МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ І МОЛОДІ ТА СПОРТУ УКРАЇНИ
Вінницький національний технічний університет
Інститут автоматики, електроніки та комп'ютерних систем управління
Кафедра автоматики та інформаційно-вимірювальної техніки

МЕТОДИЧНІ ВКАЗІВКИ

до самостійного вивчення розділу «**Обєктно-орієнтовне програмування**»

з курсу: «Програмування»

для студентів напряму 6.0502 "Системна інженерія"

Затверджено
на засіданні кафедри автоматики та інформаційно-вимірювальної техніки.
Протокол № __ від ____ _____ _____ р.

Вінниця ВНТУ

Вступ

В даний час можна виділити декілька типів мов програмування. Однією з найважливіших ознак їх класифікації є приналежність їх до одного із стилів, основними з яких є наступні стилі:

- процедурний
- функціональний
- логічний
- об'єктно-орієнтований.

Прототипом об'єктно-орієнтованого програмування послужили ряд засобів, що входять до складу мови Simula-67. Але в самостійний стиль воно оформилося з появою мови SMALLTALK, розробленого А. Кєєм в 1972 році і з початку призначеного для реалізації функцій машинної графіки.

Основна мета ООП, як і більшості інших підходів до програмування - підвищення ефективності розробки програм. Ідеї ООП виявилися плідними і знайшли застосування не тільки в мовах програмування, але і в інших областях Computer Science, наприклад, в області розробки операційних систем.

Концепція об'єктно-орієнтованого програмування має на увазі, що основою управління процесом реалізації програми є передача повідомлень об'єктам. Тому об'єкти повинні визначатися спільно з повідомленнями, на які вони повинні реагувати при виконанні програми. У цьому полягає головна відмінність ООП від процедурного програмування. Таким чином, об'єктно-орієнтована програма складається з об'єктів - окремих фрагментів кодів, які оброблюють дані, які взаємодіють один з одним через певні інтерфейси.

1. Об'єктно-орієнтована методологія програмування

Останнім часом розвиток апаратних засобів істотно випереджав розвиток систем і засобів програмування. Щоб виправити положення, в 70-80-х роках були запропоновані різні підходи до збільшення продуктивності праці програміста. Серед цих спроб виділяється такий напрям, як об'єктно-орієнтований підхід до конструювання і кодування програм. Особливу роль в популярності цього підходу зіграло як його тісний зв'язок з інтерфейсами користувача (особливо графічними), так і включення елементів цього підходу в популярні (на персональних комп'ютерах фірми IBM) реалізації гібридних мов програмування C++ і Pascal with Objects фірми Borland.

До цих пір більшість використовуваних програмних систем побудовані на принципах структурного підходу, суть якого полягає в декомпозиції системи на ряд модулів, процедур, функцій і структур даних, зв'язаних загальним алгоритмом функціонування. Але розповсюдження могутніх персональних комп'ютерів (порівнянних з робочими станціями 70-80-х років) створило в 90-х роках основу для широкого застосування об'єктно-орієнтованого підходу на практиці. Останнім часом ширше починають використовуватися мови програмування, створені в рамках об'єктно-орієнтованої методології, такі як Smalltalk і Java.

Об'єктно-орієнтована методологія (ООМ) орієнтована, перш за все, на створення великих систем, колективну їх розробку, подальший активний супровід при експлуатації і регулярні модифікації. Серед типових завдань, для яких ООМ є перспективною, можна виділити такі:

- автоматизація експерименту, робототехника;
- планування;
- інтерфейс користувача, анімація;
- комунікації, зв'язок;
- медицина, експертні системи;
- обробка комерційної інформації;
- операційні системи;

- системи управління;
- тренажери, моделювання.

