**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

*Кафедра “Системи автоматизованого проектування”*



**Звіт**  
до розрахунково-графічної роботи  
з курсу: «Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні»  
на тему:  
**«Open BEAGLE»**

Виконав:  
ст. гр. КНСП-11

Дацко Л. А.

Перевірив:  
Кривий Р. З.

# Вступ

Open BEAGLE - це еволюційна система обчислення (EC), повністю закодована в C ++. Рекурсивний акронім BEAGLE означає, що Beagle Engine є Advanced Genetic Learning Environment. Beagle був також назвою англійського судна, на якому Чарльз Дарвін приступив як натураліст до свого знаменитого кругосвітнього плавання. Назва Beagle раніше використовувалася в 1980-х роках для програмного забезпечення для розпізнавання образів на основі принципів EС (Forsyth, 1981). У наш фреймворк було додано прикметник Open, щоб відрізняти його від цього програмного забезпечення, а також наполягати на аспекті проекту з відкритим вихідним кодом.

Архітектура Open BEAGLE слідує принципам об'єктно-орієнтованого (OO) програмування, де деякі абстракції представляються дуже щільно об'єднаними об'єктами і легким для повторного використання кодом. Open BEAGLE був розроблений з метою забезпечення загального, зручного для користувача, портативного, ефективного, надійного, елегантного та вільного для EC фреймоворка.

**Узагальненість** у Open BEAGLE користувач може виконувати будь-який тип EC, якщо він виконує деякі мінімальні вимоги. Необхідною умовою є наявність популяції осіб, до якої послідовно застосовується послідовність операцій, що розвиваються. До цих пір було реалізовано два спеціалізовані фреймворки з використанням Open BEAGLE: генетичних алгоритмів (GA) і генетичного програмування (GP). Планується також розробка еволюційних стратегій (ES). Користувач може прийняти будь-який з цих спеціалізованих фреймворків і модифікувати їх далі, щоб створити свій власний спеціалізований атрибут еволюційних алгоритмів.

**Зручність** для користувачів Open BEAGLE надає декілька механізмів, які забезпечують зручний інтерфейс для користувача. Наприклад, управління пам'яттю динамічно розподілених об'єктів значно спрощується за рахунок використання підрахунку посилань і автоматичного збору сміття.

**Переносимість** код Open BEAGLE відповідає стандарту C ++ ANSI / ISO 3. Для цього потрібна бібліотека стандартних шаблонів (STL) (Musser and Saini, 1996). На операційну систему або на апаратне забезпечення не здійснюються спеціальні виклики.

**Ефективність** щоб забезпечити ефективне виконання, особлива увага приділяється оптимізації критичних засад. Детальне виконання протоколів було зроблено. Крім того, той факт, що Open BEAGLE написаний на мові C ++, сприяє його загальній ефективності.

**Надійність** багато операторів перевірки вбудовані в код, щоб забезпечити правильну роботу і повідомити користувача про проблему. Надійні механізми для періодичного збереження поточного стану еволюції також були реалізовані для того, щоб дозволити автоматичний перезапуск перерваних еволюцій.

**Елегантність** інтерфейс Open BEAGLE розроблений з турботою. Великі зусилля були вкладені у розробку узгодженого програмного пакета, що відповідає хорошим принципам ОО та загального програмування. Крім того, були застосовані суворі правила програмування, щоб зробити код C ++ легким для читання, розуміння та модифікації.

**Вільний доступ** вихідний код Open BEAGLE є безкоштовним, доступним під GNU Lesser GeneralPublicLicense (LGPL) (FreeSoftwareFoundation Inc., 2000). Таким чином, він може бути поширений і модифікований без будь-якої оплати. Open BEAGLE доступний в Інтернеті за адресою http://www.gel.ulaval.ca/~beagle.

