國立台北大學資訊工程學系專題報告
基於 **LLVM** 的新型態物件導向語言編譯器開發
New type of object-oriented programming language Compiler Development based on LLVM

專題組員:劉又瑄、宋紘陞、陳文彥、楊承恩

專題編號:PRJ-NTPUCSIE-112-002
執行期間:112年09月 至113年06月

1. 摘要

探討基於 LLVM 和 PEGTL 的新型態物件導向語言編譯器開發。LLVM 提供了強大的編譯器基礎架構，而 PEGTL 則提供了靈活的語法分析工具。我們討論了在編譯器開發中的技術挑戰，並提出了解決方案和技術策略。通過實驗結果驗證了提出方案的有效性，同時探討了未來研究的方向。

2. 簡介

藉由開發基於 LLVM 的編譯器嘗試針對物件導向程式設計的概念進行編譯器層面的改進。目標是設計出更加符合現代實用的程式語言並研究如何利用 LLVM 的功能實現在編譯器中，使用 PEGTL 解決語法分析問題。完成本專題後，我們期望加深對 LLVM 技術的理解及反思語言設計模型的利弊。同時，將展示基於這些技術的編譯器開發的實用性和有效性，促進編譯器技術的發展和應用，為軟體開發者提供有價值的資源及對軟體設計模式的思考。

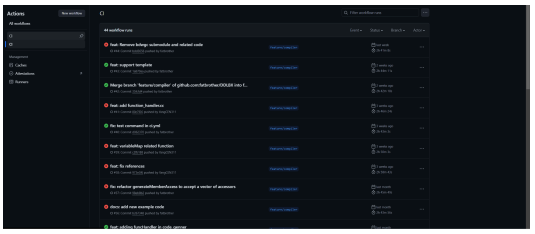
3. 專題進行方式

3-1. 基礎環境及依賴:

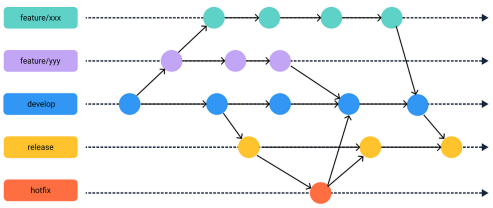
我們利用 GitHub 的專案管理功能來設定明確的里程碑和時間表，以便團隊成員隨時了解任務的狀態。

在開發的同時我們也會同步進行測試，藉由 GitHub 中的 action flow 實施持續整合 CI（Continuous Integration），包括單元測試、集成測試和系統測試等。

通過這種方法及時發現和解決問題，確保系統的功能和性能符合預期要求。



我們使用Git Flow的方式來管理專題的開發流程，具體分為主分支（master）、開發分支（develop）、特性分支（feature branches）、發布分支（release branches）和修復分支（hotfix branches），以確保程式碼的版本和發佈的穩定性。

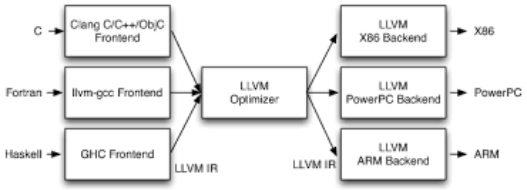


在開發環境上我們使用了相對穩定的 Ubuntu-20.04 版本，藉由 Linux 開源環境的特性更加快捷的進行專案的建設及相關依賴的安裝。

我們選擇 LLVM 和 PEGTL 作為本專題的主要開發工具和框架，除了其豐富的功能外，更因為它們在效率方面的卓越表現。

LLVM 在程式碼生成方面具有優秀的效率。其強大的程式碼優化能

力可以讓我們輕鬆地對中間表示(IR)進行多種優化，包括無用程式碼消除、循環展開等，從而提高生成的目標程式碼的性能和效率。此外，LLVM 的模塊化設計使得它可以輕鬆地與其他工具和框架統合。



PEGTL 是基於 C++ 模板的函式庫，可以在編譯時進行優化，更好地處理遞歸、回溯等問題，提高語法分析的效率。

在測試方面，我們使用 Google Test 為測試的框架。除了單元測試外，我們還定期進行整合測試和性能測試，以確保系統的整體功能和性能符合預期要求。通過這些測試方法和技術的綜合應用更全面地評估系統的品質和性能，及時發現和解決問題。

3-2. 語法設計：

在語法設計上我們參考了各個主流的物件導向語言的設計並思考了物件導向的本質，試圖簡化目前主流的語法設計。

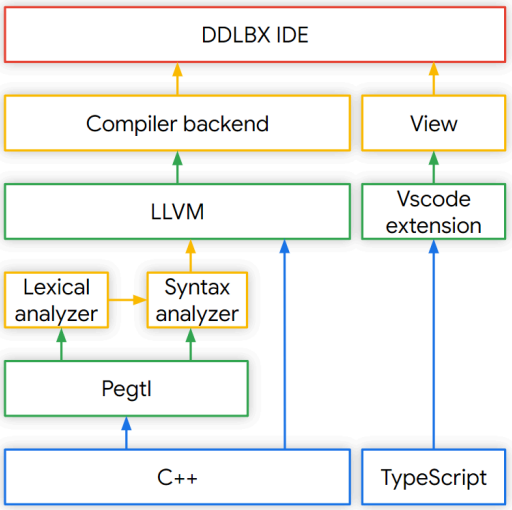
在我們的語言中，我們希望強調物件在資料封裝上的特性，因此我們捨棄了傳統語言中常見的資料與方法共同封裝的語法形式，而用更單純類似於 C 中的 struct 的設計，在方法的封裝上，我們採用了動態綁定的形式進行，也就是將方法從原本的傳統的直接在物件中宣告移除，使用者可以在需要的地方將函式綁定至指定的資料型態上，這個方法讓使用者可以更加靈活的操作物件的函式調用，減少對既有程式碼改動的頻率，增加程式碼的穩定性。

