ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP LỚN GIỮA KÌ

MÔN HỌC: CHƯƠNG TRÌNH DỊCH

ĐỀ BÀI: XÂY DỰNG BỘ PHÂN TÍCH TỪ VỰNG CHO NGÔN NGỮ VC

Giảng viên: Nguyễn Văn Vinh

Nhóm 14:

Sinh viên: Đỗ Thu Uyên
Lê Thị Mơ

Lớp: K63-K2

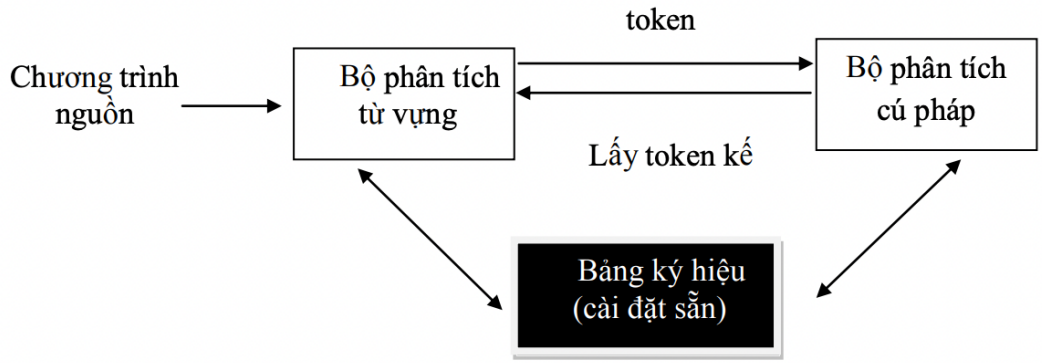
Hà Nội, tháng 4 năm 2022

MỤC LỤC

1. Giới thiệu chung về bộ phân tích từ vựng

Đây là giai đoạn đầu tiên của quá trình biên dịch. Bộ phân tích từ vựng có nhiệm vụ là đọc các kí tự nhập vào từ chương trình nguồn và phân tích đưa ra danh sách các từ tố (từ vựng và phân loại cú pháp của nó) cùng một số thông tin thuộc tính, lưu vào một bảng tạm gọi là bảng chuyển trạng thái.

Đầu ra của bộ phân tích từ vựng là danh sách các từ tố đã lưu vào một bảng tạm gọi là bảng chuyển trạng thái và là đầu vào cho phân tích cú pháp. Phần này có thể coi là tiền xử lý của văn bản chương trình nguồn, làm cho nhiệm vụ của các giai đoạn sau đơn giản hơn.



Hình 1.1: Vị trí khối phân tích từ vựng

2. Giới thiệu chung về ngôn ngữ VC

VC là một biến thể của ngôn ngữ C. Nó có cả những đặc điểm chính của ngôn ngữ C và một vài đặc điểm của ngôn ngữ Java. VC bao gồm một số kiểu dữ liệu cơ sở, kiểu mảng một chiều, các cấu trúc điều khiển, các biểu thức, các câu lệnh phức hợp (ví dụ như các khối) và các hàm. Đó là những đặc tính phần lớn được "lấy" từ ngôn ngữ C.

Bài toán "Xây dựng bộ phân tích từ vựng ngôn ngữ VC" được thực hiện với mục đích sử dụng bộ phân tích từ vựng để nhận diện chính xác các từ vựng trong một file mã nguồn VC dựa vào bảng chuyển trạng thái có sẵn được tích hợp và đồng thời báo lỗi nếu có.

3. Hướng giải quyết bài toán

3.1 Các bước thực hiện

Để xây dựng được một bộ phân tích từ vựng đạt đúng yêu cầu, cần thực hiện theo các bước sau:

- Xây dựng danh sách tất cả mẫu (luật từ vựng), những mẫu này thường được mô tả bằng ngôn ngữ bình thường.
- Vẽ đồ thị chuyển tiếp cho từng mẫu (luật) một.
- Kết hợp tất cả các đồ thị chuyển thành một đồ thị duy nhất.
- Tạo bảng chuyển trạng thái từ đồ thị chuyển và xuất sang định dạng thích hợp.
- Xây dựng chương trình của bạn để đọc bảng được định dạng.
- Thêm cơ chế phát hiện cho lỗi biên dịch.

a. Xây dựng danh sách luật từ vựng

Loại từ tố	Luật từ vựng
Định danh (id)	Một chuỗi các ký tự, chữ số và dấu gạch dưới không giới hạn độ dài, phải bắt đầu bằng một kí tự hoặc dấu gạch dưới.
Từ khóa	Các chuỗi ký tự dưới đây là các từ khóa và không thể sử dụng chúng như là các định danh: *boolean, break, continue, else, for, float, if, int, return, void, while*.

