Лекція №9 ПОіП «МЕТОДИ ПРОЕКТУВАННЯ ПРОГРАМНИХ СТРУКТУР» (Модуль 1 -)

План лекції.

Основні причини, що викликають

необхідність стандартизації програмування

На стадії внутрішнього проектування закладаються не тільки технічні характеристики програмного виробу, а й визначається зміст і характер робіт на решті стадій проектування. Рішення, прийняті на стадії внутрішнього проекту, визначають простоту чи складність майбутнього супроводження програми. Легкість супроводження — це одна з властивостей програми, яку не можна додати до неї після її розробки. Раціональна технологія проектування повинна забезпечити зниження загальних трудових витрат із урахуванням всього життєвого циклу програмного виробу. Необхідно враховувати, що супроводження програм коштує набагато більше, ніж їх розробка.

Один із шляхів удосконалення технології: введення стандартів, що обмежують програміста у виборі засобів розробки.

Основні причини, що викликають необхідність стандартизації програмування:

- прагнення зробити систему достатньо простою, доступною для сприйняття програмістом, який знайомий з відповідними стандартами;
- вимога зробити систему легко модифікованою;
- необхідність спрощення налагодження програм;
- необхідність підвищення якості програм;
- можливість планування робіт зі створення програмного виробу і підвищення ефективності контролю із забезпечення якості програм.

Різноманітність інформаційних технологій, апаратних засобів, мов програмування, їх динамічний розвиток, мільйонна армія їх виробників та користувачів зробили однією з головних у комп’ютерному виробництві проблему його стандартизації.

Стандартами програмування називають вимоги, які висуваються до розробника оточенням майбутнього виробу. Ці вимоги можуть диктувати (зумовлювати) керівник, умови контракту, колектив програмістів, галузеві особливості використання, сумісність з іншими компонентами інформаційних систем тощо. Ці вимоги можуть бути занадто високими і складними для виконання. Їх нав’язують

програмістові-розробнику з метою підвищення якості програмування і покращання контролю. Дотримання стандартів полегшує супроводження програм, оскільки вони більш зрозумілі та зручні. Щоб стандарти програмування не використовувались виробниками у власних корпоративних інтересах, потрібно встановити перелік стандартів для їх розробників:

1. Не встановлюй нові механічні умови, якщо вони не поліпшать результати.
2. Роби специфікацію невеликого обсягу, щоб краще запам'ятовувалось.
3. Узгоджуй свої стандарти зі стандартами мови та систем оброблення інформації.
4. Визнач, як можна обминути стандарти.

Введення стандартів на розробку програмних виробів дозволяє спростити процес розробки, експлуатації, супроводження програм, полегшить читання та розуміння програм, поліпшить їх якісні та вартісні показники.

Важливу роль відіграють:

Стандартизація окремих компонентів програми, техніки програмування, каталогізація типових проектних рішень у конкретних умовах: мова програмування, операційна система, методика налагодження і тестування програм та особливості самої задачі. Це дає змогу зняти певну частину уваги програміста, зменшити стомлюваність, уникнути технічних помилок, підвищити продуктивність праці і взагалі автоматизувати працю програміста на базі підтримуючих інструментальних і програмних засобів при підготовці вхідних даних, зберіганні та обміні інформацією, побудові блок-схем.

Стандартизація та уніфікація процесу програмування безпосередньо пов'язані з питаннями ефективної організації праці, її контролю та регламентації.

Типові сучасні стандарти містять угоди, обмеження складності, конструктивні обмеження тощо. Для кожної розроблюваної системи створюють **довідник стандартів** (який може мати обсяг до кількох десятків сторінок), до якого обов'язково увійдуть такі стандарти:

1. Глобальні бази даних слід розбити на блоки, щоб звести до мінімуму кількість модулів, які потребують доступу до блока.
2. Імена змінних мають бути однозначними.
3. Запити вводу/виводу мають оброблятися централізовано робочим модулем.
4. Програма має містити коментарі, написані мовою проектування.
5. Для кожного модуля програми слід завести папку його розробки і супроводження, яка містить:
   - титульну сторінку;
   - журнал вимірів;
   - вимоги до модуля;
   - проектування;
   - поточний лістинг модуля;
   - план перевірки;
   - результати перевірки;
   - звіти про помилки;
   - коментарі;
   - зауваження розробника.