**Огляд існуючого програмного забезпечення EС**

Незважаючи на те, що за останні десять років розроблено багато систем програмного забезпечення EС, лише деякі з них є достатньо загальними для реалізації різних пріоритетів алгоритмів. Тепер ми розглянемо три найважливіші з них, які були розроблені з використанням парадигм програмування OO, і які ми вважаємо порівнянними з Open BEAGLE: gpc ++ (Fraser, 1994), EO (Merelo et al., 2001) і ECJ (Luke, 2001).

gpc ++ є одним з перших відомих C ++ фреймворків для GP2 на основі дерева. Хоча gpc ++ і Open BEAGLE поділяють деякі філософські аспекти та аспекти реалізації, gpc ++ містить C-подібні конструкції, які не сприяють гарним практикам OO. Наприклад, дерево GP побудовано у вигляді попереднього списку імен функцій і терміналів (список символів \*). Для оцінки дерев користувачеві необхідно визначити складний випадок перемикання, який дуже важко переробити. Крім того, gpc ++ не поширюється на шаблон дизайну (Lenaerts and Manderick, 1998).

EO широко використовує загальне програмування (іноді називається статичним поліморфізмом), на відміну від Open BEAGLE, яке використовує деякі загальні концепції програмування для поліпшення користувацького досвіду, але в основному використовує поліморфізм у спадок (іноді званий динамічний поліморфізм). Всі еволюційні процеси і параметри в ЕО повинні бути зафіксовані під час компіляції. Це може підвищити продуктивність, але, звичайно, погіршує досвід користувачів і знижує загальну доступність архітектури. На момент написання статті EO реалізує GA і ES. Також доступна експериментальна реалізація GP.

ECJ є повноцінним, заснованим на Java середовищем для ЕК. Ми вибрали ECJ серед усіх доступних систем на базі Java, оскільки це повнофункціональна, добре створена система OO. Як і Open BEAGLE, він розроблений за методологією OO і широко використовує поліморфізм шляхом успадкування. Однак, незважаючи на те, що Java є аналоговою мовою, її процеси з відносно низькою швидкістю виконання, що дійсно є серйозним обмеженням для процесорів, які потребують інтенсивного використання, такі як ЕК.

**Архітектура програмного забезпечення**

Open BEAGLE має трирівневу архітектуру. Основи розташовані внизу, як розширення OO мови C ++ і STL. Основна структура EC побудована на цих основах і складається з функцій, спільних для всіх еволюційних алгоритмів (EA). Нарешті, різні незалежні модулі спеціалізуються на цій загальній структурі, кожен модуль реалізує конкретний EA.

**Об’єктно-орієнтоване застосування**

У Open BEAGLE всі класи виводяться з абстрактного класу Object. Як повний об'єкт C ++, об'єкт Open BEAGLE - це інтерфейс, який включає в себе набір загальних операцій. Об'єкт складається з функцій порівняння, вхідних / вихідних функцій і функцій для взаємодії з опорним лічильником. Використання лічильників посилань і розумних покажчиків спрощує управління динамічно розподіленими об'єктами. Дійсно, кожен об'єкт володіє лічильником, який відстежує кількість існуючих посилань на себе. Розумний покажчик поводиться як стандартний покажчик C ++, але також збільшує і зменшує лічильник опорних об'єктів, коли це необхідно. Коли довідковий лічильник об'єкта дорівнює нулю, об'єкт звільняється. Цей механізм дозволяє емуляцію процесу збору сміття, подібно до тих, які можна знайти в мовах OO вищого рівня. Інтелектуальний покажчик асоціюється з кожним типом об'єкта. Це асоціаціявиділяєдекларингауправління членомтипу у всіх класах Open BEAGLE. Наприклад, інтелектуальний покажчик, асоційований з типом MyClass, буде мати тип MyClass :: Handle. Крім того, якщо між двома класами існує взаємозв'язок успадкування, між двома типами членів Handle цих класів існує однакове співвідношення успадкування. Наприклад, якщо клас DerivedClass успадковується від класу BaseClass, то тип члена DerivedClass :: Handle успадковує frommembertype BaseClass :: Handle. Цей дуже важливий механізм в BEAGLE дозволяє типу члена Handle ідеально моделювати стандартні C ++ покажчики, навіть у контексті поліморфізму шляхом успадкування. Цей механізм реалізується за допомогою шаблонів, які виконують більшу частину роботи. Програміст, який створює нові потреби підкласу, тільки оголошує typedef, використовуючи цей шаблон. Open BEAGLE використовує поліморфізм успадкуванням, що означає, що екземпляри об'єктів повинні бути динамічно виділені. Більш того, важко копіювати або клонувати даний об'єкт, коли його точний тип невідомий. Таким чином, були реалізовані розподільники, тобто фабрики об'єктів, які можуть виділяти, клонувати та копіювати даний тип даних. Існує розподільник, названий Alloc, асоційований з кожним класом Open BEAGLE. Наприклад, клас з ім'ям MyClass включає тип MyClass :: Alloc, який можна використовувати для виділення, копіювання та клонування об'єктів типу MyClass. Загальний контейнер для об'єктів також був включений в основу OO Open BEAGLE: сумку. Сумка є таблицею випадкового доступу для об'єктів BEAGLE. Вона реалізована як динамічний масив розумних покажчиків (тобто std :: vector <Object :: Handle>). Таким чином, це контейнер, яким можна керувати загальним алгоритмом

