Лекція 1

Тема лекції: абстракція, інкапсуляція, спадкоємність та поліморфізм. Об'єкти в світлі абстракції та інкапсуляції. Створення об'єктів та ініціалізація даних стану. Протокол опису класу: поля даних і функції-члени

Основні визначення ООП

Існує 5 ключових компонентів будь-якої об’єктно-орієнтованої мови програмування, в тому числі і мови C++. Це є наступні поняття:

• Об’єкт – інкапсульована абстракція, яка містить інформацію про стан та чітко визначену множину протоколу доступу (повідомлення, котрі обробляє об’єкт).

• Повідомлення – це спеціальний символ, ідентифікатор чи ключове слово з параметрами чи без них, яке представляє дію, яку виконує об’єкт.

• Клас – це визначений тип об’єктів. Він задається за допомогою опису класу, де визначаються змінні стану та протокол доступу до об’єктів даного класу. Класи можуть мати ієрархічну або контейнерну організацію.

• Екземпляр об’єкту. Всі об’єкти належать якомусь класу. Властивості екземпляра об’єкту визначаються описом класу.

• Метод. Метод існує для кожного повідомлення, визначеного для якогось класу. Метод визначає реакцію об’єкта на повідомлення. Зазвичай складається з низки виразів і може використовувати протокол іншого класу.

Таким чином, всі об’єкти належать до якихось класів. Об’єкти обробляють повідомлення згідно з методами, що були визначені в описі класу. Ці змінні можуть мати однакові або різні значення в різних екземплярах класу.

В мові C++ прийнята наступна термінологія:

• Клас – ключове слово class – новий тип даних (розширення структурного типу даних).

• Об’єкт – змінна типу ClassName, де ClassName – визначений раніше клас.

• Дані стану – закриті дані або змінні екземпляру об’єкта. Оголошується в описі класу і називаються даними-члени класу.

• Повідомлення вказується за допомогою прототипів функцій в описі класу. Прототипи функцій містять тип повертаємого значення, ім’я функції та список параметрів. Список параметрів повинен містити типи параметрів та необов’язкові імена параметрів.

• Метод – це визначення (реалізація) функцій. Прототипи функцій та їх визначення разом – це є повідомлення, які може обробляти об’єкт. Разом вони називаються функціями-членами класу. До членів класу належать функції-члени та дані-члени.

Абстракція, інкапсуляція, спадкоємність та поліморфізм

• Абстракція – формування уявлення про якості та властивості предмета шляхом уявного видалення якихось матеріальних об’єктів.

• Інкапсуляція – результат схову в капсулі, розміщення в обмеженому просторі. Одиниця інкапсуляції – це об’єкт. Капсула чи об’єкт, містить у собі дані стану та протоколи повідомлень, які обробляє об’єкт.

• Спадкоємність – це передача властивостей, що були отримані від попередника. Отримуються дані стану та протоколи повідомлень з протоколу батьківського класу.

• Поліморфізм – це умови, в яких вид має дві різні морфологічні форми. Це означає, що окремі повідомлення можуть викликати різні дії на етапі виконання програми. Конкретна форма реалізації повідомлення визначається та зв’язується з об’єктом під час виконання програми.

Об’єкти в світлі інкапсуляції та абстракції

Об’єкт утримує в собі значення, які відображають його внутрішній стан. Це змінні стану для об’єктів даного класу. Крім того, він утримує в собі дані про засіб обробки повідомлень, котрі до нього надходять. В об’єкті інкапсульовані всі властивості абстракції, включаючи значення абстракцій даних та функціональні абстракції (повідомлення та методи). Дві основні компоненти об’єкту як інкапсуляції абстракцій задаються за допомогою визначення абстракцій (поля даних та визначення функцій-членів, заданих в протоколі опису класу) та внутрішніх станів (фактичні значення полів даних для конкретного об’єкту).

Для об’єктів бажані такі властивості:

• *Чітка межа*. Вона визначає область бачення інкапсуляції. В C++ область бачення інкапсуляції для об’єкта поширюється на опис його класу та опис батьківських класів.

• *Добре розроблений інтерфейс*. Інтерфейс описує те, як саме об’єкт взаємодіє з іншими об’єктами чи сегментами програми. В C++ він задається за допомогою прототипів функцій-членів. Інші об’єкти можуть здійснювати доступ до об’єкту тільки через такий контролюємий інтерфейс.