У основі об'єктно-орієнтованої методології програмування лежить об'єктний підхід, коли прикладна наочна область представляється у вигляді сукупності об'єктів, які взаємодіють між собою за допомогою передачі повідомлень. Об'єкт - це сукупність даних (змінних) і способів роботи з ними (компонентних процедур і функцій). Стан об'єкту характеризується переліком всіх його можливих (зазвичай статичних) властивостей і значеннями кожної з цих властивостей (зазвичай динамічних). Стан об'єкту описується його змінними. Поведінка об'єкту (або його функціональність) характеризує те, як об'єкт взаємодіє з іншими об'єктами або піддається взаємодії інших об'єктів, проявляючи свою індивідуальність. Індивідуальність - це такі властивості об'єкту, які відрізняють його від всіх інших об'єктів. Поведінка об'єкту реалізується у вигляді функцій, які називають методами. При цьому структура об'єкту доступна тільки через його методи, які в сукупності формують інтерфейс об'єкту.

Такий підхід дозволяє локалізувати ухвалювані рішення рамками об'єкту, об'єднуючи в нім і структуру, і поведінку, а, отже, знижуючи складність окремої програми (що реалізовує об'єкт). Ця ідея об'єднання структури і поведінки в одному місці і захована всіх даних усередині об'єкту, що робить їх невидимими для всіх, за винятком методів самого об'єкту називається інкапсуляцією. Це дозволяє об'єктам функціонувати абсолютно незалежно один від одного, приховуючи за інтерфейсом деталі реалізації. Інкапсуляція дозволяє розглядати об'єкти, як ізольовані "чорні ящики", які знають і уміють виконувати певні дії. З цієї точки зору, внутрішній устрій "чорних ящиків" для нас значення не має, нам все одно, що відбувається усередині. Важливо тільки знати, що треба покласти в ящик при зверненні до нього і що ми при цьому з нього отримаємо. Таким чином, об'єкти об'єктно-орієнтованих систем - це мінімальні одиниці інкапсуляції. Але як управляти таким обсягом об'єктів, коли їх стало досить багато? Адже багато хто з них буде дуже сильно відрізнятися один від одного, а інші об'єкти будуть дуже схожі один на одного. Тут на сцену виходить одна з клю-

чових концепцій об'єктно-орієнтованого програмування - ідея угрупування об'єктів в класи, відповідно до того як вони влаштовані і діють. Така ідея вперше була реалізована ще в 60-і роки в мові Simula. Під класом розуміється безліч об'єктів, зв'язаних спільністю структури і поведінки. Клас можна порівняти з шаблоном, по якому створюються об'єкти. Саме клас спочатку описує змінні і методи об'єкту, тобто структуру і поведінку об'єкту, і визначає механізми створення об'єкту, що реально існує в системі, який, коли створюється, є екземпляр класу.

Наведення за допомогою класів порядку в світі об'єктів - велике досягнення, але можна піти далі, визначаючи деякий порядок і серед класів. Досягається це за допомогою введення механізму спадкоємства - мабуть, наймогутнішого засобу в будь-якій об'єктно-орієнтованій системі, оскільки воно дозволяє багато разів використовувати одного разу створений код. Механізм спадкоємства дуже простий: один клас, званий в рамках цих відносин суперкласом, повністю передає іншому класу, який називається підкласом, свою структуру і поведінку, тобто всі свої змінні і всі методи. Що далі робити з цим багатством визначає тільки підклас: він може додати в структуру щось своє, щось з успадкованого інтерфейсу він може використовувати без змін, щось змінити, і, зрозуміло, може додати свої власні методи. Тобто клас за допомогою підкласів розширюється, і як результат, створювані об'єкти стають все більш і більш спеціалізованими. Класи, розташовані за принципом спадкоємства, починаючи з найзагальнішого, базового класу, утворюють ієрархію класів.

Зрозуміло, система, що реалізовує такі принципи побудови, пред'являє жорсткіші, ніж при структурному підході, вимоги до продуктивності обчислювальної системи.

