Тема 1. Сучасні технології програмування та C++Builder

1. Об’єктно-орієнтоване програмування (ООП)

2. Візуальне програмування інтерфейсів програмного забезпечення

3. Технологія швидкої розробки додатків (RAD)

4. Узагальнене програмування (generic programming)

Borland C++Builder – це одна з найпотужніших систем, що дозволяють на найсучаснішому рівні створювати як окремі прикладні програми Windows, так і розгалужені комплекси, призначені для роботи в корпоративних мережах і в Інтернет [1].

Серед нових можливостей C++Builder 6 можна відзначити крос-платформенні додатки, технології доступу до даних ADO, dbExpress, InterBaseExpress, компоненти-сервери COM, технології розподілених додатків COM, CORBA, MIDAS, нову методику диспетчеризації дій, програмування обміну інформацією по каналах зв'язку з використанням протоколів WebBroker, InternetExpress і XML, програмування Web-додатків.

Нова версія продукту C++Builder 2007 поєднує підтримку операційної системи Windows Vista, технологій Web 2.0, включаючи AJAX, із найновішими стандартами ANSI C++, включаючи нову бібліотеку підтримки Dinkumware. Продуктивність продукту зросла в п'ять разів у порівнянні з попередніми версіями. Розробникам запропоновані додаткові можливості візуалізації програмного коду й інтегровані засоби модульного тестування коду на C++. C++Builder поєднує у собі зручності візуального середовища розробки, об’єктно-орієнтований підхід, різноманітні можливості повторного використання коду, відкриту архітектуру й високопродуктивні компілятори мов Object Pascal і C++, які є одними із самих популярних мов програмування, а також масштабований доступ до даних, що зберігається в різних СКБД, як настільних, так і серверних.

Серед всіх сучасних технологій, які підтримує C++Builder, розглянемо коротко тільки ті технології програмування, які необхідно застосовувати при виконанні запропонованих у посібнику лабораторних робіт.

1. Об’єктно-орієнтоване програмування (ООП)

Об’єктно-орієнтоване програмування – це методика, що концентрує основну увагу програміста на зв'язках між об'єктами, а не на деталях їхньої реалізації.

У середовищі C++Builder класичні принципи ООП [11-13, 23, 30] (інкапсуляція, успадкування, поліморфізм, створення класів і об'єктів) інтерпретуються й доповнюються новими поняттями й термінологією, прийнятими інтегрованим середовищем візуальної розробки (IDE).

Мова C++ у середовищі C++Builder розширена новими можливостями (компоненти, властивості, методи, обробники подій) і останніми доповненнями стандарту ANSI C++ (шаблони, простори імен, явні й непостійні оголошення, ідентифікація типів при виконанні програми RTTI, виключення, динамічні масиви, ключові словами explicit, mutable, typename, automated і ін.). У C++Builder введено нові типи даних: булевий тип даних bool, рядковий тип AnsiString (реалізований як клас, оголошений у заголовочному файлі vcl/dstring.h), тип “множина” (реалізований як шаблон класу, оголошений у заголовочному файлі vcl/sysdefs.h) та інші типи.

Вважається [1, 2], що останні версії C++Builder найбільш повно відповідають стандарту ANSI/ISO серед всіх компіляторів на платформі Windows.

Компанія Borland пропонує потужний набір власних розширень мови C++, що забезпечує підтримку VCL бібліотеки (Visual Component Library). У C++Builder поряд із звичайними об'єктними класами мови C++, з'явилися нові компонентні класи (компоненти).

Форми є основою додатків C++Builder. Створення користуваць-кого інтерфейсу додатка полягає в додаванні на форму елементів (компонентів). Компоненти C++Builder розташовуються на палітрі компонентів, виконаної у вигляді багатосторінкового блокнота. Важлива особливість C++Builder полягає в тому, що він дозволяє створювати власні компоненти й настроювати палітру компонентів, а також створювати різні версії палітри компонентів для різних проектів.

Компонент (component) – спеціальний клас, властивості якого подають атрибути об'єкту, а його методи реалізують операції над відповідними екземплярами компонентних класів. Поняття властивість, метод, обробник подій розглянемо далі, а зараз перелічимо основні відмінності компонентних класів від об'єктних класів [3, 27]:

- всі компоненти є прямими або непрямими нащадками одного загального класу-прародича (TComponent);
- компоненти звичайно використовуються безпосередньо, шляхом маніпуляції з їхніми властивостями;
- компоненти розміщаються тільки в динамічній пам'яті купи (heap) за допомогою оператора new, а не на стеку, як об'єкти звичайних класів;
- компоненти можна добавляти до палітри компонентів і далі маніпулювати з ними за допомогою редактора форм інтегрованого середовища візуальної розробки C++Builder;
- оголошення (інтерфейсна частина) компонентного класу обов'язково описується окремо (розширення .h) від реалізації (розширення .cpp).