rithmsofSTL.SinchebeisalsoAEBOOLEоб'єкт, його можна віднести до розумних покажчиків, і можна створити мішки мішків. Так само, як для розумних покажчиків і розподільників, з кожним об'єктом асоціюється тип члена мішка. Наприклад, для типу MyClass, асоційований тип мішка - MyClass :: Bag. Більш того, колинаслідуєтьсявзаємозв'язокміж двома класами, вона також дотримується їх вбудованими мішками типів членів, точно так само як і для ручки і Alloc. Введення і виведення даних також є важливим аспектом будь-якої структури високого рівня. Мова C ++ інтегрує потоки вводу-виводу, які дозволяють об'єктам вставлятися і вилучатися з таких носіїв, як файли, консоль або пам'ять. Поєднуючи концепцію потоку вводу-виводу C ++ з форматом XML, ми розробили потоки вводу-виводу XML. XML (eXtensible Markup Language (Anderson et al., 2000)) є ідеальною мовою для представлення даних, оскільки вона є гнучкою, стандартизованою, читається людьми і легко редагується. Вставка та вилучення даних у Open BEAGLE XML-потоки ідентичні вставці та вилученню даних у стандартні потоки C ++. Оператори << і >> визначені для класу Object так само, як вони визначені для атомарних типів C ++. Їх виклики просто зіставляються з асоційованими віртуальними членами читання і запису. Уцих функціях компоненти класу вставляються і витягуються за допомогоювикликів, що відповідають їх операторам вставки і вилучення. На додаток до цих операторів, деякі функції інтерфейсу-члена визначаються в класі потоку XML для вставки і вилучення тегів і атрибутів XML.

**Приклад застосування**

У менш ніж ста рядках коду, цей розділ представляє реалізацію простої проблеми GP з Open BEAGLE. Мета полягає в тому, щоб продемонструвати простоту використання середовища, і в той же час дати додатковий погляд на його філософію програмування. Проблема полягає в класичній символічній регресії простої поліноміальної функції p (x) = x4 + x3 + x2 + x (Koza, 1992). Обговорення слідує підходу «зверху вниз». По-перше, основна функція розкрита, а потім реалізація примітиву, і, нарешті, реалізація оператора оцінки логічності. Випливає п'ять кроків, необхідних для реалізації будь-якої програми GP з Open BEAGLE:

1. Створіть примітиви і вставте їх у набір примітивів (рядки 8-12). Тут ми визначаємо один набір з використанням вбудованих арифметичних функцій; Проблема x також реалізована як примітив (рядок 13, токен - це просто ім'я, загорнуте в примітив)

