

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC** **CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**Logo

Description automatically generated**

**BÁO CÁO CUỐI KỲ MÔN**

**NGUYÊN LÝ VÀ PHƯƠNG PHÁP LẬP TRÌNH**

***Đề tài :*** Nghiên cứu và cài đặt chương trình phân tích cú pháp của ngôn ngữ lập trình Python.

**Giảng viên:** Trịnh Quốc Sơn

**Sinh viên:** Nguyễn Huỳnh Hải Đăng

**Lớp:** CS111.M21.KHCL

**MSSV:** 20521159

**Mục Lục**

[I. Giới thiệu về trình biên dịch 2](#_Toc106375485)

[II. Cấu trúc của bộ phân tích từ vựng 3](#_Toc106375486)

[**1.** **Vai trò của bộ phân tích từ vựng** 3](#_Toc106375487)

[**2.** **Biểu thức chính quy** 4](#_Toc106375488)

[3. **Các bước xây dựng bộ phân tích từ vựng** 5](#_Toc106375489)

[III. Cấu trúc của bộ phân tich cú pháp 5](#_Toc106375490)

[**1.** **Vai trò của bộ phân tích cú pháp** 5](#_Toc106375491)

[**2.** **Những yêu cầu của bộ phân tích cú pháp** 5](#_Toc106375492)

[**3.** **Văn phạm phi ngữ pháp CFG** 6](#_Toc106375493)

[**4.** **Cây phân tích cú pháp** 6](#_Toc106375494)

[5. **Các bước xây dựng bộ phân tích cú pháp** 7](#_Toc106375495)

[IV. Mô tả về chương trình phân tích cú pháp của ngôn ngữ Python 7](#_Toc106375496)

[**1.** **Ý tưởng của chương trình** 7](#_Toc106375497)

[**2.** **Cấu trúc của chương trình** 8](#_Toc106375498)

[**a.** **Giao diện đồ họa** 8](#_Toc106375499)

[**b.** **Chương trình phân tích từ vựng** 8](#_Toc106375500)

[**c.** **Chương trình phân tích cú pháp** 9](#_Toc106375501)

[**3.** **Lưu đồ thuật toán** 9](#_Toc106375502)

[V. Danh mục các tài liệu tham khảo 10](#_Toc106375503)

1. **Giới thiệu về trình biên dịch**

Trình biên dịch, còn được gọi là complier hay một chương trình biên dịch giúp biên dịch một chuỗi các câu lệnh của các ngôn ngữ lập trình cấp cao sang hợp ngữ hoặc ngôn ngữ máy sao cho ý nghĩa của các câu lệnh hay chương trình được biên dịch vẫn được đảm bảo. Trình biên dịch ra đời đã giúp các lập trình viên có thể tạo ra các ngôn ngữ cấp cao thân thiện với con người và diễn đạt các ý tưởng phức tạp hơn, từ đó xây dựng được nhiều các chương trình hay hệ thống có độ phức tạp ngày càng cao. Cấu tạo của trình biên dịch được mô hình hóa như sau:

Bảng ký hiệu

Phân tích từ vựng

Phân tích cú pháp

Phân tích ngữ nghĩa

Sinh mã trung gian

Tối ưu mã trung gian

Sinh mã đích

Bộ quản lý lỗi

Mã nguồn

Chương trình đích

Đầu tiên chương trình được viết bằng các ngôn ngữ lặp trình cấp cao sẽ được trình biên dịch truyền vào bộ phân tích từ vựng để kiểm tra lỗi liên quan đến các từ khóa mặc định của chương trình, đồng thời giai đoạn này cũng giúp xây dựng bảng ký hiệu. Sau khi đi qua bộ phân tích từ vựng mã nguồn lúc đầu sẽ trở thành một chuỗi token, đây là đầu vào cho các giai đoạn sau.

