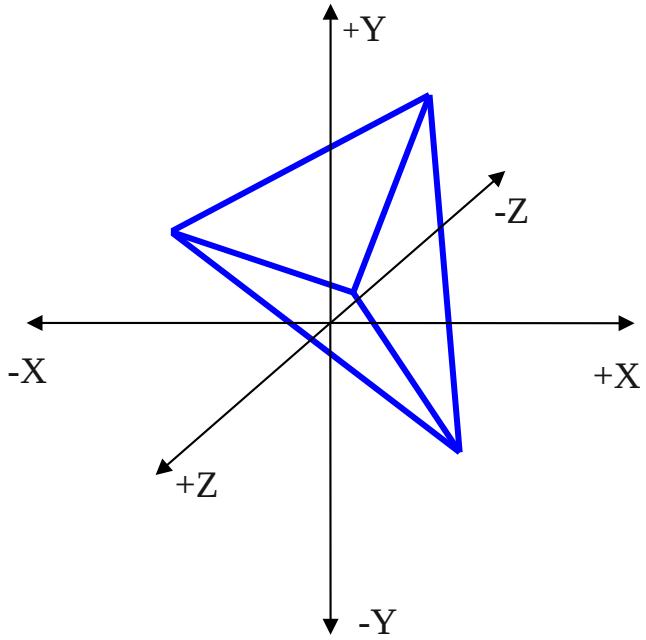
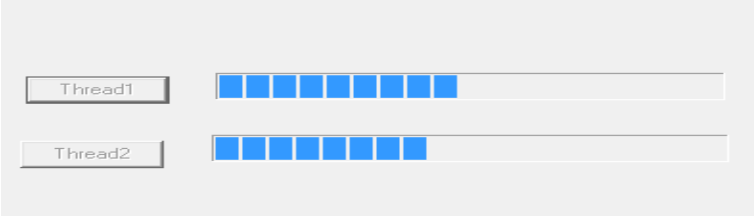
В. П. Семеренко

СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

СИСТЕМНЕ ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ C++

Курсове проектування
Самостійна та індивідуальна робота студентів





Міністерство освіти і науки України
Вінницький національний технічний університет

В. П. Семеренко

СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

СИСТЕМНЕ ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ С++

Курсове проектування
Самостійна та індивідуальна робота студентів

Навчальний посібник

Вінниця
ВНТУ
2015

УДК 681.3.06.(075)
ББК 32.973.26-018.1
С34

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 5 від 12.12.2013 р.)

Рецензенти:
В. А. Лужецький, доктор технічних наук, професор
О. М. Роїк, доктор технічних наук, професор
О. О. Коваленко, кандидат технічних наук, доцент

Семеренко, В. П.

С34 Системне програмне забезпечення. Системне об’єктноорієнтоване програмування мовою С++. Курсове проектування. Самостійна та індивідуальна робота студентів : навчальний посібник / В. П. Семеренко. – Вінниця : ВНТУ, 2015. – 195 с.

В посібнику розглянута методика складання програм мовою С++ на основі бібліотеки класів MFC програмного пакета Microsoft Visual Studio. Детально викладені питання роботи з файлами, керування потоками, створення динамічно підключуваних бібліотек, розробці візуального інтерфейсу з використанням меню, панелі інструментів, діалогових вікон. Велику увагу приділено тривимірній графіці на основі бібліотеки OpenGL. Посібник призначений для студентів бакалаврського напрямку 6.050102 “Комп’ютерна інженерія” для вивчення дисциплін "Системне програмне забезпечення", а також може бути корисним при вивченні дисципліни "Системне програмування".

УДК 681.3.06.(075)
ББК 32.973.26-018.1

ЗМІСТ

Передмова

Все програмне забезпечення (ПЗ або software) в сучасних комп'ютерах можна розділити на дві складові:

– системне програмне забезпечення (СПЗ),

– прикладне програмне забезпечення (ППЗ).

СПЗ – це комплекс програм, який виконує керування комп'ютером або комп'ютерною мережею, забезпечує виконання прикладних програм та забезпечує зручний інтерфейс з користувачем. Відповідно СПЗ також можна розбити на дві великі частини: керуюче СПЗ та оброблювальне СПЗ.