• *Захищене внутрішне представлення*. Внутрішне представлення об’єкта невидиме для інших об’єктів і не може бути ними знищено. В C++ цей захист є гнучким і може управлятися завдяки використанню концепції закритих, захищених та відкритих секцій опису класу. Задля тієї ж мети використовується принцип дружніх функцій та класів.

Класи і об'єкти

Термін “об'єкт” в програмному забезпеченні вперше введений в мові SIMULA і позначав будь-який аспект моделюємої реальності.

Об'єкт - це особливий предмет, блок, що розпізнається або сутність (реальна або абстрактна), що має важливе функціональне призначення в даній предметній області. Тобто об'єкт — це щось, що має межі. Існують такі об'єкти, для яких визначені явні межі, але самі об'єкти є неосяжними подіями або

процесами. Наприклад, хімічний процес. Його межі явно визначені взаємодією компонентів, котрі протягом певного часу поводяться відомим чином. Або — лінія перетину сфери і куба. Хоча ця лінія не існує окремо від сфери і куба, вона залишається самостійним об'єктом і її межі чітко визначені. Об'єкт володіє станом, виявляє чітко висловлену поведінку і індивідуальність. Об'єкти можуть бути помітними, але мати розмиті фізичні межі. Наприклад: ріки, туман, натовпи людей.

Висновок: об'єкт володіє станом, поведінкою, індивідуальністю; структура і поведінка подібних об'єктів визначає загальний для них клас; терміни “примірник класу” і “об'єкти” взаємозамінні.

Стан об’єкту

Стан об'єкту характеризується переліком всіх можливих властивостей даного об'єкту (зазвичай статичних) і поточними значеннями кожного з цих властивостей (зазвичай динамічними).

До числа властивостей об'єкту відносяться притаманні йому і надбані ним характеристики, риси, якості або можливості, що роблять даний об'єкт самим собою.

Всі властивості об'єкту характеризуються значеннями їхніх параметрів. Ці значення можуть бути простими кількісними характеристиками, а можуть означати інший об'єкт.

Всередині кожного об'єкту зберігаються в захищеному вигляді елементи, що відбивають його стан, і стан всієї системи в цілому, розподілений між об'єктами.

Поведінка

Поведінка характеризує те, як об'єкт впливає або підлягає дії інших об'єктів з точки зору зміни стану цих об'єктів і передачі повідомлень.

Певний вплив одного об'єкту на інший з метою викликати відповідну реакцію називається операцією. Наприклад, об'єкт-користувач може активізувати операції *add* і *pop* для того, щоб дати черговому об'єкту приріст або скоротити його.

У об’єктних і об’єктно-орієнтованих мовах операції, що виконуються над даним об'єктом, називаються *методами* і входять складовою частиною до опису класу. В C++ з цією метою використовується також термін *функція-елемент* або *функція-член*.

Існує п'ять основних видів операцій над об'єктами:

1.Модифікатор — операція, що змінює стан об'єкту шляхом запису або доступу.

2.Селектор — операція, що дає доступ для визначення стану об'єкту без його зміни (операція читання).

3.Ітератор — операція доступу до змісту об'єкту по частинам (в певній послідовності).

4.Конструтор — операція створення і ініціалізації об'єкту.

5.Деструктор — операція руйнування об'єкту і звільнення пам'яті, яку вони займали.

Останні два види операцій застосовуються в мові C++. Таким чином, будь-який метод є операцією, але не будь-яка операція є методом.

Сукупність всіх методів і загальнодоступних процедур, які стосуються конкретного об'єкту, утворюють *протокол* цього об'єкту. Таким чином *протокол об'єкту* визначає оболонку поведінки об'єкту, що охоплює його внутрішній статичний і зовнішній динамічний прояв.

Об'єкти як автомати

Наявність внутрішнього стану об'єкту означає, що порядок виконання операцій має істотне значення. Це дозволяє представити об'єкт в якості складного незалежного автомату. Об'єкти можуть бути активними і пасивними. Активним є той об'єкт, який розміщений в каналі управління. Активний об'єкт

автономний, тобто він може реалізувати свою поведінку без впливу з боку інших об'єктів. Пасивний об'єкт може змінювати свій стан тільки під впливом інших об'єктів. Таким чином, активні об'єкти системи складають канал управління. Якщо система має декілька каналів управління, то і активних об'єктів може бути безліч.

