Державний університет телекомунікацій

Навчально-науковий інститут телекомунікацій та інформатизації

Кафедра комутаційних систем

Методичне керівництво для виконання практичної роботи

# **Системи програмування**.

з дисципліни «Програмне забезпечення телекомунікаційних систем».

освітньо-кваліфікаційного рівня магістр

Київ - 2013

Укладачі: проф. каф. КС Кунах Н.І.
асистент. каф. КС Невдачина О.В.

Методичні вказівки обговорені і схвалені
на засіданні кафедри КС

Протокол № _____
від «___» _____ 2013р.

**Тема: Системи програмування.**

**Мета заняття:**
Вивчення базових основ систем програмування. Розгляд основних груп мов програмування, які використовуються для розробки програмного забезпечення телекомунікаційних систем.

**Час проведення заняття:** 90 хвилин.

**Список літератури.**

1. Гребешков А.Ю. «Микропроцессорные системы и программное обеспечение в средствах связи», ПГУТИ Самара, 2009. (надається в електронному вигляді).
2. Голицина О.Л., Партыка Т.Л. «Програмное обеспечение», 2-ое издание, Москва, 2008. (надається в електронному вигляді).

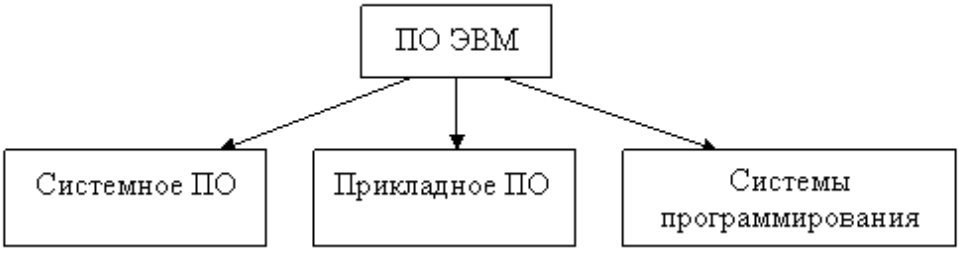
**Зміст заняття.**

1. **Ознайомча частина.**

Перевірка присутності студентів, оголошення теми заняття та порядок його проведення.

2. **Матеріал для вивчення:**

Існує класифікація програмного забезпечення (ПЗ), згідно з якою всі програми діляться на системні, прикладні та системи програмування.




**Системи програмування** - це комплекс інструментальних програмних засобів, призначений для роботи з програмами на одній з мов програмування. Системи програмування надають сервісні можливості програмістам для розробки їх власних комп'ютерних програм. Розробку засобів системного ПЗ і систем програмування прийнято називати системним програмуванням; розробку прикладних програм називають прикладним програмуванням.

Розробка будь-якого системного та прикладного ПЗ здійснюється за допомогою систем програмування, до складу яких входять:

- Транслятори з мов високого рівня;
- Засоби редагування, компонування і завантаження програм;
- Макроассемблера (машинно-орієнтовані мови);
- Відладчики машинних програм.

*Транслятор* - це комплекс програм, що забезпечує переклад програми, написаної на символічній мові, в сукупність машинних кодів. Залежно від функціонального призначення виділяють два основних види трансляторів: компілятор і інтерпретатор.

*Компілятор* - це транслятор, що забезпечує переклад програми з алгоритмічної мови на машинну без її виконання.

*Інтерпретатор* - це транслятор, що забезпечує переклад кожної конструкції алгоритмічної мови в машинний код з одночасним виконанням.

*Відладчики* - це спеціальні програми, призначені для трасування (відстеження виконання програми в пооператорному варіанті), ідентифікації місця і виду помилок у програмі, «спостереження» за зміною значень змінних, виразів і т.п.

Системи програмування, як правило, включають в себе:

- Текстовий редактор (Edit), який здійснює функції запису та редагування вихідного тексту програми;
- Завантажувач програми (Load), що дозволяє вибрати з директорії потрібний текстовий файл програми;
- Запускач програм (Run), який здійснює процес виконання програми;
- Компілятор (Compile);
- Відладчик (Debug);
- Диспетчер файлів (File), що надає можливість виконувати операції з файлами: збереження, пошук, знищення тощо

Ядро системи програмування становить мова. Мова програмування - формалізована мова для опису алгоритму розв'язання задачі на комп'ютері.