До керуючого СПЗ належать операційні системи, операційні оболонки, окремі драйвери, системні утиліти. До оброблювального СПЗ можна віднести системні оброблювальні програми: транслятори, компоновані, завантажувачі, налагоджувачі.

Основою керуючого СПЗ є операційна система (ОС), яка виконує такі основні функції:

– керування процесами,

– керування основною пам'яттю,

– керування введенням-виведенням,

– керування файлами,

– керування конфігурацією комп'ютера,

– організація діалогу з користувачем,

– виконання додаткових функцій (забезпечення зв'язку з іншими комп'ютерами, мультимедійні функції та інші).

До ППЗ належать інструментальне програмне забезпечення (інтегровані середовища розробки програм, офісні пакети, системи керування базами даних тощо), навчальні та ігрові програми, програми технічного обслуговування комп'ютера. ППЗ включає як програми відомих комп'ютерних фірм, так і програми користувачів.

За останні десятиліття створена величезна кількість програм, які іноді важко віднести до того чи іншого традиційного розділу. Тому з'явився новий термін – проміжне (middleware) програмне забезпечення. Таке ПЗ здійснює керування ресурсами, що створено іншими програмами (наприклад, менеджери транзакций, сервери комунікацій).

Системне програмування – це є створення СПЗ, а програмістів, які спеціалізуються на цьому, називають системними програмістами.

Звичайно, створити нову операційну систему не під силу одному програмісту (чи не єдиним винятком може служити Лінус Торвальдс, який першим написав ядро майбутньої ОС Linux). Однак роботи системним програмістам вистачає, оскільки постійно розробляються різноманітні сервіси до відомих ОС, створюються нові інформаційні системи та комплекси програм.

Такі програми містять в собі як системні елементи, так і елементи прикладного ПЗ. Тому сучасні програмісти мають бути універсалами і добре орієнтуватися в багатому різномаїтті ПЗ.

Саме з таких позицій і розглядається курсове проектування з дисципліни “Системне програмне забезпечення”. В посібнику розглядаються такі елементи СПЗ:

1) мови системного програмування: C++ та Асемблер;
2) створення багатопотокових програм;
3) робота зі статичними та динамічними бібліотеками;
4) виконання операцій з файлами;
5) керування введенням-виведенням (на прикладі дисплея).

До ППЗ можна віднести розробку сучасного інтерфейсу користувача на основі меню, панелі інструментів, діалогових вікон та чудової тривимірної графіки на основі бібліотеки OpenGL.

Методичні рекомендації до вивчення навчального посібника

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

Перший розділ буде корисний тим, хто вперше починає працювати з програмним пакетом Microsoft Visual Studio. У цьому розділі основна увага приділена практичним порадам зі створення проектів для 32-розрядних програм Windows та виконання програм в різних режимах налагодження. В посібнику розглядається версія Microsoft Visual Studio Ultimate 2012, але наведені програми будуть також працювати і в попередніх версіях цього пакета.

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 багатьох книгах і довідниках з Visual Studio. 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, за потреби, можливо, що у використанні деяких команд чи режимів роботи взагалі не виникне потреба.

В більшості книг, присвячених пакету Microsoft Visual Studio розглядається спосіб створення програм на основі різних “Майстрів” (Wizard). Такий “Майстер” може швидко згенерувати програмний код під невеликий набір стандартних завдань користувача. Зате в результаті утворюються великі за обсягом програми, в яких важко розібратись

новачку, а тим більше 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.

Для того, щоб зрозуміти ідею створення програм з використанням бібліотеки класів MFC пакета Visual Studio, необхідно спочатку добре засвоїти принципи об’єктно-орієнтованого програмування (ООП). Цим питанням присвячений другий розділ. Тут пояснюються три складові частини ООП: інкапсуляція, успадкування і поліморфізм. На відміну від першого розділу, цей розділ можна вважати теоретичним. Приклади наведених тут програм можуть бути виконані в рамках консольного типу проекта Visual Studio. Якщо читачі вже знайомі з основами ООП, тоді третій розділ можна пропустити і приступити відразу до вивчення наступних розділів.