6. Назва модуля має відбивати сутність функції та назву сукупності чи блока даних.
7. Слід виділяти елементи структурного програмування.
8. Коментарі обов'язково повинні містити наступне:
   - деталізацію функцій та інтерфейсів кожної підпрограми;
   - опис застосування і сфери існування кожної змінної;
   - пояснення кожної підфункції;
   - пояснення кожної ненадійної або складної програми;
   - глибина вкладання підмодулів повинна бути мінімальною;
   - розмір модулів визначається їх функцією, але він не повинен перевищувати 100 рядків.

До появи Internet провідні виробники самостійно встановлювали стандарти, примушуючи малі компанії створювати додатки, які підтримували ці стандарти (платформи, протоколи, мови): розробники ПЗ для настільних систем користувались інтерфейсами Windows API, розробники для мейн-фреймів — на платформі OS/390 і т. д. Розвиток Internet сприяв колективній розробці стандартів. Найбільш поширені стандарти TCP/IP та HTML.

Основні аспекти, що визначають технологію програмування:

- зміст і порядок виконання окремих технологічних етапів проектування программ (основне питання: яким чином загальна задача має бути поділена на окремі самостійні підзадачі);
- принципи розподілу робіт між членами колективу розробників;
- спосіб об'єднання окремих частин програмного комплексу, що розроблюється, в єдине ціле;
- порядок розробки документації на програму, що створюється.

Методи розробки програмних комплексів

Метод розробки програм — сукупність правил, обмежень, вимог, яких повинен дотримуватись програміст у процесі проектування програми.

До переліку найбільш поширених методів, що спрямовані на використання процедурно-орієнтованих мов програмування, належать:

1)процедурне,
2)модульне,
3)структурне програмування,
4)низхідне проектування.

Процедурне програмування

Процедурне програмування — метод, який регламентує виділення у задачі, що розв'язується, загальних елементів, що реалізуються у вигляді окремих процедур і використовуються потім у формуванні програмного комплексу.

Це історично перший підхід, що з'явився, при якому програміст повинен вирішити, які процедури повинна виконувати програма, а потім вибрати найбільш відповідні алгоритми для їх реалізації. Під процедурою розуміється особливим чином оформлений фрагмент програми, що має власне ім'я. Згадка процедури в тексті програми приводить до її активізації і називається її викликом. Для обміну

інформацією і основною програмою можуть використовуватися один або декілька параметрів виклику.
Процедури можуть викликатися з різних частин програми багато разів. При цьому в процедуру можуть передаватися змінні і параметри-значення, сама процедура може повертати результат роботи в головну програму. Важливою властивістю процедур є використання ними локальних, доступних тільки даній процедурі, змінних, що дозволяє виключити побічні ефекти при роботі процедури. Процедурне програмування може бути використане в більшості сучасних мов програмування, таких як Pascal, Modula, C і так далі

Програма на процедурній мові програмування складається з послідовності операторів (інструкцій), які задають процедуру рішення задачі. Основним є оператор привласнення, який слугує для зміни вмісту областей пам'яті. Концепція пам'яті як сховища значень, вміст якого може оновлюватися операторами програми, є фундаментальною в імперативному програмуванні.

Виконання програми зводиться до послідовного виконання операторів з метою перетворення початкового стану пам'яті, тобто значень початкових даних, в завершальне, тобто в результати. Таким чином, з погляду програміста є програма і пам'ять, причому перша послідовно оновлює вміст останньої.

Процедурна мова програмування надає можливість програмістові визначати кожен крок в процесі рішення задачі. Особливість таких мов програмування полягає в тому, що завдання розбиваються на кроки і вирішуються крок за кроком. Використовуючи процедурну мову, програміст визначає мовні конструкції для виконання послідовності алгоритмічних кроків.

Процедурні мови програмування - Ada (мова загального призначення), Basic (версії починаючи з Quick Basic до появи Visual Basic), Сі, КОБОЛ, Фортран, Модула-2, Pascal, ПЛ/1, Рапіра, REXX.

Знайомство з процедурним програмуванням необхідно розпочати з функціонального програмування. Концепція функції є одним із фундаментальних понять у математиці.

Функція — це закон (правило), за яким кожному елементу з певної сфери (сфери визначення) зіставляє єдиний елемент з іншої сфери (сфери значень). Існує багато способів опису функції (аналітичний, табличний, графічний і т. ін.). Змінна зі сфери визначення є незалежна змінна. Значення функції може залежати від однієї чи кількох змінних, тобто є функції багатьох змінних, але вихідне значення функції завжди єдине. Цей факт є визначальним у програмуванні за допомогою функцій, бо функція є програмою, що сприймає вхідний сигнал (її аргументи) і виробляє вихідний (її результат). Щоб програмувати, необхідно визначити множину основних (базових) функцій і використовувати їх композицію для визначення нових функцій у термінах базових. Для кожного класу задач необхідний свій набір базових (стандартних) функцій.