Sau đó, các token sẽ được truyền vào bộ phân tích cú pháp để kiểm tra xem mã nguồn có được tổ chức đúng với cú pháp của ngôn ngữ không bằng cách dựng cây cú pháp, đồng thời cũng giúp xây dựng bảng ký hiệu và nếu phát hiện lỗi thì sẽ báo cho bộ quản lý lỗi. Cây cú pháp sẽ được truyền qua bộ phân tích ngữ nghĩa để kiểm tra xem chương trình có tuân theo các quy tắc của ngôn ngữ lập trình hay không đồng thời cũng bộ phân tích cũng kiểm tra các kiểu dữ liệu của các biến và biểu thức của chúng có thích hợp hay không, hay các biến có được khai báo trước khi gán giá trị hay biến đã có giá trị hay chưa trước khi được thực hiện các biểu thức. Đầu ra của bộ phân tích ngữ nghĩa là một cây cú pháp nhưng có các chú thích.

Sau khi mã nguồn đã được kiểm tra thì mã nguồn sẽ được qua bộ tạo mã trung gian để sinh mã trung gian, đây là mã nằm giữa ngôn ngữ cấp cao và mã máy nhầm giúp việc biên dịch về mã máy dễ dàng hơn, tuy nhiên chương trình được chuyển về mã trung gian vẫn phải bảo toàn được ý nghĩa của chương trình gốc. Sau khi mã trung gian được sinh ra thì sẽ được tối ưu bằng bộ tối ưu, bộ tối ưu sẽ loại bỏ tất cả các đoạn mã không cần thiết nhầm giúp tăng tốc cho quá trình biên dịch mã máy. Trong giai đoạn cuối, mã trung gian sau khi được tối ưu sẽ được ánh xạ sang mã máy, là mã được biểu diễn dạng nhị phân để máy có thể thực hiện.

Bộ phân tích cấu pháp và bộ phân tích từ vựng được tách ra thành hai giai đoạn khác nhau nhầm giúp đơn giản hóa chương trình phân tích từ vựng và chương trình phân tích cú pháp, điều này giúp tăng tính hiệu quả của trình biên dịch. Bên cạnh đó việc tách hai bộ phân tích giúp cho việc bảo trì và nâng cấp dễ dàng hơn đồng thời cũng tăng tính linh động cho cả hai bộ phân tích. Ví dụ đối với bộ phân tích cú pháp thì chỉ cần khai báo tập luật sinh và chuỗi các token đầu vào.

1. **Cấu trúc của bộ phân tích từ vựng**
   1. **Vai trò của bộ phân tích từ vựng**

Có thể thấy rằng, bộ phân tích từ vựng là giai đoạn đầu tiên của trình biên dịch và đầu ra của bộ phân tích từ vựng là đầu vào của bộ phân tích cú pháp. Do đó bộ phân tích từ vựng có vai trò quan trọng và cần phải xây dựng bộ phân tích từ vựng trước khi xây dựng bộ phân tích cú pháp.

Bộ phân tích từ vựng có đầu vào một mã nguồn được viết bằng ngôn ngữ cấp cao dưới dạng một chuỗi. Bộ phân tích từ vựng sẽ dùng các biểu thức chính quy để trích xuất các từ khóa và ký tự được biệt như dấu “:”,”;” và nhiều dấu khác. Bên cạch đó bộ phân tích từ vựng còn tách tên các biến, các giá trị như số hoặc chuỗi và tạo các token cho các thành phần đó.

Token thường có cấu trúc dữ là một danh sách trong đó chứa thông tin là ký hiệu kết thúc, giá trị - thường là tên biến, 1 chuỗi hay 1 con số bao gồm các số nguyên và các số thập phân, đôi khi token còn chứa các thông tin khác.

Ví dụ về các token: <id,”a”>,<str,”abbd”>,<num,”10.3”>. Trong đó, id,str và num là các ký hiệu kết thúc và “a”,”abbd” và “10.3” là giá trị của token đó. Giá trị này thường được lưu trong bảng ký hiệu của trình biên dịch.

* 1. **Biểu thức chính quy**

Biểu thức chính quy là một chuỗi các ký tự dùng để tạo ra khuôn mẫu cho các chuỗi nhằm mục đích tìm kiếm và kiểm tra dễ dàng hơn. Trong xây dựng bộ phân tích từ vựng, biểu thức chính quy giúp phân tách các ký tự đặc biệt, các từ khóa, chứ số, một chuỗi, tên của các biến

Các ký hiệu trong biểu thức chính quy:

