проектування програмного забезпечення.	+	+	+	+						
ПРН03. Знати і застосовувати базові концепції і методології моделювання інформаційних процесів.					+	+	+			
ПРН08. Проводити аналітичне дослідження параметрів функціонування програмних систем для їх валідації та верифікації, а також проводити аналіз обраних методів, засобів автоматизованого проектування та реалізації програмного забезпечення.				+			+	+		
ПРН11. Набувати нові наукові і професійні знання, вдосконалювати навички, прогнозувати розвиток програмних систем та інформаційних технологій.						+			+	+
ПРН13. Набувати нові наукові і професійні знання, вдосконалювати навички, прогнозувати розвиток програмних систем та інформаційних технологій.			+				+	+	+	+
ПРН14. Пояснити, аналізувати, цілеспрямовано шукати і обирати необхідні для вирішення фахових наукових і прикладних задач інформаційно-			+		+	+	+	+		

довідкові та науково-технічні ресурси і джерела знань з урахуванням сучасних досягнень науки і техніки.										

7. Схема формування оцінки.

7.1. Форми оцінювання студентів.

Семестрове оцінювання:

1. Контрольна робота 1: РН1.1, РН1.2 – **12 балів/7 балів**.
2. Контрольна робота 2: РН1.2, РН1.3 – **12 балів/7 балів**.
3. Лабораторна робота 1: РН2.1, РН3.1, РН3.2, РН3.3, РН4.1, РН4.2, РН4.3 – **10 балів/6 балів**.
4. Лабораторна робота 2: РН2.1, РН3.1, РН3.2, РН3.3, РН4.1, РН4.2, РН4.3 – **20 балів/12 балів**.
5. Поточне оцінювання: РН3.1, РН4.1, РН4.2 – **6 балів/4 бали**.

Підсумкове оцінювання (у формі іспиту):

- Максимальна кількість балів які можуть бути отримані студентом: **40 балів**.
- Результати навчання які будуть оцінюватись: РН1.1, РН1.2, РН1.3, РН2.1.
- Форма проведення і види завдань: письмова.
- Види завдань: 5 письмових завдань (2 теоретичних питання та 3 практичних завдання).
- Для отримання загальної позитивної оцінки з дисципліни оцінка за іспит повинна бути не меншою ніж 24 бали.
- Студенти не допускаються до іспит, якщо протягом семестру вони набрали менше ніж 36 балів.
- Студенти не допускаються до іспиту, якщо протягом семестру вони не виконали та не здали 100 % лабораторних робіт передбачених даною програмою.

Критерії оцінювання на іспиті.

Завдання	Тема завдання	Максимальний відсоток від 40 балів	Всього відсотків
Завдання 1	Теоретичні питання за матеріалами курсу	15%	15%
Завдання 2		22,5%	22,5%
Завдання 3	Практичне завдання на основі теоретичного матеріалу курсу	25%	25%
Завдання 4		20%	20%
Завдання 5		17,5%	17,5%
			100%

7.2 Організація оцінювання.

Терміни проведення форм оцінювання:

1. Контрольна робота 1: до 7 тижня семестру.
2. Контрольна робота 2: до 14 тижня семестру.
3. Лабораторна робота 1: до 7 тижня семестру.
4. Лабораторна робота 2: до 13 тижня семестру.
5. Поточне оцінювання: протягом семестру.

Студент має право на одне перескладання кожної контрольної роботи у визначений викладачем термін.

У випадку відсутності студента з поважних причин відпрацювання та перездачі контрольних робіт здійснюються у відповідності до «Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу» від 1 жовтня 2010 року. Якщо студент пропустив без поважних причин більше половини занять, не виконав хоча б однієї лабораторної роботи або не писав контрольної роботи, то він не допускається до складання іспиту.

Студент повинен здавати лабораторні роботи в назначені викладачем терміни. Якщо лабораторні робота не здається в назначені терміни, вона вважається такою, що не виконана студентом.

7.3. Шкала відповідності оцінок.

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і лабораторних занять.

