В. П. Семеренко

ПРОГРАМУВАННЯ МОВАМИ

С та С++

В СЕРЕДОВИЩІ WINDOWS

УДК 681.3.06.(075)
С30

Р е ц е н з е н т и:

С.В. Юхимчук, доктор технічних наук, професор
В.П.Тарасенко, доктор технічних наук, професор
С.Т.Володарський, доктор технічних наук, професор

Рекомендовано до видання Міністерством освіти і науки України.
Лист №14/18-2-1696 від 22.10.03

Семеренко В.П.
С30 **Програмування мовами С та С++ в середовищі Windows**. Навчальний посібник. - Вінниця: УНІВЕРСУМ – Вінниця", 2003. - 128 с.

ISBN 966-641-075-3

В посібнику розглянута методика складання програм мовою С з використанням функцій API Windows та мовою С++ з використанням бібліотеки класів MFC програмного пакета Visual C++ 6.0. Даються основи об'єктно-орієнтованого програмування мовою С++. Посібник призначений для студентів спеціальності 7.091501 "Комп'ютерні системи та мережі" для вивчення дисциплін "Системне програмне забезпечення" і "Програмування", а також може бути рекомендований студентам інших спеціальностей, пов'язаних з вивченням сучасного програмного забезпечення.

УДК 681.3.06.(075)

ISBN 966-641-075-3

ЗМІСТ

Передмова

Писати підручники посібники по сучасному програмуванню ризиковано і важко. Ризиковано, тому що вік таких книг, як правило, недовгий. Нові мови програмування, нові стилі програмування, нові програмні пакети змінюються з калейдоскопічною швидкістю. Щоб нові книги по програмному забезпеченню були корисними, вони повинні видаватись та оновлюватись такими ж швидкими темпами.

Чисельні видавництва випускають велику кількість книг і на перший погляд здається, що важко знайти ті сфери, де існує гостра потреба в навчальній літературі.

Однак при прискіпливому аналізі можна помітити деякі загальні закономірності.

Вимога ринкових відносин бути першими в представленні чергового програмного продукту змушує деяких авторів використовувати “конвейерні” способи написання нових книг. Цей спосіб полягає в 4сумлінному перерахуванні всіх існуючих режимів роботи нового програмного продукту, всіх форматів команд, всіх пунктів меню, всіх керуючих кнопок і т. д. По суті книга перетворюється в звичайний довідник, який мало відрізняється від документації фірми-виробника. Безумовно, такі книги теж потрібні, однак вони мало допоможуть у навчанні програмуванню, особливо на першому етапі.

Існують також книги, основний зміст яких складається з програмного коду з невеликими коментарями. Як правило, ці програми є копіями програм, що їх автоматично генерують різні “Майстри” (Wizard) програмних пакетів. Такий “Майстер” може швидко створити програмний код під невеликий набір стандартних завдань користувача. Зате в результаті утворюються великі за обсягом програми, в яких важко розібратись новачку, а тим більш внести свої корективи. Звичайно, аналіз таких програм теж корисний та цікавий.

І все ж для того, щоб навчитись самому програмувати і створювати ефективні програми, варто йти іншим шляхом.

Методика навчання програмування, подібно іншому предмету, передбачає рух від простого до складного. Спочатку необхідно спробувати самостійно написати найпростішу, але працюючу програму. Подальше засвоєння нових тем і розділів буде ускладнювати початкову просту програму. Самостійно змінюючи власну програму та додаючи до неї нові можливості, можна в кінцевому результаті навчитись створювати досить складні програми.

Неоціненну роль в такому процесі навчання і повинні відіграти підручники і навчальні посібники. І ось тут виявляється, що підготувати такі навчальні книги дуже непросто. Адже, з однієї сторони, необхідно мати відпрацьовану роками методику вивчення конкретної мови програмування, а з іншої сторони, швидке моральне старіння програмних

продуктів встановлює дуже жорсткі строки в підготовці навчальних методик і програмних текстів.

Найбільш корисні і вдалі книги з вивчення мов програмування, зокрема мов С та С++, пишуть ті автори, які постійно працюють в цій області програмного забезпечення. Тому кожна їх нова книга лише доповнює і збагачує новими матеріалом попередню. Це справедливо, зокрема, до відомого у світі програміста Герберта Шилдта. В його книгах є все: і доступність викладення, і детальний аналіз різних “підводних каменів” в програмах, і повні тексти програм, причому працюючих програм. І недаремно кожна книга цього чудового програміста стає справжнім бестселером.

