# ТРЕНІНГ-КУРС З ІНФОРМАЦІЙНОГО УПРАВЛІННЯ ПІДПРИЄМСТВАМИ ТА ПРОЄКТАМИ

## *Самостійна робота за змістовним модулем 4*

### 4.1. 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 [21]:

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 програм (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.

### 4.2. 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: процедурне, модульне, структурне програмування, низхідне проектування.

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.

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 – 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.

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.

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.