* Ký hiệu “|”
  + Các lựa chọn thay thế
  + Ví dụ: alpha|beta có thể khớp với chuỗi “alpha" hoặc “beta".
* Ký hiệu phân nhóm “()”
  + Được sử dụng để xác định phạm vi và mức độ ưu tiên của các toán tử
  + Ví dụ: bin|pin và (b|p)in là các mẫu tương đương, cả hai đều mô tả tập hợp “bin" or “pin".
* Ký hiệu đại diện “.”
  + Được dùng để đại diện các cho một ký tự bất kỳ nhưng không phải là ký tự “\n”
  + Ví dụ: d.b khớp với bất kỳ xâu nào bắt đầu là “d", sau đó là ký tự bất kỳ khác “\n” và kết thúc là "b”
* Ký hiệu định lượng
  + Chỉ định tần suất mà phần tử trước, có thể là ký tự hoặc nhóm, được phép xảy ra.
  + “?”: Cho biết phần tử trước hoặc là không xuất hiện hoặc là xuất hiện 1 lần.
    - Ví dụ: colou?r khớp với cả "color" and "colour".
  + “\*”: Cho biết phần tử trước có thể không có hoặc có thể xuất hiện nhiều lần.
    - Ví dụ: de \* c khớp với “de",”dec”, “deec”
  + “+”: Cho biết phần tử trước có thể xuất hiện một hoặc nhiều lần.
    - Ví dụ: ab + c khớp với "abc", "abbc", "abbbc", v.v., nhưng không phải là "ac".
  + {n} Mục trước được khớp chính xác n lần.
  + {min,} Mục trước được khớp tối thiểu hoặc nhiều lần hơn.
  + {min,max} Mục trước được khớp ít nhất lần tối thiểu, nhưng không quá lần tối đa.
  1. **Các bước xây dựng bộ phân tích từ vựng**

Đầu tiên, cần xây dựng tập các từ khóa của ngôn ngữ và các ký hiệu đặc biệt như các ký hiệu toán học, các ký hiệu logic như >,<,<=. Sau đó, xây dựng các biểu thức chính quy như đã trình bày ở trên để trích xuất được câc thông tin cần thiết như từ khóa, các ký tự đặc biệt, tên biến, tên hàm, số hay một chuỗi ký tự.

Trong phạm vi bài tập, tác giả dùng ngôn ngữ lập trình python để cài đặc nên dùng thư viện re trong python để cài đặt các biểu thức chính quy. Trong file cài đặt – lexer\_analysis.py – tập các từ khóa và các ký tự đặt được được tổ chức theo cấu trúc dữ liệu “dict” trong python.

Các ví dụ về biểu thức chính quy được:

* Biểu thức [.0-9] dùng để trích xuất ra các giá trị là số
* Biểu thức [\_a-zA-Z09] dùng để trích xuất các giá trị là chuỗi - tên biến.

Ví dụ về cách bộ phân tích từ vựng:

Mã nguồn đầu vào: print(a+3)

Các token được sinh ra: <print> <(> <id,”a”> <+> <num,”3”> <)>

1. **Cấu trúc của bộ phân tich cú pháp**
2. **Vai trò của bộ phân tích cú pháp**

Bộ phân tích cú pháp cung cấp đầu vào cho bộ phân tích ngữ nghĩa là một cây cú pháp, đồng thời cũng góp phần xây dựng bảng ký hiệu và pháp hiện các lỗi liên quan đến cú pháp và các lỗi về từ khóa. Đầu vào của bộ phân tích cú pháp là một chuỗi các token do bộ phân tích từ vựng tạo ra và đầu ra là một cây cú pháp.

Để xây dựng bộ phân tích cú pháp, có hai cách tiếp cận là top-down và bottom-up. Đối với cách tiếp cận top-down, thì chương trình sẽ xây dựng cây cú pháp từ ký tự bắt đầu và dùng các luật sinh để chuyển đổi ký hiệu bắt đầu thành các ký hiệu đầu vào – token. Đối với bottom-up, thì ngước lại, chương trình sẽ dựa vào các token đầu vào và cố gắng áp dụng các luật sinh để đưa về ký hiệu bắt đầu.