У сучасних мовах програмування використовуються конструкції, які називають процедурами або підпрограмами, що базуються на понятті «функції», але відрізняються від останніх тим, що не видають результат як значення функції, а

присвоюють його (їх) деяким своїм параметрам. Це означає, що процедура змінює значення не локальної, а глобальної змінної, яка може використовуватись іншими процедурами, отже, така змінна може набувати різних значень залежно від послідовності виконання процедур, які її використовують. Це відіграє певну роль у розподіленій обробці даних і можливій появі побічного ефекту під час виконання програми.

Переваги процедурного програмування: створює передумови для стандартизації робіт, спрощує накопичення бібліотек; збільшує рівень використання раніш розробленого програмного забезпечення; збільшує наочність і концептуальну стрункість програми.

Модульне програмування

Модульне програмування — метод побудови складних програм по ієрархічному принципу на базі невеликих програмних блоків чи модулів. Формально програмний модуль — скінченний набір операторів, що реалізує певний алгоритм. Структурно модуль — це окрема функціонально-завершена програмна одиниця, яка може застосовуватись самостійно або бути частиною програмного комплексу.

Модульне програмування є розвитком і вдосконаленням процедурного програмування і бібліотек спеціальних програм. Основна властивість модульного програмування - стандартизація інтерфейсу між окремими програмними одиницями - це окрема функціонально-закінчена програмна одиниця, яка структурно оформляється стандартним чином по відношенню до компілятора і по відношенню до об'єднання її з іншими аналогічними одиницями і завантаження. Як правило, кожен модуль містить паспорт, в якому вказані всі основні його характеристики: мова програмування, об'єм, вхідні і вихідні змінні, їх формат, обмеження на них, точки входу, параметри настройки і т.д. Об'єм модуля зазвичай не перевищує 1000 команд ЕОМ або операторів мови програмування. Інакше модуль стає громіздким і важким до сприйняття і використання.

Модульне програмування - це мистецтво розбиття завдання на деяке число різних модулів, уміння широко використовувати стандартні модулі шляхом їх параметричної настройки, автоматизація збірки готових модулів з бібліотек, банків модулів.

Основні концепції модульного програмування:

кожен модуль реалізує єдину незалежну функцію;

кожен модуль має єдину точку входу і виходу;

розмір модуля по можливості повинен бути мінімізований;

кожен модуль може бути розроблений і закодований різними членами бригади програмістів і може бути окремо протестований;

вся система побудована з модулів;

модуль не повинен давати побічних ефектів;

кожен модуль не залежить від того, які реалізовані інші модулі.

При такому підході складна система розділяється на декілька частин, що одночасно створюються різними програмістами. Кожен модуль реалізує єдину функцію. Розмір модуля невеликий, тому тестування кероване і може бути проведено ретельно. Після кодування і тестування всіх модулів відбувається їх інтеграція, і тестується вся система.

При супроводі тестується і відладжується тільки той модуль, який погано працює. Очевидні переваги в полегшенні написання і тестування програм, зменшується вартість їх супроводу.

Концепція модульного програмування реалізована у ряді мов, таких як Modula 2, Turbo Pascal 5.0 і вище, C, і ін.

Основні *властивості модулів*: структурна замкненість (наявність однієї точки входу і однієї точки виходу); функціональна незалежність (виконання однієї визначеної функції. Ця функція може бути подана вибором елементарних складових функцій, але кожна з них не є самостійною з урахуванням загального призначення програми).

Процес конструювання у разі модульного програмування являє собою поетапне (згори-донизу) планування програмної системи. Етап кодування та налагодження виконується знизу-вгору: при цьому кодуються всі модулі, що визначались під час планування, автономно налагоджуються; компонується та налагоджується вся система в цілому.