Існуючі мови програмування можна розділити на дві групи:

1) процедурні: - низького рівня,
- високого рівня,
2) непроцедурні: - об'єктно-орієнтовані,
- декларативні.

**Процедурні** (або алгоритмічні) програми являють собою систему приписів для вирішення конкретного завдання. Роль комп'ютера зводиться до механічного виконання цих приписів.

Мови низького рівня (машинно-орієнтовані) дозволяють створювати програми з машинних кодів. З ними важко працювати, але створені з їх допомогою програми займають менше місця в пам'яті і працюють швидше (мови автокодом, Асемблери).

Програми на мовах високого рівня близькі до природної (англійської) мови та представляють набір заданих команд.

МП високого рівня мають наступні переваги:

- Алфавіт мови значно ширше машинного, це робить його набагато більш виразним;
- Конструкції команд відображають змістовні види обробки даних і задаються в зручному для людини вигляді;
- Використовується апарат змінних і дії з ними;
- Підтримується широкий набір типів даних.

До процедурних мов відносяться такі мови програмування, як Algol, Fortran, Basic, Pascal, C.

Прикладом мов ООП є C + +, Java.

Класами декларативних мов є функціональні мови ( Lisp) і логічні мови ( Prolog).

## Системи програмування.

### 1. Об'єктно-орієнтований підхід до програмування.

Основу цього підходу складають такі поняття.

1. Об'єкт, який представляє пристрій або інформацію про нього, що підлягає обробці в програмі.

2. Атрибут. Кожен об'єкт описується списком атрибутів, які, з одного боку, дають визначення всіма характеристиками об'єкта, з іншого - вказують на параметри, які можуть бути використані зовнішнім оточенням. Співвідношення об'єкт-атрибут аналогічно відношенню функція-параметр в математиці. Це іноді дозволяє записувати об'єкт у вигляді функції.

3. Повідомлення (метод.) Запити на роботу об'єкта називаються повідомленнями або методами. Об'єкти можуть підтримувати більш ніж один

метод. Методи складаються з двох частин: опис (signature) і реалізація (implementation). Опис складається з імені методу, імен і типів параметрів (у тому числі повертається значення) та вилучення (exceptions).

4. Інтерфейс об'єкта являє собою частину об'єкта, призначену для взаємодії із зовнішнім оточенням. Сукупність інтерфейсів визначає поведінку об'єкта. Інтерфейс об'єкта складається з імені об'єкта і набору методів.

5. Часто з інетерфейсом пов'язане поняття класу. Клас має внутрішню частину, визначену структурну якість, і зовнішню частину, що містить групування всіх об'єктів класу і конструктор для кожного класу.

6. Поведінка об'єкта - це контракт, опис поведінки об'єкта при управлінні його програмою, яка пропонується споживачеві. Контракт складається з опису та необхідних передумов (Preconditions), інваріантів (invariants) і постумови (post-conditions). Інваріанти - передумови - умови, правильність яких повинна бути забезпечена перед викликом методу. Післяумови - умови, правильність яких перевіряється після виклику методу і підтверджує правильність виконання контракту.

## 2.Архітектура CORBA для розподілених обчислювальних систем

Для побудови програмного забезпечення, його розробки та експлуатації в багатьох додатках застосовується CORBA (Common Object Request Broker Architecture) - Загальна архітектура з передачею запитів до об'єкта через посередника. Часто вона класифікується як система роботи з розподіленими обчислювальними ресурсами. Її властивості багато в чому відповідають вимогам до розробки та експлуатації програмного забезпечення станцій в мережах телекомунікацій.

Частина архітектури CORBA, необхідна для подальшого розгляду, показана на рис. 1.


**Рис. 1. Фрагмент архітектури CORBA**

Відповідно до сучасної класифікації зазначена архітектура належить класу архітектури "клієнт-сервер". Як можна бачити на рис. 1, є два типи мережевих пристроїв:

• клієнти (при об'єктно-орієнтованому підході це об'єкти);

• сервери (обслуговуючі вузли мережі).

Робота здійснюється сервером за запитом клієнта на послугу.

Основною особливістю архітектури CORBA є наявність програмної шини ORB (Object Request Broker). ORB "відповідає" за розподіл запитів клієнта по відповідним серверам, а також є транспортним середовищем для взаємодії в процесі виконання завдання. Деталі цієї взаємодії приховані від користувача і існують як частина реалізації "програмної шини".