Третій розділ посібника, який присвячений програмуванню мовою C++ з використанням бібліотеки класів MFC, є найскладнішим. Він потребує великої уваги і творчого підходу до вивчення цього матеріалу. Порівняно з попереднім виданням цього посібника [1] значно розширено зміст всіх тем і додано нові теми: робота з файлами, багатопотокове програмування.

В четвертому розділі посібника, який є новим, розглядається програмування тривимірної графіки на основі бібліотеки OpenGL. Завдяки загальнодоступності та багатоплатформності ця графічна бібліотека отримала широку популярність. І хоча їй присвячено вже багато книг, однак особливості застосування OpenGL в MFC-програмах майже ніде не розглядається. Саме тому цей матеріал і включено до посібника.

Засвоїти програмування в операційних системах Windows 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 мовою 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.

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 ефектив-но 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 програ-мування – 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 Microsoft Visual Studio, 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 60 років було створено кілька тисяч мов програмування. 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 Studio, C++ Builder та Delphi. Ті мови, які містяться в останніх трьох пакетах (C++, 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. 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.

Варто також відмітити, що відбувається взаємозбагачення різних стилів і різних мов програмування. Наприклад, до складу класів в мові C# входять властивості – параметр компонентного типу даних.

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

Список літератури

1. Бондаренко М. Ф. Операционные системы: Учебное пособие / М. Ф. Бондаренко, Е. Г. Качко. — Харьков : “Компания СМИТ”, 2006. — 444 с.
2. Шеховцов В. А. Операційні системи / Шеховцов В. А. — К. : Видавнича група BHV, 2005. — 576 с.
3. Луис М. Visual C++6 / Луис Б. И. — М. : Лаборатория Базовых Знаний, 1999. — 720 с.
3. Пахомов Б. И. C/C++ и MS Visual C++ 2012 для начинающих / Пахомов Б. И. — СПб. : БХВ-Петербург, 2013. — 512 с.
5. Райт Р. С. OpenGL. Суперкнига, 3-е изд. / Р. С. Райт, Б. Липчак. — М. : Издательский дом “Вильямс”, 2006. — 1040 с.
6. Семеренко В. П. Програмування мовами С та С++ в середовищі Windows: навчальний посібник / Семеренко В. П. — Вінниця : Універсум, 2003. — 128 с.
7. Семеренко В. П. Системне програмування мовою Асемблера. Лабораторний практикум. Част. 2. / В. П. Семеренко, В. А. Каплун. — Вінниця : ВНТУ, 2004. — 93 с.
8. Хортон А. Visual C++ 2005: базовый курс / Хортон А. — М. : Издательский дом “Вильямс ”, 2007. — 1152 с.
9. Глушаков С. В. Visual C++. Руководство разработчика / С. В. Глушаков, А. В. Коваль. — Харьков : Фолио, 2006. — 727 с.
10. Шеферд Дж. Программирование на Microsoft Visual C++ NET: Мастер-класс. — 2-е изд. / Шеферд Дж. — М. : “Русская Редакция”; СПб. : Питер 2007. — 928 с.
11. Шилдт Г. Программирование на C и C++ для Windows 95 / Шилдт Г. — К. : Торгово-издательское бюро HBV,1996. — 400 с.
12. Шилдт Г. Самоучитель C++ / Шилдт Г. — СПб. : HBV-Санкт-Петербург, 1997. — 512 с.
13.Эйнджел Э. Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2-е изд. / Эйнджел Э. — М. : Издательский дом “Вильямс”, 2001. — 592 с.
14. Юров В. Assembler: Учебник / Юров В. — СПб и др. : Питер, 2000. — 622 с.
15. ДСТУ 7.1:2006. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : метод. Рекомендації з впровадження / уклали: Галевич О. К., Штогрин І. М. Львів, 2008. — 20 с.
16. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. / Держстандарт України. Офіційне видання. — Київ, 1995. — 37 с.
17. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті / Лисенко Г. Л., Буда А. Г., Обертюх Р. Р. — Вінниця : ВНТУ, 2006. — 60 с.