VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/211867931379013

Nguyên Lí Ngôn Ngữ Lập Trình (PPL)

PPL2 - HK242

Cuối Kì

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/211867931379013

TP. HỒ CHÍ MINH, THÁNG 6/2024

Mục lục

1 Lý thuyết Lexical structure

Tokens là khối xây dựng cơ bản của chương trình, chuỗi ký tự ngắn nhất có ý nghĩa riêng, Chúng là các thành phần cơ bản hay nguyên tố để xây dựng lên chương trình
Dễ hiểu nhất trong quá trình sản xuất bánh mì thì các nguyên vật liệu như **bột, hương vị, nước ...** là các **Tokens** mỗi thứ đều có ý nghĩa riêng của nó.

- **Keywords** là các *Tokens* đã mặc định trước.
- **Operators** là các toán tử cho phép hiện thực trong chương trình
- **Separators** các kí hiệu phân cách của chúng
- **Identifiers** tên của biến, thuộc tính, hàm, class loại này là vô số *Tokens* được tạo ra
- **Literal** là chuỗi trong ngôn ngữ để thể hiện giá trị số nguyên, một chuỗi
- **Lexical errors** gồm 3 loại
  - **Unclosed string** không có kí tự để đóng chuỗi lại khi lập trình thì các phần sau hiện xanh lá trong *vscode*
  - **Illegal escape** in string một kí tự không cho phép trong ngôn ngữ
  - **Error token** không có *token* nào thõa mãn

Chú ý

- *Tokens* luôn bắt được chuỗi dài nhất có thể
- *Tokens* chuỗi bằng nhau thì ưu tiên chuỗi đầu tiên gặp được
- Nên viết tất cả các kí tự hoa (nếu không thì bắt buộc kí tự đầu viết hoa ràng buộc của ngôn ngữ rồi)

Danh sách các kí hiệu và cú pháp thường dùng trong `ANTLR`

1. `'kí tự'`
   - **Ý nghĩa:** Bất kỳ đoạn văn bản nào được đặt trong dấu nháy đơn (`'...'`) sẽ được khớp đúng theo nội dung bên trong, bao gồm cả chữ, số hay kí tự đặc biệt.
   - **Ví dụ:** `'a'` khớp đúng với kí tự `a`. Hoặc `'xyz'` khớp đúng với `xyz`.
2. `A B`
   - **Ý nghĩa:** Yêu cầu `A` xuất hiện trước, rồi ngay sau đó là `B`.
   - **Ví dụ:** `'a' 'b'` khớp với chuỗi `"ab"`. Nếu `A` = `'a'` và `B` = `'b'`, bạn phải thấy `ab` liền kề.
3. `A | B`
   - **Ý nghĩa:** Cho phép chọn một trong hai dạng `A` hoặc `B`.
   - **Ví dụ:** `'a' | 'b'` khớp với `a` hoặc `b`. Tức là chuỗi `a` hoặc `b` đều được chấp nhận.
4. `'text'`
   - **Ý nghĩa:** Tương tự `'kí tự'`, nhưng biểu diễn một chuỗi nhiều kí tự liên tiếp (chuỗi tĩnh).
   - **Ví dụ:** `'Hello'` khớp đúng với `Hello`. Chuỗi này không chấp nhận thiếu hay thừa kí tự nào.
5. `A?`
   - **Ý nghĩa:** `A` có thể xuất hiện hoặc không (tương đương `A | rỗng`).
   - **Ví dụ:** `'a'?` khớp với '' (rỗng) hoặc `'a'`. Tức là có `a` cũng được, mà không có cũng không sao.
6. `A*`
   - **Ý nghĩa:** `A` có thể lặp lại 0 hoặc nhiều lần ($\{\epsilon, A, AA, AAA, \dots\}$).

- **Ví dụ:** `'a'*` có thể khớp `''` (chuỗi rỗng), `'a'`, `'aa'`, `'aaa'`, ...