Зазвичай прихованими є наступні елементи взаємодії:

• Розташування об'єкта. Клієнт не знає, де розташовується цільовий об'єкт - в окремому процесі в іншому комп'ютері мережі або в іншому процесі того ж самого комп'ютера мережі.

• Реалізація об'єкта. Клієнт не знає, яким чином реалізований об'єкт, яка мова програмування або мова сценаріїв використана для реалізації, яка операційна система або яке обладнання працює в об'єкті.

• Стан об'єкта. При передачі запиту до цільового об'єкту клієнт не повинен знати, чи активізований об'єкт (тобто чи запущений відповідний процес) і чи готовий він прийняти запит. У разі необхідності, перед тим як передати до об'єкта запит послуги, компонент ORB невидимо для клієнта запускає процес на об'єкті.

• Механізм зв'язку з об'єктом. Клієнту невідомий механізм (наприклад, протокол, Колективна пам'ять, виклик локального методу і т. д.).

Ці особливості ORB дають можливість розробникам додатків сконцентрувати увагу на додатках і не турбуватися про проблеми розподіленого системного програмування нижнього рівня.

Такий підхід спрощує модернізацію, пошук збоїв і відхилень від нормального функціонування та інші дії при експлуатації станції.

## 3.Табличний метод, або метод універсальної програми.

Розглянемо ще один метод, який відповідає викладеним властивостям об'єктно-орієнтованого підходу і розподілених систем програмування. Він використовувався при створенні вітчизняних систем комутації.

Його характерні особливості:

• Математичне завдання об'єктів, яке виявляється у запису алгоритмів у вигляді рекурсивних або табличних функцій.

• Опис процесів з точки зору автоматної моделі.

Об'єкти

Об'єктами, над якими проводяться дії, є комплекти станції, комутаційні поля. У даному випадку будуть розглядатися станції. Іноді для обробки використовуються віртуальні об'єкти.

Згідно концепції об'єкта він має атрибути, необхідні для роботи з цим об'єктом. Наприклад, це можуть бути:

1. Тип комплекту. Цей атрибут у нашому прикладі дозволяє визначити перелік програм обробки абонентського виклику.
2. Номер комплекту. На станції може бути сотні тисяч комплектів одного типу. Номер дозволяє визначити, з яким з них треба працювати.
3. Стан комплекту. Це атрибут вказує, на якому етапі знаходиться робота, здійснювана зовнішнім оточенням з цим комплектом (комплект вільний, зайнятий, заблокований і т. д.) Значення цього атрибута, на відміну від розглянутих вище атрибутів, змінюється в процесі роботи.

Кожен з цих атрибутів може бути пронумерований і представлений, як параметр функції, званий абонентським комплектом – $f_{ак}$ (x1, x2, x3, ..... xn).

Аналогічні атрибути мають і інші комплекти. Наприклад, в повністю децентралізованою системі приймач багаточастотного набору (БЧПП) має ті ж атрибути, що і абонентський комплект:

1. Тип комплекту
2. номер комплекту
3. стан комплекту

Однак він має кілька своїх характерних атрибутів, наприклад:

1. Номер прийнятої цифри (можливі значення 1, 2, 3, ...)
2. Значення 1-й цифри
3. Значення 2-й цифри
4. і т. д.

Ці атрибути також беруть участь в обробці виклику.
Динамічний об'єкт - з'єднання

Об'єкти можуть бути віртуальними (уявними), які не існують реально, але описані логічно і відображаються тільки в пам'яті. Одним з таких важливих об'єктів є з'єднання.

Для багатьох систем з'єднання існує в реальному обладнанні, відображаючи тимчасову сукупність приладів, що реалізують фізичне з'єднання двох кінцевих терміналів.