Основні *переваги модульного програмування*:

- ієрархічна декомпозиція алгоритму (поділ на модулі), яка дає змогу порівняно просто зрозуміти функції кожного модуля та всього комплексу в цілому, а також впорядковано розподілити зусилля розробників, регулюючи поділ праці між ними відповідно до рівня їх кваліфікації;
- скорочення витрат на проведення робіт із тестування, налагодження та супроводження, оскільки в модулях невеликого розміру легше зрозуміти логіку програми, організувати перевірку, оцінити час, який потрібен для проведення робіт;
- розгалуження процесів проектування, кодування та налагодження модулів, що дозволяє знизити трудомісткість розробки; прискорити проектування всієї системи за рахунок ступінчатого графіка виконання робіт;
- можливість рівномірно завантажити розробників і технічні засоби, що використовуються;
- можливість контролю за станом і ходом розробки, що сприяє реальній оцінці обсягу роботи, що виконана; а також перерозподілу, якщо є необхідність, зусиль розробників;
- можливість багаторазового застосування окремих підпрограм, що сприяє зменшенню загального обсягу розробки;
- підвищення рівня модифікованості всієї системи в цілому за рахунок функціональної незалежності модулів, тому що додання модулів з новими характеристиками не потребує зміни інших раніше налагоджених і випробуваних програм;
- можливість підвищення рівня програмно-алгоритмічної надійності системи за рахунок повторного виконання ділянки програми, якщо виникли збої в роботі з вини ЕОМ або перекручення даних, що обробляються;
- можливість економного використання оперативної пам’яті за рахунок завантаження модулів в одну і ту ж сферу.

Недоліки модульного програмування: відсутність формального засобу виділення в загальній задачі окремих незалежних частин; висока трудомісткість комплексного налагодження; складність використання техніки логічних схем.

Відмінність в реалізації процедурного програмування від модульного полягає в тому, що модуль не видно програмі. На відміну від стандартних мов процедурного програмування, в модульних мовах зайві модулі просто не прикомпановываются на етапі збірки.

Показники технологічності модульних програм

Один з показників надійності — складність структури програмного виробу, яка обумовлена складністю кожної окремої частини та взаємодією між ними. Мета проектування — такий поділ програми на модулі, за якого можна досягти мінімальної складності програмного виробу. Показники технологічності: структура модульних програм; незалежність модулів.

Структура модульних програм. Типи структур: монолітно-модульна; послідовно-модульна; ієрархічна; ієрархічно-хаотична; модульно-хаотична.

Монолітно-модульна: ядро програми складає достатньо велика монолітна частина, де проводяться основні розрахунки, і під час їх виконання здійснюється послідовне звернення до окремих модулів. Цей тип побудови модульних програм характерний для процедурного програмування. Модулі, що обслуговують головну частину, можна розглядати як нові конструктивні елементи, що поширюють можливості мови програмування.

Послідовно-модульна: центральна частина програми складається з модулів, що виконуються послідовно, які, у свою чергу, звертаються до інших модулів.

Ієрархічна: програма складається з модулів, зв'язки між якими підлягають суворій ієрархії. Принцип ієрархії: кожен модуль може звертатися до модулів, що безпосередньо йому підпорядковані. Поворот управління завжди має здійснюватись до модуля, який його викликав, навіть у тому разі, якщо в модулі, який викликається, виявлено помилку, що стоїть на перешкоді подальшому виконанню.

Ієрархічно-хаотична: структура побудови, в якій ієрархічна підпорядкованість модулів порушується додатковими зв'язками. Як правило, це модулі, що виконують стандартні розрахунки або типові операції над структурами даних.

Модульно-хаотична: наявність горизонтальних зв'язків між модулями порушує принцип ієрархії.

Структура є не допустимою до використання.

Незалежність модулів. Модулі, що складають програмний комплекс, не тільки виконують визначені функції, але й утворюють зв'язки між собою. Між елементами окремого модуля також існують зв'язки, що характеризують його функціональну однорідність. Сила цих зв'язків є мірою незалежності модулів. Існує дві міри модульності: зв'язність, або зчеплення, модулів; міцність модуля, або зчеплення його елементів.

Зв'язність модулів є мірою взаємозв'язку модулів за даними. Зв'язність модулів характеризується як способом передачі даних, так і властивостями цих даних. Слабке зчеплення має перевагу, бо це означає високий рівень незалежності модулів. Незалежні модулі можуть бути модифіковані без змінювання будь-яких інших модулів. Модулі є повністю незалежними, якщо кожен з них не містить жодної інформації про інші (ситуація, що дуже рідко зустрічається). Чим більше інформації про інші модулі використовується в певному модулі, тим менше вони незалежні і тим міцніше їх зв'язність.

Мета проектування — визначення сполученості модулів таким чином, щоб всі дані, що передаються від одного модуля іншому, передавалися у вигляді явних і простих параметрів.