Оголошення компонентного класу має вигляд:

class className : [специфікатор доступу:] parentClass

{

<0бява дружніх класів>

private: //внутрішні деталі реалізації

< приватні члени-дані,конструктори, методи>

protected: //інтерфейс розробника

<захищені члени-дані, конструктори, методи>

public: // run-time інтерфейс

<загальнодоступні властивості, члени-дані, конструктори, деструктор, методи>

__published: //design-time інтерфейс

<загальновідомі властивості, члени-дані>

<Об'ява дружніх функцій>

};

Зазначимо, що parentClass – ім’я базового класу – обов’язково присутнє, оскільки будь-який компонент є прямим або непрямим нащадком класу TComponent.

Нова секція з ключовим словом __published – це доповнення, яке C++Builder вводить у стандарт ANSI C++ для оголошення загальновідомих елементів компонентних класів. Ця секція відрізняється від загальнодоступної тільки тем, що компілятор генерує інформацію про властивості, члени-дані та методи компонента для інспектора об’єктів.

Форма – це також компонент, і тому з кожною формою пов’язані файл оголошення та файл реалізації. Скелети обох файлів автоматично генеруються середовищем C++Builder при візуальному створенні нової форми, а програміст доповнює ці файли власним кодом. При додаванні у форму будь-якого компонента з палітри компонентів C++Builder автоматично формує програмний код для створення об'єкта (змінної) даного типу. Змінна додається як член класу даної форми.

2. Візуальне програмування інтерфейсів програмного забезпечення

Візуалізація – це процес графічного відображення складних процесів (у цьому випадку побудови) на екрані комп'ютера у вигляді графічних примітивів (графічних фігур) [24]. Візуалізовувати можна будь-які процеси: управління, побудови, малювання та інші. Приклад найпростішого варіанта візуалізації – лінійка прогресу (прямокутник, відсоток заповнення якого прямо пропорційний об'єму виконання якої-небудь операції). Дивлячись на лінійку, можна оцінити об'єм невиконаних операцій, який залишився.

Візуалізовувати можна інтерфейси програмного забезпечення. Це дозволяє спростити взаємодію програмного продукту з користувачем. Зображення на елементах інтерфейсу (зовнішнього вигляду програмного забезпечення) дозволяють користувачу інтуїтивно розбиратися в призначенні цих елементів.

Для візуалізації інтерфейсів програмного забезпечення існує цілий ряд спеціально розроблених елементів інтерфейсу – візуальних компонентів, що дозволяють відображати різну інформацію й здійснювати керування програмою в цілому. Найпростіший приклад – візуальна кнопка на екрані комп'ютера. Візуальна кнопка імітує поведінку звичайної кнопки на пульті керування будь-якого приладу. Її можна "натискати" як справжню.

Саме наявність візуальних засобів побудови інтерфейсів для Windows в C++Builder, а також створюване ними візуальне програмне забезпечення закріпили за ним термін "***візуальне програмування***".

Визначальними елементом процесу візуалізації є візуалізована модель, тобто модель, що піддається відображенню з метою можливості зміни її структури або її параметрів (або параметрів її окремих частин). Візуалізованою моделлю в C++Builder є вікно (форма, діалог) Windows, а не код програми. У C++Builder прийнято візуалізовувати тільки роботу з елементами інтерфейсу, коли об'єкти візуалізації розглядаються як візуальні компоненти, з яких складаються форми (вікна й діалоги) інтерфейсу програми.

Інструменти візуальної розробки в середовище C++Builder включають в себе дизайнер форм, інспектор об'єктів, палітру компонентів (вікно, що містить набір компонентів, з яких будується візуальна модель), менеджер проектів і редактор коду.

Ці інструменти включені в інтегроване середовище системи (Integrated Development Environment, IDE), яке забезпечує продуктивність багаторазового використання візуальних компонентів у поєднанні з удосконаленими інструментами й засобами доступу до баз даних.

Конструювання способом “перетягування” (drag-and-drop) дозволяє створювати додаток простим перетаскуванням захоплених мишею візуальних компонентів з палітри на форму додатка.

Механізми двунапрямної розробки (Two-Way-Tools) забезпечують контроль коду за допомогою гнучкої, інтегрованої й синхронізованої взаємодії між інструментами візуального проектування й редактором коду.