Після короткого опису деяких ключових понять об'єктно-орієнтованої методології, можна зробити вивід, що концептуально об'єктно-орієнтована методологія програмування спирається на об'єктний підхід, який включає чотири основні принципи:

Абстрагування.

Виділення таких істотних характеристик об'єктів, які відрізняють його від всіх інших об'єктів і які чітко визначають особливості даного об'єкту з погляду подальшого розгляду і аналізу. Тільки істотне для даного завдання і нічого більш. Мінімальною одиницею абстракції в ООМ є клас.

Обмеження доступу.

Процес захисту окремих елементів об'єкту, що не зачіпає істотних характеристик об'єкту, як цілого.

Модульність.

Властивість системи, пов'язана з можливістю декомпозиції на ряд тісно зв'язаних частин (модулів). Модульність спирається на дискретне програмування об'єктів, які можна модернізувати або замінювати, не впливаючи на інші об'єкти і систему в цілому.

Існування ієрархій.

Ранжирування, впорядковування за деякими правилах об'єктів системи.

Об'єктно-орієнтована методологія (ООМ) складається з наступних частин:

- об'єктно-орієнтований аналіз (ООА)
- об'єктно-орієнтоване проектування (OOD)
- об'єктно-орієнтоване програмування (ООР).

ООА - методологія аналізу суті реального світу на основі понять класу і об'єкту, складових словник наочної області, для розуміння і пояснення того, як вони (суть) взаємодіють між собою. Моделі ООА надалі перетворюються в об'єктно-орієнтований проект.

OOD - методологія проектування програмного продукту, що сполучає в собі процес об'єктної декомпозиції, що спирається на виділення класів і об'єктів, і прийоми представлення моделей, що відображають логічну (структура класів і об'єктів) і фізичну (архітектура моделей і процесів) структуру системи. Наступні поняття є в OOD фундаментальними:

Інкапсуляція.

Інкапсуляція є найважливішою властивістю об'єктів, на якій будується

об'єктно-орієнтоване програмування. Інкапсуляція - це об'єднання в одному класі даних і дій над ними. При цьому включені в об'єкт підпрограми (методи) безпосередньо працюють з даними цього об'єкту, звертаються до інших методів цього об'єкту або методів об'єктів-предків.

Інкапсуляція дозволяє багато в чому ізолювати клас від решти частин програми, зробити його «самодостатнім» для вирішення конкретного завдання. В результаті клас завжди несе в собі деяку функціональність. Наприклад, клас Tform в Delphi містить (інкапсулює в собі) все необхідне для створення Windows-вікна, клас Ttimer забезпечує роботу таймера і так далі.

Одиницею інкапсуляції в OOD є об'єкт.

Спадкоємство дозволяє створювати ієрархію класів, починаючи з деякого первинного базового класу (предка) і продовжуючи складнішими, такими, що включають (що успадковують) елементи попередніх класів (нащадків або похідних класів).

Будь-який клас може бути породжений від іншого класу. Для цього при його оголошенні вказується ім'я класу-батька:

Tchildclass = class(Tparentclass) (мова Object Pascal)

Створений клас автоматично успадковує поля, методи і властивості свого батька і може доповнювати їх новими. Таким чином, принцип спадкоємства забезпечує поетапне створення складних класів і розробку власних бібліотек класів. Клас, поведінка якого успадковується, називається суперкласом, а клас, який успадковує поведінку, називається підкласом.

Принцип спадкоємства призводить до створення дерева з гілками класів, що постійно розростається. Кожен нащадок доповнює можливості свого батька новими і передає їх своїм нащадкам.

Поліморфізм - це властивість споріднених класів вирішувати схожі по сенсу проблеми різними способами. Для різних споріднених класів можна задати єдиний образ дії (наприклад, вивід на екран будь-якої геометричної фігури).