Міцність модуля (зчеплення елементів модуля) визначає такий розподіл програми на модулі, при якому кожен з них виконує по змозі тільки одну функцію.

Висновок: хороша модульна програма повинна мати мінімальну зв'язність модулів, а її модулі — максимальну міцність. Високий рівень міцності і слабка зв'язність сприяють незалежності модулів, оскільки послаблюються їх взаємозв'язок і взаємозалежність.

Рекомендовані міри модульності: для забезпечення зв'язності модулів рекомендується інформаційний обмін через механізм параметрів; для забезпечення міцності елементи повинні бути пов'язані між собою виконанням однієї функції.

Крім внутрішньої міцності та зовнішнього зчеплення рівень незалежності модуля визначається такими факторами: розміром модуля; застосуванням передречених модулів структурою прийняття рішення; мінімізацією доступу до даних; використанням внутрішніх процедур.

Правила формування структури та взаємодії модулів у програмному виробі: структура програмного виробу та правила оформлення опису кожного модуля мають бути уніфіковані; кожен модуль має характеризуватися функціональною завершеністю, автономністю та незалежністю у сукупності модулів, які його використовують і які він викликає; застосування стандартних правил організації зв'язків з іншими модулями з управління та даних; структура програмного виробу повинна бути подана у вигляді сукупності невеликих програмних модулів, що зв'язані ієрархічно (це дає можливість повністю та порівняно просто з'ясувати функції та правила роботи окремих частин і всього програмного виробу в цілому); повинен бути відсутнім ефект післядії чергового виконання програмного модуля на наступні виконання.

Структурне програмування

Одним з перших ідеологів його був професор Э. Декстра (Эйндховен). У 1965 році він висловив думку, що ``оператор GOTO може бути виключений з мов программирования''.

Основна ідея структурного програмування - обмеження допустимих структур, що управляють. для написання будь-якої програми потрібний всього три основних складових блоку.

Структурне програмування — форма програмування, при якій логіка програми може бути подана комбінацією тільки трьох базових структур, до яких належать:

- функціональний елемент чи їх лінійна послідовність(послідовного виконання — одноразове виконання операцій в тому порядку, в якому вони записані в тексті програми);
- розподільна структура (альтернативна чи структура вибору)(галуження — одноразове виконання однієї з двох або більш за операції, залежно від виконання деякої заданої умови);
- циклічна структура(багатократного виконання однієї і тієї ж операції до тих пір, поки виконується деяка задана умова (умова продовження циклу).).

Особливість кожної з цих структур — наявність однієї точки входу і однієї точки виходу. І як наслідок, структури можуть об’єднуватись, вкладатись одна в одну будь-яким чином.

У програмі базові конструкції можуть бути вкладені один в одного довільним чином, але ніяких інших засобів управління послідовністю виконання операцій не передбачається.

Фрагменти програми (або цілісні обчислювальні блоки, що не повторюються, але є логічно), що повторюються, можуть оформлятися у вигляді т.з. підпрограм (процедур або функцій). В цьому випадку в тексті основної програми, замість поміщеного в підпрограму фрагмента, вставляється інструкція виклику підпрограми. При виконанні такої інструкції виконується викликана підпрограма, після чого виконання програми продовжується з інструкції, наступної за командою виклику підпрограми.

Розробка програми ведеться покроково, методом зверху «вниз».

Спочатку пишеться текст основної програми, в якому, замість кожного зв'язкового логічного фрагмента тексту, вставляється виклик підпрограми, яка виконуватиме цей фрагмент. Замість сьогодення, працюючих підпрограм, в програму вставляються «заглушки», які нічого не роблять. Отримана програма перевіряється і відладжується. Після того, як програміст переконається, що підпрограми викликаються в правильній послідовності (тобто загальна структура програми вірна), підпрограми-заглушки послідовно замінюються на що реально працюють, причому розробка кожної підпрограми ведеться тим же методом, що і основної програми. Розробка закінчується тоді, коли не залишиться жодної «затички», яка не була б видалена. Така послідовність гарантує, що на кожному етапі розробки програміст одночасно має справу з осяжним і зрозумілим йому безліччю фрагментів, і може бути упевнений, що загальна структура всіх вищих рівнів програми вірна. При супроводі і внесенні змін в програму з'ясовується, в які саме процедури потрібно внести зміни, і вони вносяться, не зачіпаючи частини програми, безпосередньо не пов'язані з ними. Це дозволяє гарантувати, що при внесенні змін і виправленні помилок не вийде з ладу якась частина програми, що знаходиться в даний момент поза зоною уваги програміста.