7. `A+`
   - **Ý nghĩa:** `A` lặp lại 1 hoặc nhiều lần ($\{A, AA, AAA, \dots\}$).
   - **Ví dụ:** `'a'+` khớp với `'a'`, `'aa'`, `'aaa'`, ... (Không được rỗng, phải có ít nhất một `a`.)
8. `[a-z]`
   - **Ý nghĩa:** Chọn một kí tự trong khoảng `a` đến `z` (chữ thường).
   - **Ví dụ:** `[a-z]` khớp với `'a'`, `'b'`, ..., `'z'`.
9. `[A-C]`
   - **Ý nghĩa:** Chọn một kí tự trong khoảng `A` đến `C` (chữ hoa).
   - **Ví dụ:** `[A-C]` khớp với `A`, `B`, hoặc `C`.
10. `[0-9]`
    - **Ý nghĩa:** Chọn một chữ số trong khoảng `0` đến `9`.
    - **Ví dụ:** `[0-9]` khớp với `0`, `1`, ..., `9`.
11. `[a-z0-9]`
    - **Ý nghĩa:** Chọn một kí tự trong khoảng `a-z` hoặc `0-9`.
    - **Ví dụ:** `[a-z0-9]` khớp với các chữ thường hoặc chữ số, chẳng hạn `a`, `b`, ..., `9`.
12. `[a-zA-Z0-9]`
    - **Ý nghĩa:** Chọn một kí tự trong các khoảng `a-z`, `A-Z` hoặc `0-9`.
    - **Ví dụ:** `[a-zA-Z0-9]` khớp với `a`, `Z`, `6`, v.v.
13. `\n`
    - **Ý nghĩa:** Kí tự xuống dòng (`newline`).
    - **Ví dụ:** Trong mã nguồn, `\n` đại diện cho việc xuống dòng.
14. `\f \r \\`
    - **Ý nghĩa:** Bao gồm `\r` (quay về đầu dòng), `\f` (sang trang mới), và `\\` (dấu gạch chéo ngược).
    - **Ví dụ:** Có thể gặp chúng khi xử lý chuỗi hoặc văn bản đa dòng.
15. **. (dấu chấm)**
    - **Ý nghĩa:** Khớp với bất kỳ kí tự nào trong ASCII (trừ kí tự xuống dòng trong một số chế độ).
    - **Ví dụ:** `'.'` có thể khớp với `a`, `1`, `'?'`, v.v.
16. `~` [0-9]
    - **Ý nghĩa:** Chọn bất kỳ kí tự nào thuộc ASCII nhưng ngoại trừ `0-9`.
    - **Ví dụ:** `~` [0-9] có thể khớp `a`, `'?'`, `'!'`, ...
17. `[a] -> skip`
    - **Ý nghĩa:** Khi bắt gặp kí tự `a`, ta bỏ qua (không sinh token).
    - **Ví dụ:** `'abc'` trong đó `'a'` sẽ bị bỏ qua, còn `bc` có thể được phân tích tiếp.
18. `fragment INT: [0-9]+;`
    - **Ý nghĩa:** `fragment` là một đoạn (rule) chỉ dùng lại bên trong các rule lexer khác, không thể khớp độc lập. Ví dụ, `INT` này mô tả "một hoặc nhiều chữ số".

- **Ví dụ:** `fragment INT: [0-9]+;` có thể được dùng trong rule khác như `NUMBER: INT ('.' INT)?;` để mô tả số nguyên hoặc số thực.

19. `Biểu thức { self.text = self.text[1:-1]; }`

- **Ý nghĩa:** Đoạn code Python trong dấu `{...}` cho phép tùy biến việc xử lý chuỗi vừa khớp. Ví dụ: `self.text[1:-1]` sẽ bỏ đi kí tự đầu và cuối trong `self.text`.
- **Tham khảo:** https://www.w3schools.com/python/python_strings_slicing.asp

2 Lexer

1. Một danh hiệu trong ngôn ngữ lập trình Ruby là một chuỗi các ký tự số, chữ thường và dấu gạch dưới. Nó phải được bắt đầu bằng một dấu gạch dưới hoặc một ký tự chữ thường. Chọn một biểu thức chính quy phù hợp để mô tả danh hiệu nói trên?

   a) $[a-z0-9\_]+$
   b) $[a-zA-Z0-9]+$
   c) $[a-z\_][a-z0-9\_]*$
   d) $[0-9\_][a-z0-9\_]+$