№ лекції	Назва лекції	Кількість годин		
		Лекції	**Лабораторні заняття**	**Самостійна робота**
Частина 1. Задачі і проблематика сучасного стану методів розробки програмного забезпечення.				
1	**Тема 1.** Предмет курсу та проблеми, які розглядатимуться. Поняття алгоритму та його уточнення у вигляді алгоритмічної системи. Частково рекурсивні функції та машини Тюрінга.	2		5
2	**Тема 2.** Асимптотичне порівняння функцій. Теорема майстрів. Приклади застосування теореми та оцінок росту функцій. Основні етапи процесу проектування алгоритмів.	2		5
3	**Тема 3.** Характеристика етапів розробки: етап побудови математичної моделі, аналіз моделі, розробка алгоритму, його обґрунтування та оцінка складності.	2	2	5
4	**Тема 4.** Етап програмування: тестування та від лагодження програмного забезпечення. Основні проблеми, можливості автоматизації.	2	2	5
5	**Тема 5.** Прості методи розробки. Рекурентні співвідношення та рекурсивні означення. Приклади розробки простих програм чисельних та символьних обчислень.	2		10
6	**Тема 6.** Основні методи розробки. Метод «поділяй та владарюй». Приклади застосування. Динамічне програмування. Приклади застосування та порівняння з попереднім методом.	2		9
7	**Тема 7.** Жадібні алгоритми та теоретичні основи цих алгоритмів. Приклади застосування.	2	2	10

	Метод пошуку з поверненням. Приклади застосування. Метод структурного проектування, програмування та обґрунтування. Приклади застосування.			
8	**Тема 8.** Складність алгоритмів і програм в моделі Тюрінга та арифметичній моделі. Приклади. Арифметична складність алгоритмів реалізації арифметичних операцій.	2		9
Контрольна робота 1				2
Всього по частині 1		16	6	60
Частина 2. Основні області застосування методів розробки програмного забезпечення.				
9	**Тема 9.** Алгоритми типу поділяй та владарюй: загальна схема та її аналіз. Приклади застосування: сортування злиттям, уніфікація термів в абсолютно вільній алгебрі.	2	2	5
10	**Тема 10.** Чисельні алгоритми обчислення значень поліномів, операцій на матрицях та розв’язання системи лінійних рівнянь.	2	2	5
11	**Тема 11.** Алгоритми порівняння із зразком. Паралельні алгоритми та методи їх аналізу. Моделі паралельних обчислень. Приклад множення матриць в моделі CRCW PRAM.	2		5
12	**Тема 12.** Верифікація програм. Методі потокового аналізу та їх застосування. Приклади. Оптимізація програм: оптимізація на етапі розробки та на етапі трансляції.	2	2	4
13	**Тема 13.** Оптимізація процедур на основі інваріантних співвідношень. Структури даних: АТД та їх властивості. Скалярний тип даних та його властивості. Приклади.	2	2	5
14	**Тема 14.** Структурний тип даних: підпрограми, спів програми, процедури-функції. Приклади: лінійні списки, двозв’язні списки, двічі прошиті списки.	2		5
15	**Тема 15.** Фундаментальні структури даних: множини, відображення, послідовності. Аналіз властивостей.	2		4
16	**Тема 16.** Черги, черги з пріоритетами, словники, чорно-білі дерева та їх властивості. Приклади застосування.	2		5
17	**Тема 17.** Підведення підсумків. Дискусія на тему: що таке хороша програма і хороший програміст?	2		
Контрольна робота 2				2
Всього по частині 2		18	8	40
Консультації		2		
ВСЬОГО		**34**	**14**	**100**

Загальний обсяг **150** годин, в тому числі:
Лекції – **34** год.

Лабораторні заняття – **14** год.
Самостійна робота – **100** год.
Консультації – **2** год.

Типові завдання контрольної роботи № 1.

1. Теоретичні підстави методів розробки «поділяй і владарюй», динамічного програмування, жадібного вибору. Поняття матроїда та його властивості.
2. Методи тестування програмного забезпечення та способи його автоматизації.
3. Виконати обґрунтування бінарного алгоритму обчислення НСД двох чисел.
4. Знайти його оцінку часової складності в моделі Тюрінга та арифметичній моделі.

Контрольні запитання до змістовної частини І.

