ТРЕНІНГ-КУРС З ІНФОРМАЦІЙНОГО УПРАВЛІННЯ ПІДПРИЄМСТВАМИ ТА ПРОЄКТАМИ

Самостійна робота за змістовним модулем 7

7.1. Форми програмування з використанням пакетів

Застосування ППП пов'язане з визначеними вимогами до знань користувачів в галузі програмування. Ці вимоги відмінні для різних пакетів і відмінна складність функцій користувачів, що виконуються в разі використання пакета. Таким чином, ППП забезпечують різний ступінь автоматизації програмування, який визначається рівнем вхідної мови. Чи означає це, що чим вищий ступінь автоматизації програмування, тим краще? Ні, тому що у процесі розробки ППП має бути забезпечений такий ступінь автоматизації програмування, який відповідає вимогам його використання. При оцінці потрібного ступеня автоматизації під час розробки пакета необхідно враховувати, з одного боку, кваліфікацію користувачів, для яких він передбачений, а з іншого – той факт, що розробка програми з більш високим ступенем автоматизації потребує, як правило, більших витрат часу та коштів.

Наявний досвід використання обчислювальної техніки дозволяє виділити п'ять основних груп пакетів з погляду загальних принципів їх експлуатації: системи програмування з використанням спеціалізованих мов; бібліотеки прикладних програм; програмні системи; пакети-моделі; пакети-асистенти.

Системи програмування з використанням спеціалізованих мов. Під час роботи з цими системами користувач складає збільшений алгоритм розв'язування задачі мовою, яка відповідає проблемній орієнтації задачі. Підвищення рівня автоматизації програмування в цьому випадку здійснюється за рахунок укрупненого описання алгоритму та знання термінології предметної сфери користувачем. Системи програмування з використання спеціалізованих мов виконують переклад програми користувача, яку написано найбільш придатною для даної проблеми й алгоритму мовою, у машинний код. Основна категорія користувачів цих систем – кваліфікований користувач або програміст. Прикладом пакетів цього типу є пакет моделювання дискретних систем.

Бібліотеки прикладних програм. З точки зору структури бібліотеки – це набір окремих програм, кожна з яких має самостійне значення та застосовується для розв'язування, як правило, не дуже складної задачі або реалізації незалежної функції. Звертання до програм пакету виконується з програми користувача, що написана мовою програмування, або з використанням команд операційної системи. Перевагою використання пакетів цього типу є простота розширення їх функціональних можливостей додаванням нових програм. Обмеження їх використання пов'язане з

вимогами до рівня знань основ програмування та принципів функціонування програмного забезпечення. Тому основними користувачами цих пакетів є програмісти. Прикладом пакетів цього типу є бібліотека наукових підпрограм.

Програмні системи. Програмні системи – це сукупність програм, призначених для розв'язування порівняно великої за обсягом задачі. Звертання до програмних систем користувач реалізує за допомогою програми, написаної проблемно-орієнтованою мовою або поданої у вигляді потоку директив. У програмі користувача визначаються характеристики задачі та вимоги щодо її розв'язання. Функції пакета полягають в організації розпізнавання вимог користувача за допомогою спеціальної керуючої програми та звертання до потрібних модулів, які виконують безпосередньо обробку даних. Як правило, різні запити користувачів потребують відмінних алгоритмів їх реалізації і, як наслідок, відмінних сукупностей модулів обробки. Керуюча програма утворює з потрібної для розв'язування даної задачі сукупності модулів обробки послідовність, яку називають робочою програмою. Робоча програма може створюватися як перед виконанням обробки, так і динамічно у ході виконання обробки даних. Створення робочої програми відповідає управлінню пакетом на макрорівні. У зв'язку з тим, що використання пакета не вимагає жорстких вимог до знань користувача щодо принципів алгоритмізації та мов програмування, основна категорія користувачів програмних систем – масовий користувач. Прикладом пакетів цього типу є програмні системи для розв'язання прикладних задач з будь-якої сфері людської діяльності.