Створюючи цей навчальний посібник, автор теж намагався писати в стилі Г. Шилдта. В якій мірі цього вдалося досягти, вирішувати читачам.

Методичні рекомендації до вивчення навчального посібника

Для успішного засвоєння матеріалу даного посібника необхідні базові знання з програмування та вміння складати програми мовою С у традиційному процедурному стилі. Читач має бути знайомим з програмуванням розгалужень і циклів, вміти працювати з функціями та файлами.

Навчальний посібник складається з чотирьох розділів, з яких другий і четвертий розділи є основними, а інші розділи – допоміжними.

Перший розділ буде корисний тим, хто вперше починає працювати з програмним пакетом Visual C++ 6.0. У цьому розділі основна увага приділена практичним порадам по створенню проектів для 32-розрядних програм Windows та виконанню програм в різних режимах налагодження.

Автор відмовився від того, щоб дати хоча б коротку характеристику всім командам головного меню, всім керуючим кнопкам і всім інструментам інтегрованого середовища. По-перше, це потребує значного обсягу матеріалу, по-друге, вся довідкова інформація вже наведена в чисельних книгах і довідниках з Visual C++ 6.0. Але найголовніший аргумент такої відмови полягає в тому, що для створення нескладних прикладних програм в в середовищі Windows зовсім не обов’язково спочатку вивчити всі команди інтегрованого середовища. Знайомство з новими командами буде відбуватися поступово, по потребі, можливо, що у використанні деяких команд чи режимів роботи взагалі не виникне потреба. Саме тому спочатку достатньо добре засвоїти послідовність створення лише кількох конкретних типів проектів.

Другий розділ посібника є основним, оскільки присвячений розробці програм мовою С в середовищі Windows. Тут детально пояснюється структура API-програм, принципові відмінності програмування в операційних системах Windows 95/98/ME від інших систем. Спочатку

увага звертається на програмування векторної графіки і тексту в API-програмах. Далі розглядається створення проектів з використанням найбільш поширених видів ресурсів: меню та діалогу.

Для того, щоб зрозуміти ідею створення програм з використанням бібліотеки класів MFC пакета Visual C++ 6.0, необхідно спочатку добре засвоїти принципи об’єктно-орієнтованого програмування (ООП). Цим питанням присвячений ще один допоміжний розділ посібника – третій розділ. Тут пояснюються три складові частини ООП: інкапсуляція, успадкування і поліморфізм. На відміну від першого допоміжного розділу, цей розділ можна вважати теоретичним. Приклади приведених тут програм не орієнтовані на роботу в середовищі Windows і можуть бути виконані в найпростішій однозадачній операційній системі, наприклад в MS-DOS. Якщо читачі вже знайомі з основами ООП, тоді третій розділ можна пропустити і приступити відразу до вивчення четвертого розділу.

Останній розділ посібника, який присвячений програмуванню мовою C++ з використанням бібліотеки класів MFC, є найскладнішим. Він потребує великої уваги і творчого підходу до вивчає мого матеріалу. Теми кількох підрозділів четвертого розділу (створення меню та діалогових вікон, програмування векторної графіки) схожі з однойменними темами в другому розділі. Це дає змогу порівняти різні підходи до виконання однакових задач, відчути переваги, що надає бібліотека класів MFC. В останньому розділі розглядається також і ряд нових тем.

Засвоїти програмування в операційних системах Windows 95/98/ME неможливо тільки теоретично, навіть при наявності всієї виданої документації. Вивчення програмування невіддільне від практики. Тому найкращий варіант роботи з даним посібником може бути тільки разом із комп’ютером. Перевіряючи наведені в посібнику фрагменти програм і змінюючи їх, можна зрозуміти принципи сучасного програмування.

В цілому даний посібник буде корисний при перших кроках в програмуванні мовами C та C++ в середовищі Windows. Для більш глибокого вивчення всіх тонкощів цього виду програмування необхідно, звичайно, користуватись і іншими книгами і посібниками, кращі з яких наведені в списку літератури.

Вступ

Головна задача сучасного програмування – це створення багатократно використовуваних незалежних елементів, придатних для складання різноманітних конкретних програм [1].

Традиційна технологія програмування зароджувалась в умовах, коли програма створювалась під одну конкретну задачу, а спосіб створення самої програми не мав принципового значення. Створення програм в ті далекі вже роки було майже мистецтвом, доступним вузькому колу професіоналів.