1. **Những yêu cầu của bộ phân tích cú pháp**

Bộ phân tích cú pháp phải được thiết kế để bảo chương trình phân tích phải quan lý bộ nhớ hiệu quả, đồng thời phải có độ chính xác cao và tốc độ xử lý của chương trình phải nhanh để giảm chi phí thời gian cho trình biên dịch. Bên cạnh đó, bộ phân tích cú pháp cần chịu được lỗi bằng cách có các chiến lược để giải quyết các lỗi xuất hiện.

1. **Văn phạm phi ngữ pháp CFG**

Văn phạm phi ngữ pháp là một văn phạm hình thức mà trong đố các luật sinh ở dạng A 🡪 B. Bên cạnh đó CFG được cấu tạo từ bốn phần:

* Tập các ký hiệu kết thúc: là tập các ký tự được sinh ra các luật sinh hay các ký tự nằm bên vế phải của luật sinh
* Tập các ký hiệu không kết thúc: là các tập các ký tự thay thế hay cac ký tự nằm ở vế trái của luật sinh.
* Tập các luật sinh: là tập các quy tắc thay thế các ký hiệu không kết thúc thành các ký tự kết thúc hoặc các ký tự không kết thúc.
* Ký hiệu bắt đầu: đây là một ký tự đặc biệt nhằm đánh dấu sự bắt đầu.

Ví dụ:

Luật sinh:

<para> 🡪 <id>|<num>

<factor> 🡪 <para>|<factor>\*<factor>|<factor>/<factor>

<start> -> (<factor>) | <factor>

Tập ký hiệu kết thúc: <id>, <num>

Tập ký hiệu không kết thúc: <factor>, <para>

Ký hiệu bắt đầu: <start>

1. **Cây phân tích cú pháp**

Cây phân tích cú pháp là một cây có gốc là ký tự bắt đầu, có trật tự, thể hiện cấu trúc cú pháp của một chuỗi theo tập các luật sinh hay một số ngữ pháp phi ngữ cảnh. Bên cạnh đó, một cây cú pháp là một cây thể hiện cấu trúc cú pháp trừu tượng của một chương trình được vi.ết bằng ngôn ngữ lập trình cấp cao. Cấu tạo của cây phân tích gồm có:

* Nút gốc: Là ký hiệu bắt đầu
* Nút lá: Là các token đầu vào
* Nút trung gian: Là các ký hiệu không kết thúc

Ví dụ:

Luật sinh:

<para> 🡪 <id>|<num>

<factor> 🡪 <para>|<factor>\*<factor>|<factor>/<factor>

<start> -> (<factor>) | <factor>

Chuỗi đầu vào: <id,’a’><\*><id,’5.3’>

Cây cú pháp:

start

factor

factor

factor

\*

num

id

“a”

“10.3”

1. **Các bước xây dựng bộ phân tích cú pháp**

Đầu tiên, cần xây dựng tập các luật sinh, đây là bộ quy tắc hay quy định của ngôn ngữ. Nói cách khác, tập luật sinh này đại diện cho cú pháp của ngôn ngữ lập trình. Tập luật sinh phải có tập các ký hiệu không kết thúc, tập các ký hiệu kết thúc và ký hiệu bắt đầu.

Lựa chọn hướng tiếp cận phù hợp với ngôn ngữ lập trình như hướng tiếp cận top-down hoặc bottom-up. Đồng thời, trong quá trình xây dựng bộ phân tích cú pháp, cần xây dựng các chiến lược xử lý lỗi

1. **Mô tả về chương trình phân tích cú pháp của ngôn ngữ Python**
2. **Ý tưởng của chương trình**

Trong khuôn khổ bài tập cuối kỳ, do đó tập luật sinh chỉ mô phỏng luật sinh của ngôn ngữ lập trình Python ở các điểm là các so sánh, phép gán, các phép toán, các câu lệnh đến nhập xuất và các câu lệnh vòng lặp, rẻ nhánh.

Bộ phân tích cú pháp được cài đặt theo hướng bottom-up bằng thuật toán Shift-Reduce.

Từ chuỗi token đầu vào, bộ phân tích duyệt từ trái qua phải chuỗi token đầu vào, cố gắng áp dụng các luật sinh trong văn phạm để xây dựng một cây cú pháp.