Властивості об'єкту, що відрізняють його від всіх інших об'єктів, називаються індивідуальністю. В більшості мов програмування для відмінності тимчасових об'єктів, їхньої взаємної адресації і ідентифікації використовуються імена змінних. Джерелом багатьох помилок в об’єктно-орієнтованому програмуванні є неможливість відокремити ім'я об'єкту від самого об'єкту.

Час існування об'єктів

Початком часу існування будь-якого об'єкту є момент його створення (відведення ділянки пам'яті), а закінченням — момент вилучення відведеної ділянки пам'яті. В мові C++ створення об'єкту автоматично призводить до виклику конструктора для даного класу об'єктів. В C++ можна перевизначити засоби створення об'єктів. Для створення об'єктів в вільній ділянці пам'яті застосовується процедура *new*. Об'єкт продовжує існувати доти, доки він займає місце в пам'яті, навіть якщо буде втрачене посилання на цей об'єкт. Об'єкти, створені в програмному стеку, ліквідуються автоматично при переході управління за межі області дії змінної, зв'язаної з даним об'єктом. При ліквідації об'єкту автоматично викликається деструктор.

Відношення між об'єктами

Самі по собі об'єкти не представляють ніякого інтересу, тільки в процесі взаємодії об'єктів між собою реалізується мета системи. Відношення двох будь-яких об'єктів грунтуються на припущенні, що кожний об'єкт має інформацію про інший об'єкт, про операції, які над ним можна виконувати і про очікувану поведінку. Існують два типи ієрархічних співвідношень об'єктів:

1.Відношення використання.

2.Відношення включення.

Відношення використання. При використанні одного об'єкту іншим, вони обмінюються повідомленнями. Пересилання повідомлень між об'єктами однонаправлене, але можливі і двонаправлені зв’язки. Кожний об'єкт, включений у відношення використання, може виконувати наступні три ролі:

1.*Вплив.* Об'єкт може впливати на інші об'єкти, але сам ніколи не підлягає впливу інших об'єктів; в певному сенсі відповідає поняттю активний об'єкт.

2.*Використання.* Об'єкт в цьому випадку може тільки підлягати управлінню з боку інших об'єктів, але ніколи не виступає в ролі об'єкту, що впливає на інші.

3.*Посередництво.* Такий об'єкт може виступати як в ролі того, що впливає, так і в ролі виконавця. Як правило, об'єкт-посередник створюється для виконання операцій в інтересах якогось активного об'єкту або іншого посередника.

Поняття синхронізації

При передачі повідомлень від одного об'єкту до іншого обидва взаємодіючих об'єкти повинні певним чином синхронізовуватись. Для послідовних систем така синхронізація реалізується через виклики підпрограм. В паралельних системах для багатоканального управління необхідно вирішити проблему виняткових ситуацій. Звідси випливає ще один спосіб класифікації об'єктів.

1.Об'єкт-транслятор — пасивний об'єкт, що має тільки один канал управління.

2.Блокований об'єкт — пасивний об'єкт, що має декілька каналів управління.

3.Паралельний об'єкт — активний об'єкт, що має декілька каналів управління.

Відношення включення. Об'єкти перебувають у відношенні включення, якщо одні об'єкти є елементами стану інших об'єктів.

Включення одних об'єктів в інші прийнятніше використовувати в тому плані, що при цьому зменшується число об'єктів з якими потрібно оперувати на даному рівні опису. З іншого боку, використання одних об'єктів іншими має перевагу: не виникає сильної залежності між об'єктами.

Поняття "клас"

В той час, як об'єкт означає конкретну сутність, певну в часі і просторі, клас визначає лише абстракцію, "вижимку" з об'єкту. Отже, з точки зору об'єктно-орієнтованого програмування, визначенням класу є: "клас — множина об'єктів, зв'язаних спільністю структури і поведінки". Будь-який об'єкт є просто примірником класу. Об'єкт не є класом, хоча клас може бути об'єктом. Об'єкти, які не зв'язані спільністю структури і поведінки, не можуть утворити клас.