Однак в міру збільшення обсягу програм та необхідності модернізації раніше розробленого програмного забезпечення традиційний підхід до програмування швидко себе вичерпав. Колектив програмістів міг ефективно працювати над одним проектом лише тоді, коли всі члени колективу притримувались єдиної методики створення програм. Тільки в таких умовах програмні модулі окремих програмістів могли бути зістиковані разом за короткий проміжок часу і практично без помилок. Розробка єдиного методу на всіх етапах життєвого циклу програмного проекту – специфікації, проектування, власне програмування (кодування) і тестування – свідчила про появу нового стилю програмування, який отримав назву структурного програмування. З'явившись в 70-х роках двадцятого століття структурний підхід дозволив надати виробництву програм індустріальний характер.

Однак з часом все помітнішою ставала обмеженість структурного підходу, особливо коли постала необхідність повторного використання раніше розроблених програмних модулів. Ця потреба, а також і інші вади структурного програмування стимулювали появу нової технології – об'єктно-орієнтованого програмування (ООП). На відміну від бібліотек стандартних підпрограм, в яких теж використовуються повторні модулі, об'єктний підхід дозволяв створити ще ієрархію вкладених один в одного модулів.

Основи ООП були закладені ще на початку 80-х років, але і досі продовжуються дискусії з приводу того, чи повністю виправдала нова технологія покладені на неї великі надії. Опоненти ООП часто звертають увагу, зокрема, на складність програм, написаних в цьому стилі. І дійсно, саме намагання спростити процес написання програм було однією з вагомих причин появи на початку 90-х років нового напрямку в програмуванні – компонентного програмування (КП). В основі КП лежить досить цікава ідея – повторно використовувати можна не тільки програмний код модуля, але і готові результати модуля.

Таким результатом роботи модуля можуть бути, наприклад, елементи інтерфейсу програми: меню, командні кнопки, графічні зображення тощо. В такому разі програму вже не треба писати на папері, а досить лише її "побудувати" з готових "будівельних блоків" на екрані дисплея.

Більше того, компілятор ще й самостійно згенерує програмний код під створену таким чином програму. В цьому полягає суть візуального програмування – найбільш поширеного варіанта КП.

Звичайно, програмісту ще теж вистачить роботи, хоча процес підготовки програм вже значно спрощується. Зрозуміло, що таке спрощення програмування стосується в першу чергу інтерфейсних програм, в інших випадках застосування візуального програмування може бути недоцільним.

Існує чимало пояснень причин появи кожного стилю програмування. У короткому вступі неможливо провести аналіз всіх цих причин, навести приклади інших підходів у сучасному програмуванні. Підсумовуючи короткий історичний аналіз необхідно відмітити, що сьогодні існують поруч різні стилі і напрямки у програмуванні і кожна нова технологія не відміняє попередню. В багатьох сучасних програмних пакетах використовуються елементи різних технологій і їх вибір залежить в першу чергу від особливостей поставленої задачі. Наприклад, в пакеті Visual C++ 6.0, вивченню якого присвячений цей навчальний посібник, використовується і ООП, і КП.

Історія програмування пов’язана не тільки з розвитком стилів, але і з розробкою самих мов програмування. За останні 50 років було створено кілька тисяч мов програмування [2]. Світове визнання отримали лише кілька десятків мов програмування, які знайшли своє застосування на різних типах комп’ютерів.

Обмежимо наш подальший аналіз лише універсальними мовами програмування, тобто мовами високого рівня. В основу класифікації таких мов можна покласти одне з фундаментальних понять в теорії програмування – поняття типів даних.

За загально прийнятним визначенням, тип даних – це множина значень, які приймають дані, множина операцій над даними і розмір пам’яті, які займають дані. З точки зору можливості зміни даних програмістом, дані поділяються на декілька категорій.

До першої категорії відносять дані, які програміст не має права змінювати. Це так звані базові типи даних: цілі, дійсні, символьні, булеві. Деякі з них можуть утворювати однорідні (тобто із одного типу) об’єднання – масиви. Ті мови, які містять лише базові типи даних, називаються мовами програмування 1-го покоління. Найбільш відомі мови 1-го покоління – Фортран, Алгол-60, Бейсик. Період найбільшого застосування мов 1-го покоління – 60-70-ті роки.

До другої категорії відносять дані, які програміст має право сам створити, об’єднавши кілька базових типів даних. Головна відмінність від даних першої категорії полягає в тому, що в цьому об’єднанні можна використати різні базові типи даних. В результаті утворюються конструйовані типи даних. Приклади таких типів даних: структури (struct) і об’єднання (union) в мові С або записи (record) в мові Pascal. Ті мови, які містять,

окрім базових, ще й конструйовані типи даних, називаються мовами програмування 2-го покоління. Найбільш відомі мови 2-го покоління – C, Pascal, PL/1. Період найбільшого застосування мов 2-го покоління – 70-80-ті роки.