Інспектор об'єктів надає можливість оперувати властивостями (вікно властивостей – вікно, у якому відображаються параметри обраного елемента візуальної моделі) й подіями компонентів.

Компоненти можуть бути ***візуальні,*** видимі при роботі додатку, і ***невізуальні,*** виконуючі ті або інші службові функції. Візуальні компоненти відразу видні на екрані у процесі проектування у такому ж вигляді, в якому їх побачить користувач під час виконання додатку. Це дозволяє дуже легко вибрати місце їхнього розташування та їхній дизайн – форму, розмір, оформлення, текст, колір та інше. Невізуальні компоненти видимі на формі лише в процесі проектування у вигляді піктограм, але для користувача під час виконання вони не видимі, хоча й виконують для нього за кадром корисну роботу.

Візуальне програмування дозволяє звести проектування користувацького інтерфейсу до простих і наочних процедур. Але переваги візуального програмування не зводяться лише до цього.

Саме головне полягає в тому, що під час проектування форми й розміщення на ній компонентів C++Builder автоматично формує коди програми, включаючи в неї відповідні фрагменти, що описують даний компонент. А потім у відповідних діалогових вікнах користувач може змінити задані за замовчуванням значення властивостей цих компонентів і, при необхідності, написати обробники потрібних подій. Тобто проектування зводиться, фактично, до розміщення компонентів на формі, завдання деяких їхніх властивостей і написання, при необхідності, обробників подій. Результатом візуального проектування є скелет майбутньої програми, у яку вже внесені відповідні коди.

3. Технологія швидкої розробки додатків (RAD)

Завдяки візуальному об’єктно-орієнтованому програмуванню (ООП) була створена технологія, що одержала назву ***швидка розробка додатків*** *(RAD* — Rapid Application Development). Ця технологія характерна для нового покоління систем програмування, до якого відноситься й C++Builder [6, 7, 29].

C++Builder – це інструмент для швидкої розробки додатків (RAD) на C++ під Windows, який підтримує можливість програмування, що ґрунтується на компонентах.

Компоненти – це будівельні блоки для додатків. Тобто, використовуються об’єкти-компоненти зі своїми можливостями і об’єднуються в один додаток. C++Builder сам побудований на компонентах і робота з компонентами в C++Builder проста і надійна.

Швидка розробка додатків (RAD) означає підтримку властивостей, методів і подій компонентів у рамках об'єктно-орієнтовного програмування.

Властивості дозволяють легко встановлювати різноманітні характеристики компонентів, такі як назви, контекстні підказки або джерела даних. Методи (функції-члени) роблять певні операції над компонентним об'єктом. Події зв'язують впливи користувача на компоненти із кодами реакції на ці впливи.

Працюючи спільно, властивості, методи і події утворять середовище RAD інтуїтивного програмування надійних додатків для Windows [3].

Як було сказано вище, компонент – це спеціальний клас, властивості якого подають атрибути об'єкту, а його методи реалізують операції над відповідними екземплярами компонентних класів.

Якщо вибрати компонент із палітри й додати його до форми, інспектор об'єктів автоматично покаже властивості й події, які можуть бути використані із цим компонентом. У верхній частині інспектора об'єктів є список, що випадає, що дозволяє вибирати потрібний об'єкт із наявних на формі.

Поняття ***метод*** звичайно використовується в контексті компонентних класів і зовнішньо не відрізняється від терміна ***функція-член*** звичайного класу. Щоб викликати метод, треба вказати ім'я функції в контексті даного класу. Саме схований зв'язок методу з класом виділяє його з поняття простої функції. Під час виконання методу він має доступ до всіх даних свого класу. Це забезпечується передачею кожному методу схованого параметра – вказівника**this** на екземпляр класу. При будь-якому звертанні методу до членів даних класу, компілятор генерує спеціальний код, що використовує вказівник **this.**Метод є функцією, що пов'язана з компонентом і оголошується як частина об'єкта. Методи можна викликати, використовуючи операцію доступу через вказівник -> для цього компонента.

C++Builder дозволяє маніпулювати виглядом і функціональною поведінкою компонентів не тільки за допомогою методів (як це роблять функції-члени звичайних класів), але й за допомогою властивостей і подій, властивих тільки класам компонентів. Маніпулювати з компонентним об'єктом можна як на стадії проектування додатка, так і під час його виконання.