Пакети-моделі. Пакети-моделі отримують завдання користувача у вигляді постановки задачі, що описана формалізованою мовою. Спеціальний блок керуючої програми визначає тип моделі, потрібні модулі пакета та послідовність їх виконання для розв'язування поставленої задачі, а також організує звернення до цих модулів із завданням потрібних параметрів для кожного з них. Основна вимога до користувачів – вміння виконати формалізовану постановку своєї задачі, а тому категорія користувачів цього типу пакетів – спеціаліст предметної сфери. Прикладом пакетів цього типу є системи автоматизованого проектування (САПР).

Загальне зауваження: створення програмних систем і пакетів-моделей для розв'язання складних задач є трудомісткий та дорогий процес. Тому їх розробка доцільна в таких випадках: якщо процес розв'язання задачі повинен виконуватися з якихось причин автономно, без участі людини; якщо потрібно використання обчислювальних систем користувачами, що не є спеціалістами в сфері програмування.

Пакети-асистенти. Вхідна мова цих пакетів призначена для задання параметрів і вибору альтернатив, що забезпечують розв'язування задачі. Пакет побудований таким чином, що на кожному етапі роботи користувачу пропонується можливий перелік дій, виходячи з чергового рівня заданих параметрів. Користувач формує розв'язок вибором дій, що пропонуються. Прикладом пакетів цього типу є табличні процесори.

7.2. Архітектура ППП

Під архітектурою ППП розуміють достатньо повне та докладне описання способу взаємодії користувача з пакетом, а також описання загальних принципів побудови та організації пакета.

Взаємодія користувача з віртуальною машиною, що визначається пакетом, полягає у складанні програми або формуванні запиту для розв'язування конкретної задачі. Засобом взаємодії є вхідна мова пакета. Рівень вхідної мови значною мірою зумовлює складність і функції пакета. Спосіб взаємодії користувача з пакетом виражається також у режимі його використання: пакетний (або командний), діалоговий. У разі діалогового режиму простота взаємодії багато в чому залежить від організації ведення діалогу, тобто від форми спілкування і від сервісних функцій, що надаються пакетом для спрощення цього процесу.

Таким чином, спосіб взаємодії користувача з пакетом визначається: рівнем вхідної мови; режимом використання пакета; організацією процесу спілкування.

Поняття загальних принципів побудови ППП включає: структуру пакета та спосіб взаємодії між структурними частинами пакета. За структурою ППП поділяються на два класи: ППП простої та ППП складної структури.

Пакети простої структури містять модулі, що реалізують функціональні задачі з предметної сфери (модулі обробки даних). Їх особливість полягає в тому, що це модулі одного рівня підпорядкованості. Різноманітність пакетів простої структури визначається способом звернення до модулів пакета: ППП як набір модулів, звернення до яких здійснюється з програми користувача, що написана мовою програмування (обмін даними між модулями виконується із застосуванням механізму параметрів); ППП як набір модулів, звернення до яких здійснюється командною мовою (обмін даними виконується через зовнішні накопичувачі); ППП як набір взаємопов'язаних модулів, в якому кожний наступний модуль викликається попереднім (це приклад обов'язкового виклику модулів без втручання з боку користувача).

Пакети складної структури. Відмінна особливість цього типу пакетів – наявність спеціальної програми, яка виконує функції керування і називається керуючою програмою.

Функції керуючих і обслуговуючих модулів пакета

Пакет є об'єднання керуючих, обслуговуючих і обробних модулів. Функція обробних модулів полягає в реалізації кроків алгоритму перетворення значень вхідних даних у значення вихідних даних. Аналіз моделі предметної сфери і зовнішнього керування пакетом дає змогу уточнити функції керуючих і обслуговуючих модулів, тобто системного наповнення пакета.

У загальному випадку поділ на керуючі й обслуговуючі модулі доволі умовний.

Для налагодження ППП на конкретну предметну сферу необхідно завантажити в оболонку пакета опис інформаційної бази пакета, опис функціональних зв'язків і зв'язків за визначенням, а також підключити обробні модулі.