Потім для кожного конкретного класу складається своя підпрограма, що виконує цю дію безпосередньо для нього (відображення крапки відрізняється від відображення лінії і так далі), причому всі ці підпрограми повинні мати одне ім'я. Коли потрібно буде відобразити конкретну фігуру, буде вибрана зі всієї безлічі однойменних підпрограм та, яка відповідає типу конкретного об'єкту. Якщо об'єкт, що виводиться, є крапкою, то вибирається його підпрограма, якщо лінія - то її.

Таким чином, поліморфізм проводить ідею «один інтерфейс - безліч методів». Вибір конкретної дії залежить від ситуації.

Створений проект перетворюється на програмний продукт в процесі об'єктно-орієнтованого програмування - такій методології програмування, яка заснована на представленні програмного продукту у вигляді сукупності об'єктів, кожен з яких є зліпком (екземпляром) певного класу, а класи утворюють ієрархію на принципах спадкоємства. Таким чином, при об'єктно-орієнтованому підході зникає поняття виконуваної програми. Рішення поставленої задачі зводиться до побудови необхідних класів, і управління створюваними ними об'єктами-екземплярами.

Фундаментальна концепція ООР полягає в тому, що об'єкти і класи взаємодіють один з одним шляхом передачі повідомлень. Для цього необхідно, щоб об'єкти визначалися разом з повідомленнями, на які вони реагують, на відміну від процедурного стилю програмування, коли спочатку визначаються дані, які потім передаються в процедури (функції) як параметри. При цьому засобом програмування виступає одна з об'єктно-орієнтованих мов програмування.

Мова програмування називається об'єктно-орієнтованою, якщо є підтримка об'єктів як абстракцій даних, що мають інтерфейсну частину у вигляді поіменованих операцій, і захищену область локальних даних;

- всі об'єкти відносяться до відповідних типів (класам);
- класи можуть успадковувати від суперкласів.
- будь-які дані зберігаються як об'єкти, що розміщуються з автоматичним виділенням і звільненням пам'яті. Об'єкт існує в системі до тих пір, поки

його можна іменувати.

Останній принцип відрізняє чисті об'єктно-орієнтовані мови такі як Smalltalk, Actor, від гібридних мов програмування, що виросли з раніше існуючих процедурних мов (Object Pascal, C++). Ці підходи - як би крайнощі в сімействі об'єктно-орієнтованих мов. Ближче до середини лежить абсолютно новий, повністю побудований на принципах об'єктно-орієнтованої ідеології, але все таки останній принцип, що порушує, мова Java.

2. Об'єктно-орієнтовані мови програмування

Об'єктно-орієнтовані мови програмування користуються останнім часом великою популярністю серед програмістів, оскільки вони дозволяють використовувати переваги об'єктно-орієнтованого підходу не тільки на етапах проектування і конструювання програмних систем, але і на етапах їх реалізації, тестування і супроводу.

Перша об'єктно-орієнтована мова програмування Simula 67 була розроблена в кінці 60-х років в Норвегії. Автори цієї мови дуже точно вгадали перспективи розвитку програмування: їх мова набагато випередила свій час. Проте сучасники (програмісти 60-х років) опинилися не готові сприйняти цінності мови Simula 67, і вона не витримала конкуренції з іншими мовами програмування (перш за все, з мовою Fortran). Прохолодному відношенню до мови Simula 67 сприяла і та обставина, що вона була реалізована як мова, що інтерпретувалася (а не компільований), що було абсолютно неприйнятним в 60-і роки, оскільки інтерпретація пов'язана із зниженням ефективності (швидкості виконання) програм.

Але достоїнства мови Simula 67 були відмічені деякими програмістами, і в 70-і роки було розроблено велике число експериментальних об'єктно-орієнтованих мов програмування: наприклад, мови CLU, Alphard, Concurrent Pascal і ін. Ці мови так і залишилися експериментальними, але в результаті їх дослідження були розроблені сучасні об'єктно-орієнтовані мови програмуван-

ня.

Огляд деяких сучасних об'єктно-орієнтованих мов програмування.