2. Chọn biểu thức chính qui chấp nhận ít nhất tất cả các chuỗi trong tập MATCH nhưng không chấp nhận bất kỳ chuỗi nào trong tập SKIP sau:
   MATCH = Cho, chi, Chung, Che, Chan
   SKIP = Tro, Ching, Chu, Tre, Tran

   a) $[cCT][hr][aeuio]n?g?$
   b) $[cC]h[aoiue]n?g?$
   c) $[Cc]h[oie]|Ch[au]ng?$
   d) $(C|c)h(o|i|e)|Ch(a|u)n?g?$

3. Cho một mô tả từ vựng được định nghĩa trong ANTLR4 như sau:

```
FLOAT_CONSTANT: DIGIT_SEQUENCE EXPONENT? FLOAT_SUFFIX?;
fragment DIGIT_SEQUENCE: DIGIT+ ('.' DIGIT+)?;
fragment EXPONENT: ('e' | 'E') ('+' | '-')? DIGIT+;
fragment FLOAT_SUFFIX: ('f' | 'F' | 'l' | 'L');
fragment DIGIT: [0-9];
```

   Chuỗi nào sau đây là chuỗi nhập đúng cho token *FLOAT_CONSTANT* và đồng thời có giải thích đúng:

   a) 0.0001E-2f, trong đó E-2 được tạo thành từ *EXPONENT*
   b) 6.02e23L, trong đó 2231 được tạo thành từ *EXPONENT*
   c) 0.123-456 và không có thành phần *FLOAT_SUFFIX*
   d) 123.456E+7F, trong đó 123.456E+7 được tạo thành từ *DIGIT_SEQUENCE*

4. Cho một mô tả từ vựng được định nghĩa trong ANTLR4 như sau:

```
UNIVERSE: A* S A A A A A+;
fragment A: D | C | S;
fragment D: [0-9];
fragment C: [a-zA-Z];
fragment S: [@$!%*#?&];
```

   Chuỗi nhập ứng với token *UNIVERSE* có tính chất nào sau đây?

   a) Có ít nhất 4 ký tự và phải có chứa ít nhất một ít tự đặc biệt $(@\$!\%*\#?\&)$
   b) Có ít nhất 6 ký tự và phải có chứa ít nhất một ít tự đặc biệt $(@\$!\%*\#?\&)$
   c) Có nhiều nhất 8 ký tự và phải có chứa ký tự chữ thường hoặc chữ số
   d) Có ít nhất 6 ký tự và khi chứa ký tự chữ thường thì không chưa ký tự chữ hoa và ngược lại

5. Cho biểu thức chính quy $a[\wedge abc]^*c$ và các chuỗi nhập gồm adc, abbc, ayyyyyyyyyc, abc, aabc, axc. Số chuỗi nhập thỏa mãn biểu thức chính quy là

   a) 1
   b) 5
   c) 2
   d) 3

6. Chọn biểu thức chính qui tương đương với biểu thức chính qui sau: $(a|b)^*(abb|b)a$

   a) $(b^*a^*)^*(ab)?ba$
   b) $[a|b]^*[abb|b]a$
   c) $[ab]^*[ab]?ba$
   d) $[ab]^*(ab)^+ba$

7. đoạn mã ngôn ngữ Python sau hãy liệt kê các *token* và số *token*

```
result = (lst[0] * 2) + func(x, y) - (lst[-1] if lst[1] >= -1.2 else lst[2]) % 5 # result
```

   a) Có 38 *token*
   b) Có 43 *token*
   c) Có 40 *token*
   d) Có 45 *token*

Liệt kê *token* Kết quả: _____

8. Cho biểu thức chính qui sau: $(aa)^*(a|b)?(bb)^*$ Hãy chọn các chuỗi thoả thiều thức chính qui trên?

   a) ab, *empty*, aaabb
   b) aab, aabbb, *empty*
   c) abbb, aabbb, *empty*
   d) *empty*, aabb, aabbba

9. Cho biểu thức chính qui sau: $(aa)^+(a|b)?(bb)^+$ Hãy chọn các chuỗi thoả thiều thức chính qui trên?

   a) a, *empty*, aaabb
   b) aabb, aabbb, aaabb
   c) abbb, aabbb, *empty*
   d) *empty*, aabb, aabbba

10. Kết quả các tên *token* bắt được chuỗi 2.1, 2, 0, 01 theo thứ tự trên

```
INT: '0'|[1-9][0-9]*;
FLOAT: INT ('.' INT)?;
```

   a) $FLOAT, INT, INT, ERROR$
   b) $FLOAT, FLOAT, INT, ERROR$
   c) $FLOAT, FLOAT, FLOAT, ERROR$
   d) $FLOAT, INT, INT, INT$