Наведемо приклад. Нижче (рис. 2) наводимо скорочений варіант для ілюстрації поняття віртуального об'єкта "з'єднання". У цьому прикладі абонентський комплект (АК1) з'єднується з модулем багаточастотного передавача (БЧПП) для передачі набору номера. Згідно з алгоритмом абонент набирає номер вхідного абонента, після чого інша частина багаточастотного прийомопередавача (БЧПП) набору номера під'єднується до модуля абонентських аналогових ліній вхідного абонента (абонентському комплекту АК2) для визначення його стану. Після цих дій на станції з'являється новий об'єкт "з'єднання", який відображається за допомогою адрес зв'язки. Для цього до атрибутів реального об'єкта додається атрибут номеру адреси зв'язку - Nзв. Його значення вказує номери типу та номер комплекту, поєднаного з даними. Тоді функція, що відображає, наприклад, об'єкт АК1 як функцію за допомогою його атрибутів (параметрів), має вигляд

$f_{АК1}=f(N_{тип}, N_{АК1},N_{зВ})$, де

**Nтип** - номер типу комплекту, в даному випадку тип АК (зазвичай цьому типу присвоюється номер 1);

**NАК1** - номер комплекту серед свого типу АК (наприклад, від 100 до 10000 в залежності від ємності станції);

**NБЧПП** - номер комплекту серед свого типу БЧПП (наприклад, від 100 до 10000 в залежності від ємності станції);

**Nзв** - номер зв'язку (значення визначаються типом і номером в типі комплектів, що беруть участь у з'єднанні).

Аналогічно відображається розглянутий у прикладі об'єкт БЧПП

$f_{БЧПП} = f(N_{тип}, N_{БЧПП} , N_{ЗВ,}, N_1, …, N_K)$.

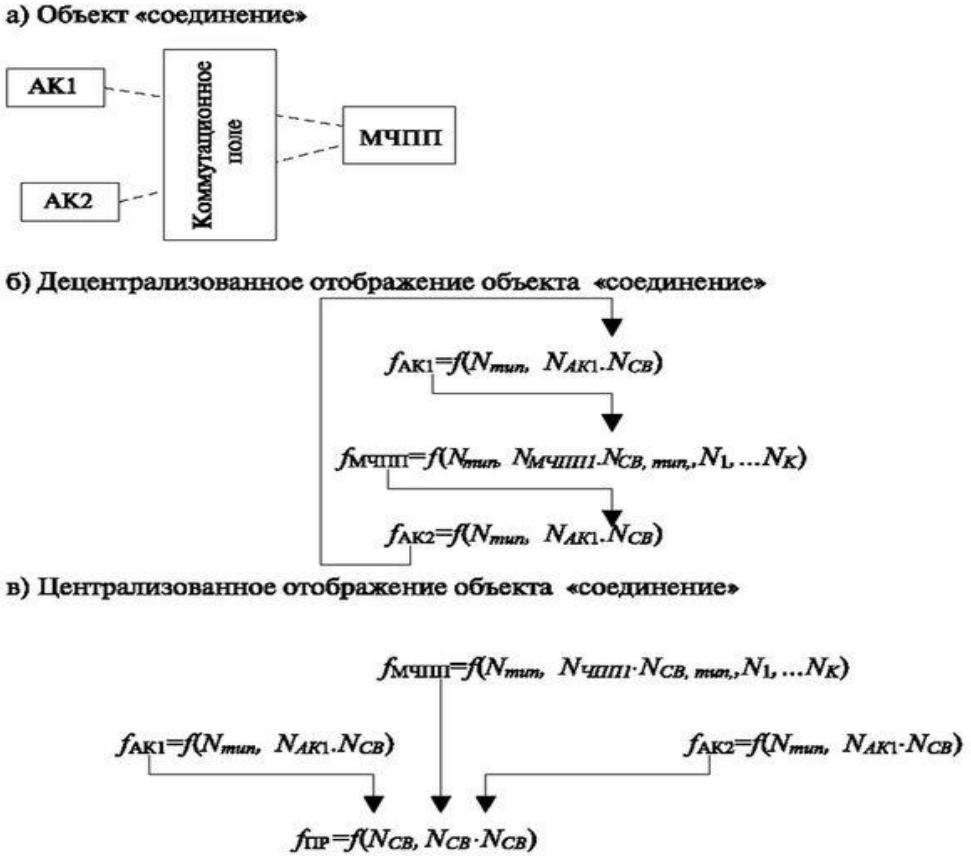



**Рис. 2. Принцип створення і відображення об'єкта "з'єднання" а) умовне зображення з'єднання реального обладнання, б) розподілене відображення з'єднання за допомогою адреси зв'язку. в) централізоване відображення з'єднання за допомогою адреси зв'язку.**

Відмінність тільки в номері типу, який повинен приймати значення, закріплене за цим типом комплекту, і в додаткових параметрах характеристики набраного номера, про які сказано вище, позначених N1, N2 ..., Nk.