Властивості (properties) компонентів являють собою розширення поняття членів даних і, хоча не бережуть дані як такі, проте забезпечують доступ до членів даних об'єкта. C++Builder використовує ключове слово **__property** для оголошення властивостей. Властивості є атрибутами компонента [8], що визначають його зовнішній вигляд і поведінку. Багато властивостей компонента мають значення, за замовчуванням (наприклад, висота кнопок). Властивості компонента відображаються на сторінці властивостей (Properties). Інспектор об'єктів відображає опубліковані (published) властивості компонентів. Крім published-властивостей, компоненти можуть і найчастіше мають загальні (public) властивості, які доступні тільки під час виконання додатка. Інспектор об'єктів використовується для встановлення властивостей під час проектування. Список властивостей розташовується на сторінці властивостей інспектора об'єктів. Можна визначити властивості під час проектування або написати код для видозміни властивостей компонента під час виконання додатка.

При визначенні властивостей компонента під час проектування потрібно вибрати компонент на формі, відкрити сторінку властивостей в інспекторі об'єктів, вибрати потрібну властивість і змінити її за допомогою редактора властивостей (це може бути просте поле для введення тексту або числа; список, що випадає; список, що розкривається діалогова панель та інше).

Сторінка подій (**Events**) інспектора об'єктів показує список подій, розпізнаваних компонентом. За допомогою ***подій (events)*** компонент повідомляє користувачу про те, що на нього зроблений деякий визначений вплив (програмування для операційних систем із графічним користувацьким інтерфейсом, зокрема, для Windows XP або Windows NT передбачає опис реакції додатка на ті або інші події, а сама операційна система займається постійним опитуванням комп'ютера з метою виявлення настання якої-небудь події). Кожний компонент має свій власний набір обробників подій. У C++Builder необхідно писати функції, які називаються обробниками подій, і зв'язувати події із цими функціями. Якщо подія відбудеться і обробник цієї події написаний, то програма виконає написану функцію.

Для того, щоб додати обробник подій, потрібно вибрати на формі за допомогою миші компонент, якому необхідний обробник подій, потім відкрити сторінку подій інспектора об'єктів і двічі клацнути лівою клавішею миші на колонку значень поруч із подією, щоб C++Builder згенерував прототип обробника події і показав його в редакторі коду. При цьому автоматично генерується текст порожньої функції, і редактор відкривається в тому місці, де треба вводити код. Курсор позиціонується усередині операторних дужок { ... }. Далі потрібно ввести код, який повинен виконуватися при настанні події. Обробник подій може мати параметри, які вказуються після імені функції в круглих дужках.

C++Builder використовує ключове слово __**closure** для оголошення подій. Основна сфера застосування методів – це ***обробники подій (event handlers)***,що реалізують реакцію програми на виникнення визначених подій. Типові прості події – натискання кнопки або клавіші на клавіатурі.

4. Узагальнене програмування (generic programming)

Поряд з ООП у C++Buider широко підтримується ***узагальнене програмування*** [15], яке спрямоване на спрощення повторного використання коду і на абстрагування загальних концепцій. Якщо в ООП акцент програмування ставиться на дані, то в узагальненому програмування – на алгоритми.

Узагальнене програмування має справу з абстрагуванням і класифікацією алгоритмів і структур даних. У багатьох випадках алгоритм можна виразити незалежно від деталей подання оброблюваних ним даних. Термін ***узагальнений*** приписується коду, тип якого є незалежним. В узагальненому програмуванні можна один раз написати функцію для узагальненого, тобто невизначеного типа і потім використовувати її для множини існуючих типів.

Концепція узагальненого програмування припускає використання типів даних як параметрів. При розробці алгоритму, що може працювати із множиною типів і структур даних, використовується якийсь абстрактний тип, що згодом параметризується. Такий підхід забезпечує простий спосіб введення різного роду загальних концепцій і позбавляє програміста від написання вручну спеціалізованого коду.

У мові програмування C++ узагальнене програмування реалізується за допомогою шаблонів. Шаблон являє собою параметризоване визначення деякого елемента програми. При цьому розроблювач усього лише вказує компілятору правило для генерації (породження) одного або декількох конкретних екземплярів цього елемента програми. У C++ допускається створення шаблонів для функцій, методів і класів. При цьому як параметр шаблона можна використовувати тип (клас), звичайний параметр відомого типу, інший шаблон. У шаблона може бути кілька параметрів.

Поряд із засобами визначення власних користувацьких шаблонів у розпорядження розробника надається стандартна бібліотека. У ній визначені шаблони, що представляють найбільш використовувані структури даних, а також алгоритми для їхньої обробки.

Отже, найбільш перспективним стає наступний (більш високий у порівнянні із класами) рівень абстрактного програмування – створення своїх і використання стандартних шаблонів і узагальнених алгоритмів стандартної бібліотеки, які вже розроблені й налагоджені професіоналами. Тому далі розглянемо стандартну бібліотеку шаблонів (STL).