11. Kết quả các tên *token* bắt được chuỗi 2.1, 2, 0, 01 theo thứ tự trên

```
FLOAT: INT ('.' INT)?;
INT: '0'|[1-9][0-9]*;
```

   a) $FLOAT, INT, INT, ERROR$
   b) $FLOAT, FLOAT, INT, ERROR$
   c) $FLOAT, FLOAT, FLOAT, ERROR$
   d) $FLOAT, INT, INT, INT$

12. Kết quả các tên *token* bắt được chuỗi 2.1, 2, 0, 01 theo thứ tự trên

```
fragment INT: '0'|[1-9][0-9]*;
FLOAT: INT ('.' INT)?;
```

   a) $FLOAT, INT, INT, ERROR$
   b) $FLOAT, FLOAT, INT, ERROR$
   c) $FLOAT, FLOAT, FLOAT, ERROR$
   d) $FLOAT, INT, INT, INT$

13. kết quả trả về chuỗi 22.11

```
fragment INT: '0'|[1-9][0-9]*;
FLOAT: INT ('.' INT)? {self.text = self.text[1:-1];};
```

   a) 2.1
   b) 22.1
   c) 2.11
   d) 22.11

14. Số *token* của đoạn *code c* sau

```
if (!a[foo(-2)]) return x[m*foo(3)]++; // code c
```

   a) 23
   b) 24
   c) 25
   d) 26

   Liệt kê *token* Kết quả: _____

15. Kết quả của chuỗi 1 – 2 – 345 – 6

```
INT: '0' | [1-9] ('-' [0-9]*)* [0-9]* {self.text = self.text.replace('-', '')};
```

   a) *error*
   b) 123456
   c) 1 – 2 – 345 – 6
   d) 0

16. Kết quả của chuỗi 1 – 2 – 345 – 6–

```
INT: '0' | [1-9] ('-' [0-9]*)* [0-9]* {self.text = self.text.replace('-', '')};
```

   a) *error*
   b) 123456
   c) 1 – 2 – 345 – 6
   d) 0

17. Biểu thức $[ab]^*$ tương đương với biểu thức nào dưới đây

   a) $[ab]^+$
   b) $[a|b]^*$
   c) $(a|b)^*$
   d) a^*b^*

18. Biểu thức $[ab]^*$ tương đương với biểu thức nào dưới đây

a) $a|b|[ab]^+$ b) $empty|a|b|[ab]^+$ c) $(a|b)^*$ d) a^*b^*

19. Số chuỗi có độ dài nhỏ hơn 4 được sinh ra bởi biểu thức chính quy $(x|y)*y(a|ab)*$ là:

a) 7 b) 12 c) 10 d) 11

20. Cho M là ngôn ngữ chứa các chuỗi không rỗng của các ký tự chữ thường (a-z), trong đó nếu một chuỗi bắt đầu bằng ký tự 'u' thì không được kết thúc bằng ký tự 'u'. Biểu thức chính quy mô tả M là gì?

a) $u*[a-tv-z]*|[a-tv-z]*u*$ b) $u+[a-tv-z]*u*$
c) $[a-tv-z][a-z]*|u[a-z]*[a-tv-z]$ d) $[a-tv-z][a-z]*|u[a-z]+[a-tv-z]$

21. Tên tài khoản trên mạng xã hội Instagram được quy định như sau:

- Có ít nhất 1 ký tự.
- Bao gồm các ký tự thường (a-z), ký tự số (0-9), dấu gạch dưới (_) và các dấu kết thúc (?, !, .)
- Không được bắt đầu hoặc kết thúc bằng các dấu kết thúc
- Không có hai dấu kết thúc liên tiếp nhau.

Biết rằng, L, D, U, P là tên các fragment đã được định nghĩa tương ứng cho ký tự thường, ký tự số, dấu gạch dưới và dấu kết thúc trong ANTLR4. Biểu thức chính quy mô tả tên tài khoản Instagram là gì?

a) $(L|D|U)(P[LDU]*)*$ b) $((L|D|U)(P(L|D|U)+)*$
c) $(L|D|U)+(P(L|D|U)+)*$ d) $(L|D|U)+(P(L|D|U)*)*$

22. Cho một số công dụng sau đây:

(a) Kiểm tra các ràng buộc như biến phải được khai báo trước khi sử dụng

(b) Xác định cấu trúc của chuỗi tokens có phù hợp không

(c) Tách chuỗi nhập thành các chuỗi con ứng với các tokens

(d) Loại bỏ các chuỗi con ứng với khoảng trắng (như dấu blank, tab, chú thích, ...)