Програмна реалізація оболонки пакета припускає певну форму внутрішнього представлення інформації про множину форм, що складають модель предметної сфери. Обробні модулі мають програмуватися відповідно до вимог їхнього зв'язку із системним наповненням пакета з керування і за даними. Якщо перелічені умови виконуються, можлива розробка програмних засобів генерації ППП для різних предметних сфер, що використовують ту саму оболонку пакета.

Доступ до баз даних за допомогою интерфейсу ODBC

Відкритий інтерфейс зв'язку з базами даних (Open Database Connectivity, ODBC) являє собою незалежну від типу баз даних технологію, призначену для організації взаємодії з реляційними СКБД. Цей інтерфейс, який є буфером між програмою користувача і базою даних, має велике значення, оскільки він є стандартним засобом роботи з базами даних. Дозволяючи працювати з базами даних будь-якого типу, для яких в системі є драйвер, він дає можливість звертатися до широкого спектра даних, не вимагаючи від користувачів знань про формати та особливості конкретних баз даних. У цьому сенсі ODBC – загальний засіб доступу до значної кількості систем управління базами даних. Для побудови запитів до баз даних через джерела даних ODBC використовується мова SQL.

Усі бази даних, доступ до яких виконується за допомогою ODBC, мусять мати так зване джерело даних ODBC, яке містить Ім'я джерела даних (DSN – Data Source Name), інформацію про тип бази даних і драйвер, використовуваний для доступу до неї. Залежно від драйвера може виникнути потреба у додатковій інформації. Наприклад, під час роботи з драйвером SQL Server необхідно повідомити ім'я комп'ютера, на якому розташовано сервер бази даних, а також ім'я бази даних.

Джерело даних ODBC може міститись або в Реєстрі Windows, або у файлі DSN. Для визначення джерела даних ODBC і розміщенні його в Реєстрі Windows використовується Адміністратор джерел даних ODBC (ODBC Data Surce Administrator), який знаходиться в Панелі управління (Control Panel). У вікні адміністратора можна як визначити нове джерело, так і модифікувати наявне. Крім того, джерела даних можуть бути створені і програмно, наприклад, із використанням DAO.

Протокол ODBC, починаючи з версії 3.0, підтримує три типи джерел: користувацькі, системні та файлові (User, System, File). До джерела першого типу має доступ лише той користувач, який його створив, і лише з того комп'ютера, на якому його було визначено. До системного джерела мають доступ усі користувачі даного комп'ютера. До файлового джерела мають

доступ користувачі всієї мережі, якщо тільки на їхніх комп'ютерах встановлено потрібні драйвери ODBC.

Драйвери ODBC для окремих СУБД являють собою проміжний прошарок, призначений для перетворення операторів мовою SQL на той формат, який вимагає конкретна система управління базами даних. Драйвери – саме ті компоненти ODBC, що забезпечують незалежність від типу бази даних. Кожний драйвер відповідає за відображення загальної функціональності SQL у функціональність конкретної СУБД.

Трансакції

Під трансакцією розуміється група з одного чи декількох операторів SQL, які виконуються як одне ціле. Трансакції мають вирішальне значення для забезпечення цілісності даних. Протягом часу в середовищі ODBC допускається тільки одна трансакція. Завершення трансакції робить зміни в базі даних постійними, при цьому база даних вже не може бути приведена до початкового стану. Виконання відкату можливе тільки при перериванні трансакції. Слід зазначити, що не всі джерела даних і драйвери підтримують трансакції, чи містять обмеження на оператори, що входять у трансакцію. Існують два режими виконання трансакцій у ODBC – автоматичний та режим ручного виконання. У першому режимі драйвер починає і завершує трансакцію після виконання оператора SQL. В ручному режимі трансакція починається автоматично при запуску операторів SQL, але завершується тільки, якщо прикладна програма сама їх завершить. При цьому оператор SQL є частиною трансакції. Більшість драйверів, що припускають трансакції, підтримують обидва режими.