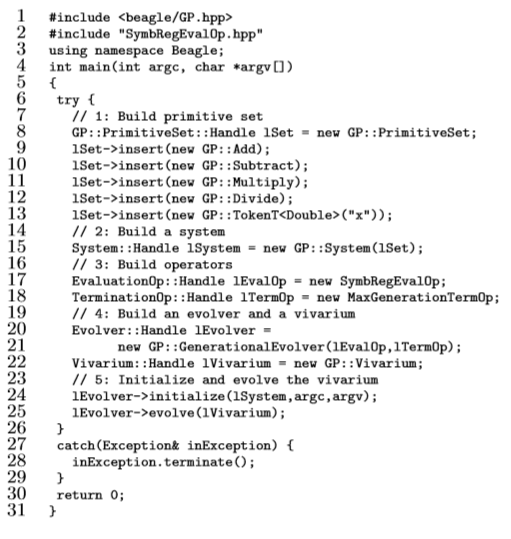
2. Побудова еволюційної системи ГП з використанням примітивного набору (ів) (рядок 15) ;

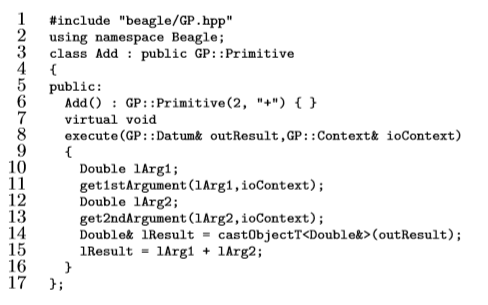
3. Побудувати операторів оцінки та припинення (рядки 17 і 18). Тут екземпляр класу SymbRegEvalOp використовується як оператор оцінкиТакож виділено екземпляр класу MaxGenerationTermOp, навіть якщо цей крок дійсно не потрібний, оскільки це оператор завершення за замовчуванням.

4. Створіть еволюцію за допомогою операторів попереднього кроку (рядки 20 і 21). Тут ми використовуємо стандартний еволюцію поколінь, що вимагає лише aоператор фітнесу з додатковим оператором припинення. Також будують віварій ГП (лінія 22). Важливою особливістю Open BEAGLE, що не проілюстровано в цьому простому прикладі, є можливість для користувача побудувати віварій спеціалізованих дім, індивідуумів або навіть генотипів. Ця функція реалізована за допомогою використання розподільників (за замовчуванням конструктор віварію у рядку 22 використовує розподільники для демов, індивідуумів і генотипів за замовчуванням).

5. Запустіть еволюцію (рядок 25) після ініціалізації (рядки 24). Це метод evolver :: initialize, який інструктує всіх операторів ініціалізувати себе, тобто реєструвати всі їх параметри в регістрі, перш ніж перейти до аналізу командного рядка (argc, argv аргументи). Зазвичай в командному рядку вказується файл конфігурації XML, який, у свою чергу, обробляється для перевизначення параметрів оператора за замовчуванням, визначених користувачем.

Зауважимо, що всі екземпляри об'єктів динамічно виділяються за допомогою нового оператора C ++, і посилаються на смарт-покажчики. Це рекомендована процедура в Open BEAGLE. Наступним кроком є ​​оголошення типу даних. Для проблеми з символічною регресією дані є номерами опорних точок. У простому C ++ ми використовуємо атомний тип double. Але в Open BEAGLE дані повинні бути отримані з класу Object.





# Висновок

Open BEAGLE побудований на міцних фундаментах OO, що посилює мову C ++ і STL, щоб забезпечити міцну і гнучку основу для еволюційних обчислень. Фреймворк ЄС складається з загального ядра, що поєднує структуру населення, систему еволюції, оператори. Специфічні фреймворки дозволяють використовувати різні еволюційні алгоритми. Open BEAGLE, таким чином, є універсальним, простим у використанні, портативним, ефективним, надійним, елегантним і вільним середовищем C ++ для проектування складних систем EC.