Hằng (literal)	Một giá trị hằng là một biểu diễn gốc (source) của một giá trị kiểu nguyên (int), kiểu thực (float), kiểu logic (boolean) hoặc kiểu xâu ký tự (string).
Kí tự phân tách	Sáu ký tự ASCII sau được qui ước là các dấu ngăn cách trong VC: { } ( ) [ ] ; ,
Toán tử	Toán tử toán học: +-*/ Toán tử liên hệ: < <= > >= = Toán tử so sánh bằng: == != Toán tử logic: || && ! Toán tử gán: =
Escape	Tập hợp đầy đủ của các chuỗi escape được hỗ trợ là: \b \f \n \r \t
Chú thích	Chú thích truyền thống: /* */ Chú thích hết dòng: //

b. Xây dựng bảng chuyển

Bảng chuyển được xây dựng từ đồ thị chuyển. Ngoài ra chúng em còn xuất ra một file .dat đính kèm. Chi tiết như sau:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
state	character	digit	.	<	=	>	e/E	+	-	*	/	|	&	!	"	#
0	1	3	x	16	20	23	1	26	28	30	32	35	38	41	12	x
1	1	1	2	2	2	2	1	2	2	2	2	2	2	2	2	2
3	11	3	4	11	11	11	7	11	11	11	11	11	11	11	11	11
4	x	5	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5	6	5	6	6	6	6	7	6	6	6	6	6	6	6	6	6
7	x	9	x	x	x	x	x	8	8	x	x	x	x	x	x	x
8	x	9	x	x	x	x	x	x	x	x	x		x	x	x	x
9	10	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10
12	13	13	x	x	x	x	x	x	x	x	x	x	x	x	x	x
13	13	13	x	x	x	x	x	x	x	x	x	x	x	x	14	x
14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
16	19	19	19	19	17	18	19	19	19	19	19	19	19	19	19	19
20	21	21	21	21	22	21	21	21	21	21	21	21	21	21	21	21
23	25	25	25	25	24	25	25	25	25	25	25	25	25	25	25	25
26	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27
28	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
30	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31
32	33	33	33	33	33	33	33	33	33	33	34	33	33	33	33	33
35	x	x	x	x	x	x	x	x	x	x	x	36	x	x	x	x
36	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37
38	x	x	x	x	x	x	x	x	x	x	x	x	39	x	x	x
39	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
41	42	42	42	42	43	42	42	42	42	42	42	42	42	42	42	42

Hình 3.1: Bảng chuyển trạng thái

c. Xây dựng cơ chế phát hiện lỗi biên dịch

Chương trình sẽ báo lỗi khi gặp một từ mà không phù hợp với bất kì một luật nào quy định trong file dữ liệu automat. Sau khi báo lỗi, chương trình sẽ bỏ qua tất cả các kí tự theo sau vị trí lỗi cho tới khi nó hết dòng. Việc phát hiện lỗi này có thể không chính xác trong mọi trường hợp, nhưng có thể chấp nhận được.

Từ vựng bị lỗi sẽ được in ra với kiểu là “compiler error at line **number**, index **number**”.

4. Input và Output của chương trình

Input: tệp văn bản *.vc chứa mã VC.

Output: tệp văn bản (* .vctok) chứa danh sách các từ vựng và kiểu của nó, mỗi từ vựng nằm trên một dòng theo định dạng:

<từ vựng><dấu cách><kiểu từ vựng>

Dữ liệu Automat (* .dat) của chúng bao gồm từng mục sau được sắp xếp theo thứ tự:

- Kích thước bảng chuyển automat
- Bảng chuyển trạng thái (tệp văn bản được định dạng là ASCII)
- Liệt kê các loại token
- Trạng thái kết thúc (ending*)
- Ánh xạ trạng thái kết thúc (*) và loại từ tố
- Trạng thái kết thúc (ending)
- Ánh xạ trạng thái kết thúc và loại từ tố

5. Cài đặt

5.1 Ngôn ngữ sử dụng

Chương trình được cài đặt bằng ngôn ngữ Python, phiên bản 3.7.

5.2 Cấu trúc chương trình

Cấu trúc chương trình gồm hai lớp:

- Class CharacterType: Xác định loại cạnh.
  - Danh sách các thuộc tính:
    - *n:* số lượng các kí tự.
    - *character2type*: lưu lại loại từng kí tự.
  - Hàm khởi tạo __init__ với tham chiếu đầu vào là một list các dòng, sau đó ta đánh số cho từng dòng một. Thực hiện tạo một dictionary character2type, phân tách từng kí tự trên một dòng thành một key, value là số thứ tự của chính dòng chứa kí tự.
  - Hàm get_edge với đầu vào là một kí tự c, kí tự c có thể đã được chứa trong dictionary, hàm có mục đích kiểm tra kí tự c đó có thuộc SEPARATOR, BRACKET hay không, nếu không thì sẽ trả về value của key c trong dictionary.
- Class DFA: Cài đặt lớp DFA
  - Danh sách các thuộc tính:
    - *graph:* lưu lại các state và các đỉnh.
    - *parser_character:* khởi tạo loại character.