Smalltalk

Мова Smalltalk була розроблена командою Xerox Palo Alto Research Center Learning Research Group як програмна частина Dynabook - фантастичного проекту Алана Кея. В основу були покладені ідеї Simula.. Smalltalk є одночасно і мовою програмування, і середовищем розробки програм. Це чисто об'єктно-орієнтована мова, в якій абсолютно все розглядається як об'єкти; навіть цілі числа - це класи. Услід за Simula, Smalltalk є найважливішою об'єктно-орієнтованою мовою, оскільки вона не тільки зробила вплив на подальші покоління мов програмування, але і заклала основи сучасного графічного інтерфейсу користувача, на яких безпосередньо базуються інтерфейси Macintosh, Windows і Motif.

Відомо п'ять випусків мови Smalltalk, що позначаються по року їх появи: Smalltalk-72 -74. -76, -78, -80. Реалізації 1972 і 1974 років заклали основу мови, зокрема ідею передачі повідомлень і поліморфізм, хоча механізм спадкоємства ще не з'явився. У подальших версіях повноправне громадянство отримали класи; цим досягла завершення точка зору, що все складається з об'єктів. Smalltalk-80 був перенесений на багато комп'ютерних платформ.

В основу мови покладено дві прості ідеї:

- все є об'єктами;
- об'єкти взаємодіють, обмінюючись повідомленнями.

Великим недоліком Smalltalk є великі вимоги до пам'яті і низька продуктивність отриманих програм. Це пов'язано з не дуже вдалою реалізацією об'єктно-орієнтованих особливостей.

C++

Мова програмування C++ була розроблена Бьерном Страуструпом, співробітником At&t Bell Laboratories. Безпосереднім попередником C++ є З with Classes, створений тим же автором в 1980 році. Мова З with Classes, у свою чергу, була створена під сильним впливом З і Simula. C++ - це в значній мірі надбудова над С. В певному значенні можна назвати C++ покращуваним З, тим З, який забезпечує контроль типів, перевантаження функцій і ряд інших зручностей. Але головне в тому, що C++ додає до З об'єктну орієнтованість.

Відомо декілька версій C++. У версії 1.0 реалізовані основні механізми об'єктно-орієнтованого програмування, такі як одиночне спадкоємство і поліморфізм, перевірка типів і перевантаження функцій. У створеній в 1989 році версії 2.0 знайшли віддзеркалення багато додаткових властивостей, що виникли на базі широкого досвіду застосування мови численним співтовариством користувачів. У версії 3.0 (1990) з'явилися шаблони і обробка виключень. C++ продовжує удосконалюватися і в даний час, так в 1998 році вийшла нова версія стандарту, що містить в собі деякі досить істотні зміни. Мова стала основою для розробки сучасних великих і складних проектів.

Common Lisp Object System (CLOS)

На початку 80-х років під впливом ідей об'єктно-орієнтованого програмування виникла серія нових діалектів Lisp, багато хто з яких був орієнтований на представлення знань. Успіх в стандартизації Common Lisp стимулював спроби стандартизувати об'єктно-орієнтовані діалекти в 1986 році.

Оскільки новий діалект повинен був стати надбудовою над Common Lisp, він отримав назву Common Lisp Object System (Об'єктна система Common Lisp) або, скорочено, - CLOS. Серйозний вплив на проект CLOS зробили мови Newflavors і Commonloops. Після дворічної роботи, в 1988 році була опублікована повна специфікація CLOS.

CLOS повинен бути:

- бути стандартним розширенням мови, що включає всі найбільш корисні властивості існуючої об'єктно-орієнтованої парадигми;
- забезпечити ефективний і гнучкий інтерфейс програміста, що дозволяє реалізувати більшість прикладних завдань;
- проектуватися як розширюваний протокол, так, щоб можна було змінювати його поведінку, тим самим стимулюючи подальші дослідження в області об'єктно-орієнтованого програмування .