7.3. Етапи технологічного процесу використання ППП

Послідовність технологічних операцій, з яких складається процес використання пакета, можна розділити на три етапи:

1) передмашинний етап (включає постановку задачі, збір вихідних даних, їхню формалізацію, підготовку первинного носія й ін.);

2) машинний етап включає всі операції, що припускають використання ЕОМ;

3) післямашинний етап (аналіз отриманих результатів, ухвалення рішення про повторні розрахунки).

Машинний етап включає дві фази: підготовчу, на якій виконуються дії з підготовки пакета безпосередньо до розв'язання задачі, і виконавчу, на якій пакет розв'язує задачу користувача. Зміст виконавчої фази є досить однотипним й у багатьох пакетів складної структури різниться несуттєвими деталями.

Зміст же підготовчої фази (склад технологічних операцій і послідовність їхнього виконання) залежить від реалізованої в пакеті глибини керування пакетом.

Глибина керування визначає структурні одиниці пакета, на рівні яких здійснюється вибір однієї чи декількох можливостей використання пакета.

Допустимість реалізації заданої глибини керування визначається ступенем завершеності програм пакета. Під ступенем завершеності тут розуміється форма подання програм (вихідний, об'єктний чи абсолютний модуль). Природно, передбачається, що настройка всіх програм довершена незалежно від форми їхнього подання. У зв'язку з цим можна використовувати три рівні керування: мікрокерування, макрокерування і зовнішнє керування.

Мікрокерування полягає в зміні окремих стандартних конструкцій модулів пакета. Типові об'єкти мікрокерування – це дескриптори і режими файлів, окремі програмні константи (наприклад, розміри оголошених масивів і т. ін.). У деяких випадках об'єктами мікрокерування можуть бути окремі оператори вихідного модуля.

Макрокерування здійснюється на рівні модульної структури пакета. Типовий приклад макрокерування – генерація заданої конфігурації пакета.

Зовнішнє керування здійснюється за допомогою вхідної мови пакета, що задає ланцюжок виконуваних модулів. При цьому в ланцюжок можуть включатися лише модулі, уведені до складу даної конфігурації на рівні макрокерування; алгоритм, реалізований кожним модулем, у свою чергу, визначається результатами мікрокерування.

Перераховані рівні утворять ієрархію: зовнішнє керування є обов'язковим елементом при використанні пакета складної структури, макрокерування – дуже розповсюдженим елементом; засоби мікрокерування доцільно включати лише в досить розвинуті пакети.

Мікрокерування є найбільш простим і практично без обмежень може бути реалізованим при збереженні пакета на рівні вихідного модуля.

Отже, кожен модуль, у стандарти якого вносяться зміни, підлягає трансляції, а це може помітно підвищити вартість розв'язування задачі пакетом. Ряд операцій мікрокерування може бути виконаний шляхом внесення змін у текст абсолютного (завантажувального) модуля. Реалізація цього способу вимагає найменших витрат, оскільки не потрібні трансляція і редагування, однак при цьому виникає ряд обмежень, пов'язаних з характером змін, що вводяться в стандарти пакета. Так, зазвичай не вдається збільшити розмір запису файла, значно складніше реалізується зміна окремих операторів (хоча така необхідність виникає значно рідше). Дії, пов'язані зі зміною стандартів файлів, простіше зважуються на рівні об'єктних модулів. Реалізація мікрокерування на рівні об'єктних або абсолютних модулів вимагає ґрунтовного знання структури створюваних транслятором таблиць: часто єдиним способом одержання цих даних є експеримент, що значно утруднює реалізацію.

Макрокерування реалізується шляхом генерації заданого викликаючого об'єктного модуля з наступною каталогізацією його в бібліотеці об'єктних модулів (БОМ).

Механізм зовнішнього керування здійснюється шляхом настроювання блоку керування на підставі переданої йому транслятором із вхідною мовою керуючої інформації.