Thuật toán cài đặt:

* + Bước 1:
    - * Yêu cầu token đầu tiên từ chuỗi token đầu vào
    - Bước 2:
      * Chọn luật sinh đầu tiên có vế phải giống với token.
      * Nếu tồn tại luật sinh thỏa mãn vế phải giống với token thì lấy đó làm nút hoạt động và lập lại bước 2
      * Nếu không thì sang bước 3
    - Bước 3
      * Nếu như đã lấy hết token từ chuỗi token đầu vào
        + Nếu nút hoạt động hiện tại không là nút gốc thì trả về cây phân tích đồng thời báo lỗi và trả lại kết quả không chấp nhận
        + Nếu nút hoạt động hiện tại là nút gốc thì trả về cây phân tích và trả về kết quả chấp nhận
      * Nếu không thì lập lại bước 1

Chương trình cài đặt bộ phân tích cú pháp được viết bằng ngon ngữ python và dùng các thư viện hỗ trợ gồm:

PyQt5: thư viện hỗ trợ hiển thị đồ họa

nltk: thư viện hỗ trợ thuật toán Shift-Reduce

pandas: thư viện hỗ trợ việc sử lý dữ liệu và thành lập bảng lưu lịch sử khi thuật toán shift-reduce thực thi.

1. **Cấu trúc của chương trình**
   1. **Giao diện đồ họa**

Chương trình sử dựng thư viện PyQt5 để hiển thị đồ hoạ, giao diện gồm 2 khung là một cửa số để nhập mã nguồn và in ra kết quả kiểm tra, cửa sổ còn lại hiển thị bảng lịch sử thực thi của thuật toán shift-reduce. Mã nguồn nhập vào được lưu trong file inpit.py. File graph\_interface.py chứa mã nguồn đồ hoạ.

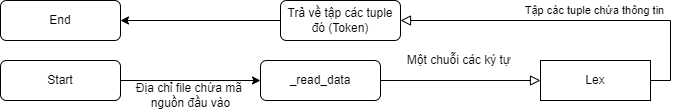
* 1. **Chương trình phân tích từ vựng**

Mã nguồn của chương trình nằm trong file lexer\_analysis.py. Bộ phân tích cú pháp được cài đặt trong hàm lex.

Còn hàm read\_data có chức năng đọc dữ liệu đầu vào ở file input.py, hàm các hàm có tên bắt đầu bằng scan có chức năng trích xuất các chuỗi ký tự hoặc số.

Lớp DATA trong file là nơi lưu trữ chuỗi dữ liệu đầu vào sau khi được load bởi hàm read\_data

Lưu đồ thuật toán:

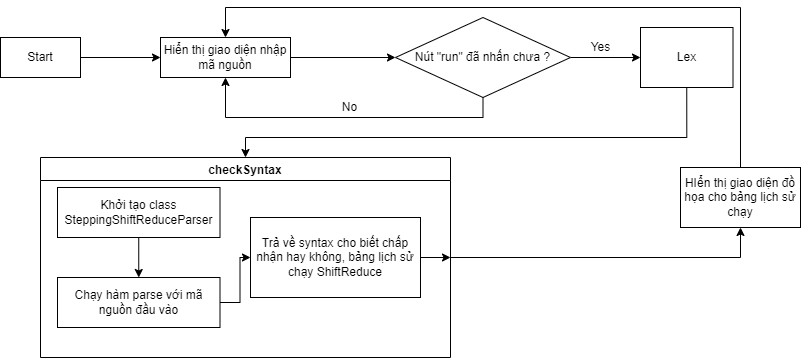


* 1. **Chương trình phân tích cú pháp**

Mã nguồn của chương trình được lưu trong file Parser.py. Chương trình dùng lớp CFG để mô tả tập các luật sinh và lớp SteppingShiftReduceParser để cài đặt thuật toán Shift-Reduce.