因為在資料封裝上的精簡，相應的繼承這個功能也可以藉由簡單的變成一個成員變數達成，也因為這種設計，讓使用者更加清楚的劃分了架構中各個物件的職責以達到更好的運行邏輯。

以經典的物件繼承例子為例：貓及狗同樣繼承於動物，在動物中風裝了走路及吼叫這兩個方法及相關成員變數，在我們的語言中，我們會更傾向於讓使用者細緻地分割動物這個物件的行為，變成有兩個物件：腳及喉嚨，貓跟狗分別擁有這兩個物件作為成員變數，在這種設計下，可以減少繼承衝突的情形及使使用者可以更加精準的控制繼承關西，減少無用程式碼的產生。

因此我們選擇了 mixin 這種方式代替原本的繼承，提升了在架構設計上的靈活度。

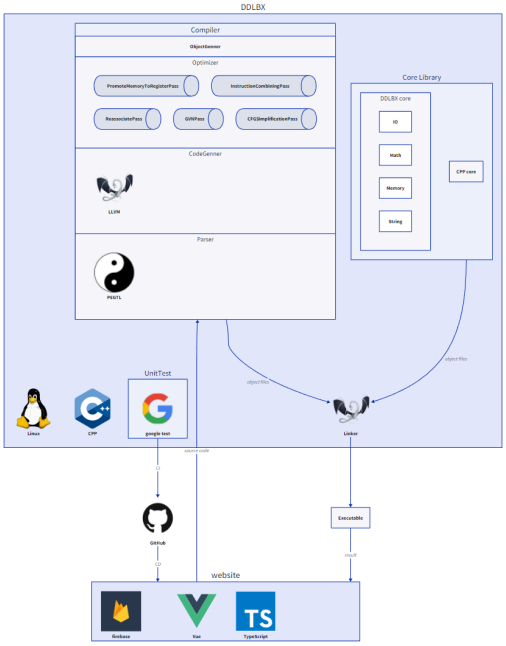
3-3. 編譯器架構



我們的編譯器可以分成核心編譯器及 VSC (Visual Studio Code) 語言插件兩個部分，在編譯器核心部份我們會先使用 PEGTL 對文本內容進行語法分析並生成語法樹，之後使用 LLVM 針對語法樹生成 IR (Intermediate Representation)，在 IR 上會進行有關程式碼優化及 GC (Garbage Collect) 的流程，之後轉入編譯器的後端轉換成 Object code，最後再藉由 Linker 將 Object code 及相關依賴編譯成執行檔。

編譯器中包含了核心函示庫，其中包含了基礎的 IO 及簡單的演算法相關函式，讓使用者可以直接調用，同時提供了使用者自訂義的接口，讓我們的語言可以輕鬆的引入 C++ 的函示庫，增加核心函示庫內容的豐富度。

Visual Studio Code（VSC）是一個流行的程式碼編輯器，具有豐富的擴展性。我們在其架構上開法了語言插件。語法提示使用 TextMate 語法（通常是 .tmLanguage 或 .plist 文件）來描述語言的語法規則。也將完成品上架至 Visual Studio Code 的擴展市場，供使用者下載和使用。



3-4. 展示網站：

在展示網頁上我們使用了 Vue.js 框架並架設在 GCP (Google Cloud Platform) 上，並且引入了 Markdown 語法，Vue.js 是一個 MVC (Model View Controller) 的架構，可以簡單地把系統根據其職責拆分成 Model 、View 及 Controller 三個部分，便於維護和分工開發。View 單純負責呈現使用者介面。Model 負責程式所需的資料與主要的程式邏輯。Controller 負責 View 與 Model 間的溝通。

4. 主要成果與評估

1.編譯器核心實現：基於 LLVM 的模塊化設計，我們開發了一個具有高度擴展性的編譯器核心，並增加了多種優化技術，如無用程式碼消除和循環展開，提升了目標程式碼的性能。

2.物件導向概念的設計研究：我們深入探討並解構了物件導向程式設計（OOP）在語言層面的實現，並研究了如何在編譯器中更有效地支持這些概念。

5. 結語與展望

我們計劃在未來進一步優化編譯器的性能，增加對更多語言特性的支持，並探索新的語法分析技術，以提升編譯器的靈活性和適用性。我們期望這些工作能夠對編譯器技術的發展做出貢獻，並為開發者提供更加高效和強大的工具。

銘謝

回顧所有對於本專題製作過程有具體貢獻之單位或個人,並表答謝忱

6. 參考文獻