В той час, як об'єкт існує конкретно і грає певну роль, клас містить опис структури і поведінку всіх об'єктів, зв'язаних відношенням спільності. Тобто клас виконує роль угоди про зв'язки в відношеннях абстракції і всіх її реалізацій. Це досягається за рахунок того, що в інтерфейсній частині описи класу містяться необхідні дані про проектні рішення. Інтерфейсна частина опису класу відповідає його зовнішньому прояву, підкреслює його абстрактність, але приховує структуру і особливості поведінки. Інтерфейсна частина класу складається з переліку дій, що припускає опис інших класів, констант, змінних і особливостей, необхідних для повного визначення даної абстракції.

Реалізація класу складає його внутрішню поведінку і визначає особливості поведінки. В цій частині розкривається реалізація тих операцій, які перераховані в інтерфейсній частині опису. Інтерфейсна частина опису класу може бути поділена на 3 складові частини: загальнодоступна, захищена, відокремлена.

Структура стану об'єкту також вимагає певного опису. Він полягає в оголошенні констант і змінних у відокремленій частині опису інтерфейсу. Це робить загальну частину структури об'єктів закритою для доступу і внесення до цієї частини змін, не відбивається на функціонуванні об'єктів-користувачів. В C++ допускаються такі визначення як у відокремленій так і в загальнодоступній частині інтерфейсу класу.

Взаємозв'язок класів і об'єктів. Відношення між класами та об'єктами

В більшості практичних випадків класи статичні, тобто всі їхні особливості і зміст визначені в процесі компіляції програми. З цього випливає, що будь-який створений об'єкт належить до фіксованого класу. Об'єкти, навпаки, в процесі виконання програми безупинно створюються і руйнуються.

На етапі аналізу і ранніх стадіях проектування вирішуються дві основні задачі:

- виявлення класів і об'єктів, що складають словник предметної області;

- побудова структур, що забезпечують спільну взаємодію об'єктів, при якій досягаються задані вимоги.

В першому випадку кажуть про ключові абстракції задачі, а в другому — про механізми реалізації.

В першій фазі проекту увага проектувальника зосереджується на зовнішніх проявах ключових абстракцій і механізмів. Такий підхід створює логічний каркас системи, яка складається з структури класів і структури об'єктів. На наступних фазах проекту, включаючи і реалізацію, увага переключається на внутрішню поведінку ключових абстракцій і механізмів і охоплює питання їхнього фізичного подання. Рішення, що приймаються в процесі проектування складають архітектуру модулів і архітектуру процесів в системі.

Для оцінки якості класів і об'єктів, що виділяються в системі існує 5 критеріїв:

1.Взаємозалежність.

2.Зв’язанність.

3.Достатність.

4.Повнота.

5.Простота.

Взаємозалежність — це ступінь глибини зв'язків між окремими модулями. Фрагменти системи, які сильно залежать від інших, значно складніше сприймати, замінити і модифікувати. Для поліпшення якості системи слід за можливістю уникати сильної залежності між окремими модулями.

Зв’язаність — це ступінь взаємодії між елементами окремого модуля, класу або об'єкту. Найменш бажаною є зв’язаність за випадковою ознакою, коли в одному модулі збираються цілком незалежні абстракції. Бажаною є функціональна зв’язаність, при якій всі елементи модуля тісно взаємодіють для досягнення мети.

Достатність — наявність в класі або модулі програми всього необхідного для реалізації логічної і ефективної поведінки. Тобто компоненти повинні бути цілком придатні до використання. Наприклад, розглянемо клас множина. В цей клас необхідно включити операції вилучення і додавання елементу. Буде помилкою не включити в клас одну з означених операцій. Порушення вимог достатності виявляється, як тільки створюється клас-користувач.

Повнота — наявність в інтерфейсній частині класу всіх необхідних характеристик абстракції. Ідея достатності пред’являє до інтерфейсу мінімальні вимоги, а ідея повноти охоплює всі істотні аспекти абстракції. Повнота є суб'єктивним чинником і розробники часто виносять на верхній рівень такі операції, що можна реалізувати на більш низькому рівні. З цього випливає вимога простоти.

Простими є тільки такі операції, що забезпечують реалізацію ефективної дії абстракції. В прикладі з множинною операція додавання елементу є примітивною, а операція додавання 4-х елементів не буде примітивною, бо ефективно реалізується через операцію додавання одного елементу.