- *ending_asterisk*: lưu lại ending* state và các tên kiểu token tương ứng.
- *ending*: lưu lại ending* và các tên kiểu token tương ứng.

○ Hàm khởi tạo __init__ :

- Đầu vào là một file automat mà chúng em đã xây dựng, đọc từng dòng trong file. Dựa theo format của file automat, khi đọc qua dòng đầu tiên sẽ đọc được kích thước bảng chuyển, bảng có kích thước là NxM. Sau đó ta đọc lần lượt N dòng tiếp theo để đọc dữ liệu bảng chuyển, thực hiện tạo một dictionary graph lưu lại các state (key) có value là các đỉnh, nếu state không có đỉnh thì lỗi, trả về -1. Bảng chuyển chỉ liệt kê các đỉnh có cạnh đi ra.
- M dòng tiếp theo trong file automat là các ghi định nghĩa các loại token. Ví dụ với token loại digit thì sẽ được định nghĩa là "0123456789". Chúng em xử lí bằng cách thêm từng dòng vào một list list_character_types. M dòng này sẽ được khởi tạo class CharacterType.
- Đọc vào hai dòng tiếp theo của file automat, dòng thứ nhất lưu lại các ending* state và các đỉnh của nó, dòng thứ hai lưu lại name* lần lượt là các tên kiểu token ứng với ending* state. Đưa các giá trị ending* state vào một list và name* vào một list và từ đó tạo ra một dictionary ending_asterisk = {"key = ending* state" : "value = name*"}. Làm tương tự với hai dòng tiếp theo là ending state và name.

○ Hàm run_forward: Hàm nhận 3 tham số là từ đang đọc ở hiện tại, trạng thái hiện tại và kí tự hiện tại. Kết quả trả về của hàm là một bộ có 4 tham số gồm có: cờ nhận giá trị đúng hoặc sai, từ được đọc, trạng thái tiếp theo và thông tin. Đầu tiên sử dụng hàm chuyển đổi về loại cạnh để chuyển ch về loại cạnh đồng thời lấy ra danh sách các cạnh xuất phát từ trạng thái hiện tại. Sau đấy sẽ đi theo cạnh được lưu trong bảng chuyển. Nếu gặp các đỉnh kết thúc thì trả về kết quả từ đang đọc và loại từ tương ứng trong tham số trả về info. Sau đó sẽ xuất phát tiếp từ đỉnh 0. Trong trường hợp đi theo cạnh không tồn tại trong bảng chuyển thì sẽ trả về lỗi.

- Hàm main: thực hiện một số nhiệm vụ chính:

- Đọc thông tin về DFA theo tham số truyền vào.
- Đọc và xử lý từng dòng một trong file mã nguồn đầu vào và xử lý theo từng kí tự của dòng đọc được.
- Output kết quả phân tích từ vựng ra file tương ứng. Lỗi thì sẽ được output ra màn hình.
- Ví dụ về dòng lệnh chạy chương trình:

python lexical.py --input_file test.vc --input_dfa dfa.dat --output_file test.vctok

6. Một số ví dụ

Input file: test.vc	Output file: test.vctok	Standard output

`int main() {`
`  int a = 0;`
`  float b = 1e+3;`
 	
`  if (a > b) a = 1;`
`  else b = 1;`
 	
`  return 1;`
`}`
	`int int_keyword`
	`main identifier`
	`a identifier`
	`= assign_operator`
	`0 int_const`
	`float float_keyword`
	`b identifier`
	`1e+3 float_const`
	`if if_keyword`
	`> GT`
	`1 int_const`
	`else else_keyword`
	`return return_keyword`	

Hình 6.1 Ví dụ về chương trình VC bình thường

Input file: test.vc	Output file: test.vctok	Standard output

`int main() {`
`  int a. = 0;`
 	
`  return 1;`
`}`
	`int int_keyword`
	`main identifier`
	`return`
	`return_keyword`
	`1 int_const`
		`Compiler error at`
		`line 2, index 10`

Hình 6.2 Ví dụ về chương trình VC với lỗi và kết quả phát hiện lỗi trên standard output

7. Phân chia công việc

STT	Họ và Tên	Nhiệm Vụ	Đóng góp
1	Đỗ Thu Uyên	Xây dựng bảng chuyển, phân tích các luật từ vựng, code, fix bug, viết báo cáo.	65%
2	Lê Thị Mơ	Kiểm thử, báo lỗi.	35%

Ý kiến đánh giá:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Điểm số: ……. Điểm chữ: ……

Hà Nội, ngày tháng năm 20 .
Giảng viên đánh giá
(Ký, ghi rõ họ tên)