Dùng hàm parse trong lớp SteppingShiftReduceParser để xây dựng cây cú pháp đồng thời tạo bảng

1. **Lưu đồ thuật toán**



1. **Danh mục các tài liệu tham khảo**
2. Anon. 2020. “Các giai đoạn của trình biên dịch.” *Comdy*. Retrieved April 30, 2022 (<https://comdy.vn/trinh-bien-dich/cac-giai-doan-cua-trinh-bien-dich/>).
3. Anon. 2021a. “Thiết kế trình biên dịch: Phân tích cụ pháp.” *Comdy*. Retrieved April 30, 2022 (<https://comdy.vn/trinh-bien-dich/phan-tich-cu-phap/>).
4. Anon. 2021b. “Thiết kế trình biên dịch: Phân tích ngữ nghĩa.” *Comdy*. Retrieved April 30, 2022 (<https://comdy.vn/trinh-bien-dich/phan-tich-ngu-nghia/>).
5. Anon. 2021c. “Thiết kế trình biên dịch: Tổng quan.” *Comdy*. Retrieved April 30, 2022 (<https://comdy.vn/trinh-bien-dich/tong-quan-thiet-ke-trinh-bien-dich/>).
6. Anon. 2022. “Lexical Analysis.” *Wikipedia*.
7. Anon. n.d.-a. “8. Analyzing Sentence Structure.” Retrieved May 4, 2022 (<https://www.nltk.org/book/ch08.html>).
8. Anon. n.d.-b. “10. Full Grammar Specification — Python 3.10.4 Documentation.” Retrieved May 4, 2022 (<https://docs.python.org/3/reference/grammar.html>).
9. Anon. n.d.-c. “A Guide to Parsing: Algorithms and Technology (Part 8) - DZone AI.” *Dzone.Com*. Retrieved April 30, 2022 (<https://dzone.com/articles/a-guide-to-parsing-algorithms-and-technology-part-2>).
10. Anon. n.d.-d. “Bài 7 : Lập Trình Giao Diện Với PyQt5 Cho RaspberryPi - Phần 1.” Retrieved April 30, 2022 (<https://mlab.vn/index.php?_route_=17161-bai-7-lap-trinh-giao-dien-voi-pyqt5-cho-raspberrypi-phan-1.html>).
11. Anon. n.d.-e. “Base of All Parser — Pyrser 0.0.10 Documentation.” Retrieved May 4, 2022 (<https://pythonhosted.org/pyrser/base.html#grammar-composition>).
12. Anon. n.d.-f. “ISBN Chapter 4 Lexical and Syntax Analysis. - Ppt Download.” Retrieved May 3, 2022 (<https://slideplayer.com/slide/9197120/>).
13. Anon. n.d.-g. “Khóa: Nguyên Lý và Phương Pháp Lập Trình - CS111.M21.KHCL.” Retrieved April 30, 2022 (<https://courses.uit.edu.vn/course/view.php?id=8829>).
14. Anon. n.d.-h. “Lexical and Syntactic Analysis Here, We Look at Two of the Tasks Involved in the Compilation Process –Given Source Code, We Need to First Break It into. - Ppt Download.” Retrieved May 3, 2022 (<https://slideplayer.com/slide/4752921/>).
15. Anon. n.d.-i. “NLTK :: Nltk.Parse.Shiftreduce Module.” Retrieved May 5, 2022 (<https://www.nltk.org/api/nltk.parse.shiftreduce.html>).
16. Anon. n.d.-j. “PEP 617 – New PEG Parser for CPython | Peps.Python.Org.” Retrieved May 4, 2022 (<https://peps.python.org/pep-0617/#s-e>).
17. Anon. n.d.-k. “PyQt - Hello World.” Retrieved May 5, 2022 (<https://www.tutorialspoint.com/pyqt/pyqt_hello_world.htm>).
18. Anon. n.d.-l. “Writing a BNF Grammar — Pyrser 0.0.10 Documentation.” Retrieved May 4, 2022 (<https://pythonhosted.org/pyrser/dsl.html>).
19. Computer Science. 2018. *Compilation - Part Two: Lexical Analysis*.
20. Helpex. 2017. “Phân tích cú pháp trong Python Công cụ và Thư viện Phần 1.” *helpex.vn*. Retrieved April 30, 2022 (<https://helpex.vn/article/phan-tich-cu-phap-trong-python-cong-cu-va-thu-vien-phan-1-5c6b1a5aae03f628d053bdc3>).
21. Huynh, Danh. [2021] 2022. *LexicalAndParser*.
22. temple. 2014. “Lexical and Syntax Analysis Chapter 4.” *SlideServe*. Retrieved May 3, 2022 (<https://www.slideserve.com/temple/lexical-and-syntax-analysis-chapter-4>).
23. Yatish Parmar. 2016. *Lexical and Syntax Analysis - A Level Computer Science*.