1. Дати означення примітивно рекурсивної функції та частково рекурсивної функції. Означення машини Тюрінга, машини Поста та нормального алгоритму Маркова.
2. Асимптотичне порівняння функцій. Теорема майстрів. Приклади застосування теореми та оцінок росту функцій. Основні етапи процесу проектування алгоритмів.
3. Характеристика етапів розробки: етап побудови математичної моделі, аналіз моделі, розробка алгоритму, його обґрунтування та оцінка складності. Навести приклади.
4. Етап програмування: тестування та від лагодження програмного забезпечення. Основні проблеми, можливості автоматизації, вибір мови програмування. Парадигми програмування: процедурна, функціональна, логічна та об’єктно-орієнтована, паралельна. Операційна та денотаційна семантика мов програмування, їх зв'язок та роль в обґрунтуванні програмного забезпечення.
5. Прості методи розробки. Рекурентні співвідношення та рекурсивні означення. Проблеми застосування цих методів та приклади розробки простих програм чисельних та символьних обчислень.
6. Основні методи розробки. Метод «поділяй та владарюй». Приклади застосування. Динамічне програмування. Приклади застосування та порівняння обох методів. Умови використання того чи іншого методу.
7. Жадібні алгоритми та теоретичні основи цих алгоритмів. Приклади застосування. Метод пошуку з поверненням та метод структурного проектування, програмування та обґрунтування. Приклади застосування.
8. Складність алгоритмів і програм в моделі Тюрінга та арифметичній моделі. Приклади. Арифметична складність алгоритмів реалізації арифметичних операцій.
9. Теорема майстрів для оцінки складності програм, розроблених на основі
10. рекурентних співвідношень та рекурсивних означень.
11. Означення матроїда та його застосування в алгоритмах жадібного вибору.
12. Алгоритм найближчого сусіда та алгоритм Дейкстри-Пріма.

Типові завдання контрольної роботи № 2.

1. Охарактеризувати основні підходи до верифікації програмного забезпечення. В чому полягає складність проблеми верифікації?

2. Методи пошуку програмних інваріантів та їх застосування до оптимізації процедур. Оптимізації за входом і виходом процедури. Обґрунтування перетворень згортки констант та їх розгортки. В чому полягають ці перетворення та яка їх оптимізаційна ефективність.
3. Складність в моделі Тюрінга та арифметичній моделі? Що означає строго поліноміальна складність алгоритмів і програм?
4. Описати основні фундаментальні структури даних, операції на цих структурах та методи їх реалізації.

Контрольні запитання до змістовної частини ІІ.

1. Охарактеризувати основні методи розробки програмного забезпечення. За яких умов використовується той чи інший спосіб розробки.
2. Сформулювати основні властивості принципу жадібного вибору. Коли жадібний вибір веде до ефективної програми?
3. Чому необхідно розглядати та розробляти методи розв'язання рекурентних співвідношень? Які основні методи розв'язання таких співвідношень Вам відомі?
4. Сформулювати задачу пошуку інваріантних співвідношень та її основні під задачі. В чому полягає необхідність пошуку такого типу співвідношень? Охарактеризувати алгоритм MBA та MBA.
5. Сформулювати означення скалярного та структурного типу структур даних. Які структури до таких типів відносяться? Обґрунтувати відповідь конкретними прикладами.
6. Які оптимізаційні перетворення Вам відомі. За яких умов використовуються оптимізаційні перетворення процедур та який ефект вони дають для оптимізації? В чому полягають перетворення пов'язані з згорткою та розгорткою констант, вилучення константних умов, оптимізації розподілу регістрів, злиття циклів в програмах, оптимізація лінійних ланок?
7. Метод реалізації та системи операцій на фундаментальних структурах даних. Описати основні властивості чорно-білих дерев, АВЛ-дерев, куп, словників тощо.
8. Застосування модальних логік до проблеми верифікації програм реактивного типу. Лінійна темпоральна логіка та її основні властивості.

9. Рекомендовані джерела:

Основні:

1. Кривий С.Л. Дискретна математика: вибрані питання. – Вид. дім «Києво-Могилянська академія». – 2007. – 570 с.
2. Глушков В.М., Цейтлин Г.Е, Ющенко Е. Л. Алгебра, языки, программирование. К.: Наукова думка, 1989. – 376 с.
3. Кривий С.Л. Вступ до методів створення програмних продуктів. Чернівці: Букрек. – 2013 р. – 424 с.
5. Манна З. Семантика неподвижной точки функциональных программ. – Киб. сборник. Вып. 15. – М.Мир. – 1979. – С. 38-100.