Зауважимо, що об'єкт "з'єднання" є динамічним. Він змінюється в процесі встановлення з'єднання. У процесах, що забезпечують передачу даних, він не обов'язково має аналог з'єднання реального обладнання. Він може відображати параметри віртуального шляху або каналу. Функціонально об'єкт "з'єднання" може бути відображений двома способами децентралізованим (рис. 2б), централізованим (рис. 2в).

При децентралізованому способі з'єднання відображається за допомогою запису номера і типу попереднього пристрою на місце змінної "адреса зв'язку" (Nзв) подальшого влаштування (це стрілками показано на малюнку). Для надійності тип і номер останнього пристрою записують на адресу зв'язку першого пристрою (замикають "кільце").

При централізованому способі створюється нове віртуальне пристрій - "процес", в якому записують в змінні адреси зв'язку - номери і тип учасників з'єднання

Об'єкт "з'єднання" може також мати свої атрибути (наприклад, категорію з'єднання), які тут не вказані.

Сервери

Згідно концепції архітектури розподілених обчислювальних ресурсів (однією з яких є CORBA) об'єкти виступають у вигляді клієнтів, обслуговування яких здійснюється серверами. Проблема полягає в тому, що за запитом клієнта повинні бути викликані тільки певні функції і в певному порядку. Для цього в розглянутому методі пропонується таблична запис функції (універсальна програма).

Універсальні програми - це програми, які в деякому сенсі реалізують всі програми. Спочатку існування універсальної програми здається неправдоподібним. Проте неважко переконатися, що вона існує. Суть полягає в тому, що універсальна програма не повинна обов'язково містити в собі всі інші програми. Вона повинна вміти кодувати і декодувати номери всіх програм, які можуть бути записані і допустимі на заданій мові програмування ".

Програми - послідовність із заданих команд. Кількість таких послідовностей велика, але не нескінченна і може бути перерахованою. Вона піддається закономірному опису у вигляді їх номерів.

## 4. Автоматний підхід і табличні функції

Автоматний підхід дозволяє впорядкувати роботу з конкретної послідовності викликом серверів за запитом об'єкта. Крім того, треба зауважити, що згідно з рекомендаціями запропоновано мову специфікацій і описів SDL (Specification and Description Language) для подання алгоритмів.

Основу мови SDL складає концепція взаємодії кінцевих автоматів. Динамічне поводження системи описується за допомогою механізмів функціонування розширених кінцевих автоматів і зв'язків між ними, що називаються процесами. Набори процесів утворять блоки. Блоки, з'єднані один з одним і зі своїм оточенням каналами, у свою чергу, утворять SDL-систему.

Відповідно до пропонованої SDL-методології опис системи передбачає такі кроки:

• визначення меж SDL-системи;

• визначення каналів SDL-системи і переданих по цих каналах сигналів;

• розбивання системи на SDL-блоки;

• розбивання SDL-блоків на взаємодіючі процеси;

• визначення вхідних і вихідних сигналів, станів і внутрішніх переходів для кожного з SDL-процесів;

• складання SDL-діаграм процесів.

Кожен блок у діаграмі SDL-системи може бути надалі розділений або на блоки, або на набір процесів. Процес описує поводження в SDL і є найбільш важливим об'єктом у мові. Поводження кожного процессу визначається розширеним кінцевим автоматом, що виконує дії і генерує реакції (сигнали) у відповідь на зовнішні дискретні впливи (сигнали).

Такий автомат має кінцеве число внутрішніх станів і оперує з кінцевою дискретною безліччю входів і виходів. Під автоматом з кінцевим числом станів розуміється об'єкт, що знаходиться в одному з дискретних станів S1, S2, ..., Sn, на вхід якого надходять ззовні деякі сигнали I1, I2, ..., Im, а на виході якого є набір вихідних сигналів J1, J2, ..., Jk. Під впливом вхідних сигналів автомат переходить з одного стану в інший, який може збігатися з попереднім, і видає вихідний сигнал. При цьому для кожного стану S і для кожного вхідного сигналу I однозначно відомо, у який стан S перейде автомат і який вихідний сигнал J він при цьому видасть.

На відміну від класичного кінцевого автомата розширений кінцевий автомат допускає можливість переходу ненульової тривалості і визначає механізм простої черги (FIFO) для сигналів, що надходять в автомат у той момент, коли він виконує деякий перехід. Сигнали розглядаються по одному в кожен момент часу в порядку їхнього надходження.