Методологія структурного програмування з'явилася як наслідок зростання складності вирішуваних на комп'ютерах задач, і відповідного ускладнення програмного забезпечення. Найбільш сильній критиці з боку розробників структурного підходу до програмування піддався оператор GOTO (оператор безумовного переходу), що був тоді майже у всіх мовах програмування. Неправильне і

необдумане використання довільних переходів в тексті програми приводить до отримання заплутаних, погано структурованих програм (т.н. спагетти-коду), по тексту яких практично неможливо зрозуміти порядок виконання і взаємозалежність фрагментів.

Переваги структурного програмування: потік управління в програмах спрямований згори-донизу, тобто оператори, що становлять програму, будуть виконуватися в тому порядку, як вони з'являються у тексті програми (відсутні стрибки в управлінні, які в більшості випадків ускладнюють логіку програми і приводять до помилок). Завдяки цьому програма стає зрозумілою, підвищується її надійність, полегшується її налагодження та супроводжування.

Низхідне і висхідне проектування

Одна з основних ідей, покладених в більшість відомих технологій програмування - низхідне проектування. Існують також інші назви: «програмування з покроковим вдосконаленням», «систематичне програмування», «ієрархічне програмування». Принцип низхідного проектування - спочатку визначаються основні функції, які повинні бути забезпечені програмою, що виготовляється, а потім довизначаються додаткові функції, витікаючі з основних.

Наприклад:

Основна функція: «обработка файла»

Довизначені функції: «открыть файл» , «обработать все записи», «закрыть файл».

Ось деякі принципи низхідного проектування:

Докладний формальний і строгий опис проектувальником входів, функцій і виходів всіх модулів програм або системи.

Як тільки ви переконаєтеся, що деяка частина завдання може бути реалізована у вигляді окремого модуля, постарайтеся більше не думати про це.

Уважно стежите, щоб Вас не залучили в обговорення неістотних частин проекту.

На кожному рівні проекту намагайтеся записати реалізацію модуля у вигляді символічних кодів або блок схеми (розмір опису в ідеалі не повинен перевершувати одного листа, щоб при подальшому аналізі перед очима знаходилася якнайповніша картина).

Проектуванню структури даних і їх руху слідує не менше часу, чим програмі.

Мови програмування систем ООБД та мови запитів

Необхідність ООБД пов'язана з потребою в певному інтегрованому середовищі створення складних інформаційних систем. У цьому середовищі не повинно бути протирічь між структурною і поведінковою частинами проекту і повинне підтримуватися ефективне керування складними структурами даних у зовнішній пам'яті. З цього погляду мовне середовище ООБД — це об'єктно-орієнтована система програмування, що природньо включає засоби роботи з довгостроковими об'єктами.

Мови програмування ООБД і БД у багатьох своїх рисах різняться тільки термінологічно; істотною відмінністю є лише підтримка в мовах першого класу

підходу до спадкування класів. Крім того, мови другого класу, як правило, більш розвинуті як стосовно системи типів, так і щодо керуючих конструкцій.

Іншим аспектом мовного середовища ООБД є потреба в мовах запитів, які можна було б використовувати в інтерактивному режимі. Якщо доступ до об'єктів зовнішньої БД у мовах програмування ООБД має в основному навігаційний характер, то для мов запитів більш зручний декларативний стиль. Декларативні мови запитів до ООБД менш розвинуті, ніж мови програмування ООБД, і під час їх реалізації виникають істотні проблеми.

Наближеність об'єктно-орієнтованого і функціонального підходів до програмування дозволяє досить вдало спиратися на функціональні мови програмування. Зокрема, мова Лісп є й інструментальною мовою, і базою об'єктно-орієнтованої мови програмування.

Контрольні питання

1. Чим зумовлюється необхідність стандартизації у програмуванні?
2. Основні аспекти, які визначають повноту технології програмування.
3. Поняття методу проектування.
4. Переваги та недоліки процедурного програмування.
5. Основні властивості модуля.
6. Переваги модульного програмування.
7. Чому метод модульного програмування не забезпечує потрібного рівня модифікованості програмних комплексів?
8. Основні труднощі застосування методів програмування «знизу-вгору».
9. Припустимі структури модульних програм.
10. Засоби зменшення складності програмних комплексів.
11. Яка відмітна особливість базових структур у структурному програмуванні?