6. Зикман Й., Сабо Р. Универсальная унификация и классификация эквациональных теорий. – Кибернетический сборник. – 1986. – вып. 21. – С. 213-234.
7. Филд А., Харрисон П. Функциональное программирование. – М. Мир. – 1993. – 742 с.
8. Коблиц И. Курс теории чисел и криптографии. – М. Научное издат. ТВП. – 2001. – 260 с.
9. Крывый С. Л. Алгоритмы решения систем линейных диофантовых уравнений в полях вычетов. – ж. Кибернетика и сист. анализ, 2007. – N 2. – С. 15-23.
10. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. – 535 с.
11. Ахо А., Ульман Дж. Теория синтаксического анализа и компиляции, т. 2 Компиляция. М.: Мир, 1978 – 486 с.
12. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Издат. дом «Вильямс». – 2000. – 382 с.
13. Вирт Н. Систематическое программирование. Введение. – М.: Мир. – 1977. – 183 с.
14. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир. – 1985. – 406 с.
15. Гери М. Р., Джонсон Д. С. Вычислительные машины и трудно вычислимые задачи. – М.: Мир. – 1982. – 416 с.
16. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм. ж. Кибернетика. – 1965. – № 5. – С. 1-9.
17. Годлевский А. Б., Кривой С. Л. Трансформационный синтез эффективных алгоритмов с учетом дополнительных спецификаций. ж. Кибернетика. – 1986. – № 6. – С. 37-43.
18. Годлевский А.Б., Капитонова Ю.В., Кривой С.Л., Летичевский А.А. Итеративные методы анализа программ. ж. Кибернетика. – 1989. – № 2. – С. 9-19.
19. Капитонова Ю. В., Летичевский А. А. Об основных парадигмах программирования. ж. Кибернетика и сист. анализ. – 1994. – № 6. – С. 3-20.
20. Кривий С. Л. Дискретна математика. – Київ-Чернівці: Букрек. – 2017. – 568 с.
21. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ (второе издание). – Издательский дом «Вильямс». – 2005. – 1290 с.
22. Мальцев А. И. Алгоритмы и рекурсивные функции. – М.: Наука. – 1986. – 367 с.
23. Fisher M.J., Ladner R.E. Propositional Modal Logic of Programs. In Proc. 9-th ACM Ann. Symposium on Theory of Computing. – 1977. – P. 286-294.
24. Fisher M.J., Ladner R.E. Propositional Dynamic Logic of regular programs. J. Comp. System Sci. – 1979. – v.18. - № 2. – P. 194-211.
25. Goldblatt R. Logics of Time and Computation. – Lecture Notes. – № 7. Center for the Study of Language and Information. – 1987. – P. 1-27.

Додаткові:

1. Седжвик Р. Фундаментальные алгоритмы на C++. Части 1-4. – М.: DiaSoft. – 2002. – 687 с.
2. Atallach M. J. (editor). Algorithms and Theory of Computation Handbook. CRC Pres. – 1999. – v.1 (Searching). – P. 29-34.

3. Ershov A.P. Mixed computation: potential applications and problem for study. Theor. Comput. Science. – 1982. – 18. – P. 41-68.
4. Johnson D. S. A Catalogue of Complexity Classes. – In J. van Leeuwen., ed. «Handb. of Theoret. Computer Science'». – v. A. – 1990: Elsevier. – P. 67-161.
5. Kam J.B., Ullman D.J. Monotone data flow analysis frame works. Acta Inform. – 1978. – №3. – P. 305-318.
6. Kildall G.A. A unified approach to program optimization. Conf. Rec. of ACM Symp. on Prince. Of Program Languages. Boston, Massachusetts. – October 1-3. – 1973. – P. 194-206.
7. Papadimitriou C. H. Computational complexity. – Addison-Wesley. – 1994. – 532 p.
8. Winograd S. On the number of multiplications necessary to compute certain functions. Journ. Pure and Applied Mathematics. – 1970. – P. 165-179.