Отже, процес у SDL-специфікації має кінцеве число станів, у кожному з яких він може приймати ряд відправлених цьому процесові припустимих сигналів (від інших процесів або від таймера). Процес може знаходитися в одному зі станів або в переході між станами. Якщо під час переходу надходить сигнал, призначений для даного процесу, то він ставиться в чергу до процесу.

Дії, які виконуються під час переходу, можуть полягати в перетворенні даних, у посилці сигналів у напрямку до інших процесів і т.д. Сигнали

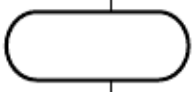
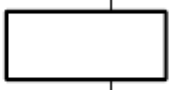
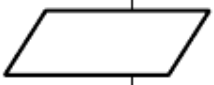
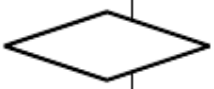
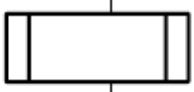
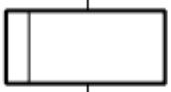
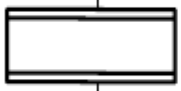
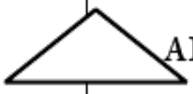
можуть містити інформацію, що визначається на основі даних процесу, що посилає сигнал, і використовується процесом-одержувачем разом з тією інформацією, який розташовує сам цей процес. Крім процесів, що утримуються в розглянутій системі, сигнали можуть також направлятися за межі системи в зовнішнє середовище, а також надходити з зовнішнього середовища. Під зовнішнім середовищем розуміється все, що знаходиться поза SDL-системою.

Посилка й одержання сигналів, передача з їхньою допомогою інформації від одного процесу до іншого, обробка і використання цієї інформації і визначають сценарій функціонування SDL-системи. Передбачається, що після виконання заданого сценарію має бути досягнутий певний результат у поводженні специфікованої системи, зокрема, протоколу сигналізації.

Як правило, очікуваний результат полягає в тому, що у відповідь на ряд сигналів, що надходять із зовнішнього середовища (наприклад, кінцевого станційного комплекту з'єднуваної лінії), система має зробити певні дії, що закінчуються передачею повідомлень у зовнішнє середовище (у цей же станційний комплект з'єднуваної лінії і/або в інший програмний процесс керування посилкою тональних сигналів, у процес запиту інформації АОН і т.ін.).

**Графічні символи SDL наведені у першому стовпчику таблиці 4.1.**

Таблиця 4.1. – Символи SDL

| Графічний SDL | Програмоподібний SDL | Значення символів |
|---|---|---|
|  | STATENEXT STATE | Стан, наступний стан |
|  | TASK | Задача |
|  | INPUT | Введення |
|  | OUTPUT | Виведення |
|  | SAVE | Зберігання |
|  | DECISION | Розв'язання |
|  | CALL | Виклик процедури |
|  | MACRO | Виклик макро |
|  | CREATE | Запрос створення процесу |
|  | ALTERNATIVE | Опції |

Поруч поміщені відповідним цим графічним символам поняття і їхнього позначення в програмоподібній версії SDL. У мові присутні такі символи: введення, виведення, рішення, опція, процес, старт, задача, стан, коннектор, зупинник, збереження. Під час з'єднання символів у діаграмі необхідно дотримуватись певних правил з'єднання. Ці правила такі:

• за символом стану може випливати тільки символ введення або символи введення і збереження;

• символ введення (збереження) може випливати тільки за символом стану;

• за символом введення може випливати кожний (один) символ, крім введення і збереження;

• за символом задачі або виведення випливає кожний (один) символ, крім введення або збереження;

• за символом рішення випливає n (n>=2) символів, що можуть бути якими завгодно, крім символів уведення, збереження;

• за символом збереження нічого не випливає.

**Питання для контролю.**

1. Дайте визначення поняття «система програмування». Що входить в систему програмування?
2. Наведіть класифікацію мов програмування.
3. Перерахуйте основні поняття та властивості об’єктно - орієнтованого підходу до програмування.
4. Які основні особливості архітектури CORBA?
5. Які характерні особливості має табличний метод?
6. В чому суть мови специфікацій та описів SDL?
7. Перерахуйте кроки, які передбачає SDL - методологія для опису системи.