До третьої категорії відносять дані, які програміст має право створити, додавши в конструйовані типи також і операції над даними . В результаті утворюються абстрактні типи даних, в яких можуть бути об'єднані кілька базових типів даних і введені функції з цими базовими типами. Приклади таких типів даних: класи (class) в мові C++ або об'єкти (object) в мові Object Pascal. Ті мови, які містять, окрім базових і конструйованих, ще й абстрактні типи даних, називаються мовами програмування 3-го покоління. Найпершими представниками мов 3-го покоління стали об'єктні мови Simula-67 та SMALLTALK. Згодом абстрактні типи даних були введені в універсальні мови програмування, як у нові, так і у вже існуючі мови 2-го покоління. Тому мови 3-го покоління також називають об'єктно-орієнтованими. До таких мов відносять: ADA, C++, Object Pascal, Modula-2, CLU. Період найбільшого застосування мов 3-го покоління – 80-90-ті роки.

До четвертої категорії відносять дані, які програміст має право створити, додавши в абстрактні типи даних нові можливості роботи над даними. Якщо абстрактний тип даних характеризується двома параметрами (дані і функції над ними), то тип даних четвертої категорії характеризується вже трьома параметрами: властивостями, подіями та методами. Методи – це ті ж функції, властивості є подальшим розвитком параметра "дані", а події – це вже новий параметр. В результаті утворюються компонентні типи даних. Приклади таких типів даних: керуючі елементи в мові Visual Basic або компоненти в програмних пакетах Visual C++, C++ Builder та Delphi. Ті мови, які містяться в останніх трьох пакетах (C++ та Object Pascal) , можна назвати мовами програмування 4-го покоління. Ці мови програмування, незважаючи на однакові назви з мовами 3-го покоління, мають дуже багато нових характеристик, що дає право віднести їх вже до нового покоління. До 4-го покоління можна віднести також ще кілька зовсім нових мов програмування, орієнтованих на роботу в комп'ютерних мережах, наприклад мову Java. Варто також відмітити, що обов'язковим атрибутом мов 4-го покоління часто вважають також можливість роботи зі стандартними базами даних. Період початку широкого застосування мов 4-го покоління – починаючи з 90-тих років і до нашого часу.

Цікаво провести спільний аналіз мов різних поколінь із стилями програмування. Зовсім неважко помітити майже повний збіг одного стилю програмування – традиційного, структурного, об'єктно-орієнтованого та компонентного – відповідному поколінню мов програмування.

Література

1. Г.С.Цейтин. На пути к сборочному программированию// Программирование. - 1990. №1. С.78-92.
2. Гради Буч. Объектно-ориентированное программирование с примерами применения: Пер. с англ.-М.:Конкорд,1992.-519с.
3. М.Уэйт, С.Прата, Д.Мартин. Язык Си. Руководство для начинающих: Пер. с англ.-М.: Мир,1988.-512с.
4. Гладков С.А., Фролов Г.В. Программирование в Microsoft Windows: В 2-х ч.-М.:"Диалог-МИФИ",1992.-320с.,288с.
5. Г.Шилдт. Программирование на С и С++ для Windows 95:Пер. с англ.- К.:Торгово-издательское бюро HBV,1996.-400с.
6. Г.Шилдт. Самоучитель С++: Пер. с англ.-СПб.:HBV-Санкт-Петербург, 1997.-512с.
7. М.Луис. Visual С++6.-М.:Лаборатория Базовых Знаний, 1999.-720с.-(справочник).
8. С.Холзнер. Visual С++6.: Учебный курс-СПб.:Питер,2001.-576с.
9. Ю.Тихомиров. Visual С++6.-СПб.:BHV-Санкт-Петербур,1999,496с.
10. А.Мешков, Ю.Тихомиров. Visual С++ и MFC. Программирование для Windows NT и Windows 95: В 3-х томах.-СПб.:BHV-Санкт-Петербург, 1997.-464с.
11. К.Паппас, У.Мюррей. Visual С++6: для пользователя: Пер. с англ.-К.: Издательская группа BHV, 2000.-336с.
12. М.Янг. Векторная графика на языке С++/ Пер. с англ.-М.: Восточная Книжная Компания, 1997.-368с.
13. Гринзоу. Философия программирования для Windows 95/NT: Пер. с англ.- СПб: Символ ПМОС, 1997.-640с.