(e) Gắn thông tin vị trí (hàng, cột) vào mỗi token

Số vai trò của bộ phân tích từ vựng trong các công dụng trên là

a) 1 b) 2 c) 3 d) 0

23. Chọn biểu thức chính qui tương đương với biểu thức chính qui sau: $a*(ba*b)+a*$

a) $(a*bba*)*a*bba*$ b) $a*ba*(bba*)*ba*$ c) $a*(bb)+a*$ d) $a*(ba*ba*)+$

24. Gọi L là ngôn ngữ gồm các chuỗi khác rỗng trên {a,b} mà không có 3 ký tự b liên tiếp. Hãy chọn biểu thức chính qui mô tả ngôn ngữ L?

a) $(a|bb?a)+b?b?|bb?$ b) $a*(bb?a+)+b?b?|bb?$
c) $a+(b?b?(ab)?)*|bb?$ d) $a*(b?a*b?(ab)?a*)+b?b?$

Áp dụng mô tả sau cho 2 câu sau:

Cho ngôn ngữ X có các mô tả từ vựng trên ANTLR như sau:

```
ID:[a-z]+;
SEP:[[\](),];
NUM:[0-9]+(',' [0-9]+)?;
OP:[+\-*>=<%];
WS:[ \r\n\t]+ -> skip;
```

Cho đoạn mã sau được viết trên ngôn ngữ X:

```
res += (lst[0] * 2) + func(x, y) - (lst[-1] if lst[1] >= -1.2 else lst[2]) % 5
```

25. Số token được phân tích từ vựng trả về khi phân tích từ vựng cho chuỗi trên là

    a) 38
    b) 42
    c) 42
    d) Một giá trị khác hoặc lỗi

26. Chuỗi lexeme của token thứ 25 là:

    a) -1
    b)]
    c) if
    d) lst

Áp dụng mô tả sau cho 5 câu sau:

Cho ngôn ngữ X có các mô tả từ vựng trên ANTLR như sau:

```
@lexer::header {
from lexererr import *
}

ID:[a-z]+;
LPAREN: '(';
RPAREN: ')';
LBRACE: '{';
RBRACE: '}';
LBRACK: '[';
RBRACK: ']';
NUM:[0-9]+(',' [0-9]+)?;
OP:[+\-*>=<%];
WS:[ \r\t]+ -> skip;

NEWLINE: '\r'? '\n' {
    tk = self.preType;
    if (tk):
        list = [/* TODO 1*/]

        if tk in list:
            self.text = /* TODO 2*/
        else:
            self.skip() # bỏ qua kí tự
    else:
        /* TODO 3*/
}
```

Nếu newline trước đó là các num,), },], ID thì bắt token còn lại thì bỏ qua, $self.preType$ lưu type trước đó của token, ví dụ 12] nếu tới] thì $self.preType = self.NUM$

27. Đoạn code /* TODO 1*/

    a) self.NUM, self.RPAREN, self.RBRACE, self.RBRACK. self.ID
    b) self.NUM, self.LPAREN, self.RBRACE, self.RBRACK. self.ID
    c) self.NUM, self.RPAREN, self.RBRACE, self.WS. self.ID
    d) self.NUM, self.OP, self.WS. self.ID

28. Đoạn code /* TODO 2*/

    a) ";"
    b) self.WS
    c) self.NUM
    d) self.text

29. Đoạn code /* TODO 3*/

    a) self.text = ";"
    b) self.text = self.WS
    c) self.skip()
    d) self.text()

30. Cho đoạn mã sau được viết trên ngôn ngữ X:

```
a = b(
2)
```

```
a = 2
```

a) $a, =, b, (, 2,), a, =, 2$
b) $a, =, b, (, 2,), \backslash n, a, =, 2$
c) $a, =, b, (, \backslash n, 2,), \backslash n, a, =, 2, \backslash n$
d) $a, =, b, (, 2,), \backslash n, a, =, 2, \backslash n$