Не підтримуючи безпосередньо механізм довготривалих об'єктів, CLOS має розширення з протоколом метаоб'єктів, що реалізовують цей механізм .

Ada

У 1983 році під егідою Міністерства Оборони США була створена мова Ada. Мова чудова тим, що дуже багато помилок може бути виявлено на етапі компіляції. Крім того, підтримуються багато аспектів програмування, які часто віддаються на відкуп операційній системі (паралелізм, обробка виключень). У 1995 році був прийнятий стандарт мови Ada 95, яка розвиває попередню версію, додаючи в неї объекно-ориентированность і виправляючи деякі неточності.

Обидві ці мови не отримали широкого розповсюдження поза військовими і іншими великомасштабними проектами (авіація, залізничні перевезення). Основною причиною є складність освоєння мови і достатньо громіздкий синтаксис.

Безпосередніми попередниками Ada є Pascal і його похідні, включаючи Euclid, Lis, Mesa, Modula і Sue. Були використані деякі концепції Algol-68, Simula, CLU і Alphard.

Розробники Ada перш за все турбувалися про:

- надійності і експлуатаційних якостях програм;
- програмуванні як різновиді людської діяльності;
- ефективності.

Eiffel

Автор Eiffel Бертран Мейер (Bertrand Meyer) створював не тільки мову об'єктно-орієнтованого програмування, але і інструмент проектування програм.

Не дивлячись на сильний вплив Simula, Eiffel - цілком самостійна об'єктно-орієнтована мова зі своїм власним середовищем розробки. Eiffel підтримує динамічне скріплення і статичну типізацію, тим самим забезпечуючи гнучкість інтерфейсів класів у поєднанні з безпечним використанням типів. У Eiffel є декілька важливих рис, що підтримують жорсткіший стиль програмування, зокрема класи, що параметризуються, твердження і виключення. Мейєр вважає, що узагальнені класи добре доповнюють спадкоємство, враховуючи горизонтальний рівень спільності; нові класи на одному рівні ієрархії можна створювати, використовуючи тип як параметр, а не створюючи практично однакові підкласи.

Невід'ємною частиною мови є умови поста, тобто твердження, які повинні виконуватися при вході в метод і виході з нього. Порушення твердження викликає виняткову ситуацію. Її можна перехопити, обробити і спробувати викликати той же метод ще раз.

Eiffel заохочує хороше програмування, добротну специфікацію класів,

сильну типізацію і повторне використання, як через спадкоємство, так і через параметризацію. Формальне трактування виняткових ситуацій дозволяє жорстко специфікувати інтерфейси класів при реалізації.

Eiffel надає закінчене середовище розробки програм, включаючи спеціальний редактор з виділенням синтаксису, генератор документації, бібліотеки класів і броузер. Крім того, підтримуються засоби управління кодом і збіркою програм.

Java

З 1995 року почала широко розповсюджуватися нова об'єктно-орієнтована мова програмування Java, орієнтована на мережі комп'ютерів і, перш за все, на Internet. Синтаксис цієї мови нагадує синтаксис мови C++, проте ці мови мають мало загального. Java мова, що інтерпретується: для неї визначено внутрішнє уявлення (bytecode) і інтерпретатор цього уявлення, які вже зараз реалізовані на більшості платформ. Інтерпретатор спрощує відладку програм, написаних на мові Java, забезпечує їх переносимість на нові платформи і адаптується до нових оточень. Він дозволяє виключити вплив програм, написаних на мові Java, на інші програми і файли, що є на новій платформі, і тим самим забезпечити безпеку при виконанні цих програм. Ці властивості мови Java дозволяють використовувати його як основна мова програмування для програм, поширюваних по мережах (зокрема, по мережі Internet).

Object Pascal

Object Pascal створювався співробітниками компанії Apple Computer (деякі з яких були учасниками проекту Smalltalk) спільно з Никлаусом Віртом (Niklaus Wirth), творцем мови Pascal. Object Pascal відомий з 1986 року і є першою об'єктно-орієнтованою мовою програмування, яка була включена в Macintosh Programmer's Workshop (MPW), середовище розробки для комп'ютерів Macintosh фірми Apple.

У цій мові немає методів класу, змінних класу, множинного спадкоємства і метаклассов. Ці механізми виключені спеціально, щоб зробити мову простою для вивчення початкуючими "об'єктними" програмістами.

Система візуального об’єктна-орієнтовного проектування Delphi.

Поява Delphi не могло пройти не поміченим серед багатьох користувачів комп’ютерів. Оцінки експертів, які вивчали можливості цього нового продукта фірми Borland, зазвичай схвального типу. Основне достоїнство Delphi є в тому, що тут реалізована ідея візуального програмування. Середовище візуального програмування перетворює процес створення програми в задоволення і легко зрозумілий конструйований додаток з великого набору графічних і структурних примітивів.

Система Delphi дозволяє вирішувати багато задач, зокрема:

- Створювати закінченні додатки для Windows самого різного напрямку, від обчислювальних і логічних, до графічних і мультимедійних.
- Швидко створювати (навіть початківцям) віконний інтерфейс з професійним виглядом для різних додатків.
- Створювати потужні системи роботи з локальними і віддаленими базами даних.
- Створювати довідкові системи (файли .hlp) для своїх додатків і багато іншого.

Delphi – система, яка досить швидко розвивається. Перша версія - Delphi 1.0 була видана в лютому 1995 році. А потім нові версії випускалися щороку.

Кожна наступна версія Delphi доповнювала попередню. Більшість версій Delphi видається в декількох варіантах: Standart – стандартному, Professional – професійному, Client\Server – клієнт\сервер, Enterprise – розробка баз даних предметних областей. Останні варіанти – Client\Server і Enterprise, в цьому відношенні найбільш потужні.

Висновки

Об'єктно-орієнтовані системи дають ширший спектр багатократного використання текстів програм. Бібліотек об'єктів також можна набувати від незалежних постачальників. В даний час найактивніше купують такі бібліотеки класів для створення призначених для користувача інтерфейсів з піктограмами. Розробка і написання таких інтерфейсів з нуля - завдання нелегке. Компанії типу Apple і Whitewater Group поставляють інструментарії для швидкої побудови таких інтерфейсів на основі декількох базових класів типу Window, Menu, Scrollbar і Icon. Користувачі можуть використовувати як ці класи, так і їх підкласи, що додають в інтерфейс, наприклад, спеціальні піктограми.

ООС легко підтримуються. Четверта перевага полягає в способі комплектування об'єктно-орієнтованих програмних модулів. Традиційне ПО складається з даних і процедур, здійснюючий доступ і зміну даних. Дані і процедури комплектуються окремо, тому зміна структури даних впливає на різні модулі, написані різними користувачами. У об'єктно-орієнтованій системі дані і процедури розглядаються разом як частина одного пакету - об'єкту. При зміні даних всі задіяні процедури легко ідентифікуються і змінюються одночасно. Оскільки зміна розповсюджується тільки на одну область системи, його побічний вплив на всю систему зменшується.

Унаслідок цих переваг, а також ще ряду причин, ООП є в даний час найперспективнішим, поширенішим і ефективнішим напрямом в програмуванні.

Використана література:

1. Г. Буч «Объектно-ориентированный анализ и проектирование с примерами приложений на C++» Пер. с англ. - М.: Бином; СПб.: Невский диалект, 1999.
2. В. Фаронов «Deiphi 6» - СПб.: Питер, 2002.
3. Э. Ишкова «C++ начала программирования» - М.: Бином, 2001.
4. С. Немнюгин, Л. Перколаб «Изучаем Turbo Pascal» - СПб.: Питер, 2002.