1 Giới thiệu

Ký pháp Ba Lan ngược (Reverse Polish Notation) được phát minh vào khoảng giữa thập kỷ 1950 bởi Charles Hamblin – một triết học gia và khoa học gia máy tính người Úc – dựa theo công trình về ký pháp Ba Lan của nhà Toán học người Ba Lan Jan Łukasiewicz. Hamblin trình bày nghiên cứu của mình tại một hội nghị khoa học vào tháng 6 năm 1957 và chính thức công bố vào năm 1962.

2 Biểu thức hậu tố

Đối với con người, việc tính toán giá trị một biểu thức là tương đối đơn giản . Nhưng để lập trình cho máy tính thực hiện, việc này tương đối phức tạp. Lấy ví dụ biểu thức $5 + ((1 + 2) \times 4) - 3$, để lập trình cho máy để phải tính $(1 + 2)$ trước là rất khó khăn. Vì vậy, cần phải chuyển đổi biểu thức trên sang dạng mà máy tính có thể làm việc dễ dàng, đó là biểu thức hậu tố.

Biểu thức con người sử dụng được gọi là biểu thức trung tố, nghĩa là toán tử sẽ nằm giữa toán hạng. Ví dụ $4 + 5$, toán tử $+$ nằm giữa hai toán hạng là 4 và 5. Còn trong biểu thức hậu tố, toán tử sẽ nằm sau các toán hạng, biểu thức $4 + 5$ sẽ có dạng hậu tố là $4\ 5\ +$, toán tử $+$ nằm sau hai toán hạng là 4 và 5.

Điểm đặc biệt của biểu thức hậu tố là không phải sử dụng dấu () để chỉ ra thứ tự ưu tiên của các phép tính. Điểm này khiến cho việc lập trình việc tính toán biểu thức hậu tố tương đối đơn giản.

3 Ký pháp Ba Lan ngược

Ký pháp Ba Lan ngược là thuật toán dùng để tính toán giá trị của một biểu thức trung tố. Thuật toán này gồm hai phần:

1. Chuyển biểu thức trung tố thành biểu thức hậu tố.
2. Tính toán giá trị của biểu thức hậu tố

Bài tập tuần này chúng ta sẽ làm phần 1.

Để chuyển đổi biểu thức trung tố sang biểu thức hậu tố, ta có thuật toán sau. Để cho đơn giản, chúng ta chỉ xét đến 4 phép toán cơ bản là $+$, $-$, $\times$ và $\div$.

Algorithm 1: Thuật toán Shunting-yard

Input : Biểu thức toán học dạng trung tố
Output: Biểu thức toán học dạng hậu tố

```
1  s ← stack trống
2  res ← ''
   // token là từ chỉ chung toán tử và toán hạng của biểu thức
3  while còn token:
4      đọc lên một token
5      if token là toán hạng:
6          res ← res + token
7      if token là một toán tử:
           // dấu mở ngoặc được xem là có độ ưu tiên thấp nhất
8          while stack không trống and toán tử trên cùng của stack có độ ưu tiên
            không thấp hơn hơn token:
9              o ← s.pop()
10             res ← res + token
11         s.push(token)
12     if token là dấu mở ngoặc (:
13         s.push(token)
14     if token là dấu đóng ngoặc ):
           // pop stack cho đến khi gặp dấu mở ngoặc (
15         while True:
16             o ← s.pop()
17             if o == '(':
                   // gặp dấu mở ngoặc, thoát vòng lặp
18                 break
19             else:
20                 res ← res + token
   // hết token
21 while s không trống:
22     o ← s.pop()
23     res ← res + token
24 return res
```

Minh họa thuật toán để chuyển đổi biểu thức $5 + ((1 + 2) \times 4) - 3$ thành dạng hậu tố.

Token	Hành động	s	res	Ghi chú
5	thêm vào `res`		5	
+	thêm vào `s`	+	5	
(	thêm vào `s`	+ (	5	
(	thêm vào `s`	+ ((	5	
1	thêm vào `res`	+ ((	5 1	
+	thêm vào `s`	+ ((+	5 1	(ưu tiên thấp hơn +
2	thêm vào `res`	+ ((+	5 1 2	
)	pop `s` và thêm vào `res`	+ ((	5 1 2 +	
	pop `s`	+ (	5 1 2 +	gặp dấu (
×	thêm vào `s`	+ (×	5 1 2 +	(ưu tiên thấp hơn ×
4	thêm vào `res`	+ (×	5 1 2 + 4	
)	pop `s` và thêm vào `res`	+ (	5 1 2 + 4 ×	
	pop `s`	+	5 1 2 + 4 ×	gặp dấu (
−	pop `s` và thêm vào `res`		5 1 2 + 4 × +	− cùng ưu tiên với +
	push vào `s`	−	5 1 2 + 4 × +	s trống
3	thêm vào `res`	−	5 1 2 + 4 × + 3	

Hết token, chúng ta pop stack rồi thêm vào `res` cho đến trống thì được biểu thức 5 1 2 + 4 × + 3 −.

Tương tự, chúng ta có dạng hậu tố của biểu thức $3 + 4 \times 2 \div (1 - 5) + (2 \div 3)$ là 3 4 2 × 1 5 − ÷ + 2 3 ÷ +.

Bài tập

Lưu ý: Biểu thức trung tố trong bài thực hành này có cấu trúc:

- Chỉ có các phép toán $+$, $-$, $*$ và $/$.
- Các token cách nhau đúng một khoảng trắng.

Ví dụ: 33␣+␣412␣*␣42␣/␣(␣21␣-␣58␣) (␣ là khoảng trắng).

Bài 1. Viết chương trình python để trả về độ ưu tiên của các toán tử, biết rằng:

- Toán tử (có độ ưu tiên là 10.
- Toán tử $+$ và $-$ có độ ưu tiên là 1.
- Toán tử $*$ và / có độ ưu tiên là 2.

Bài 2. Viết chương trình python để kiểm tra một biểu thức trung tố có thiếu dấu đóng ngoặc hay không?

Bài 3. Viết chương trình python để chuyển biểu thức trung tố sang hậu tố.

Tham khảo

- Shunting-yard Algorithm.
- Chuyển đổi Ba Lan ngược

4 Tính toán biểu thức hậu tố

Để tính toán giả trị của một biểu thức hậu tố, ta có thuật toán sau

Algorithm 2: Tính giá trị của biểu thức hậu tố

Input : Biểu thức toán học dạng hậu tố
Output: Kết quả của biểu thức

```
// Biểu thức hậu tố được lưu trong mảng expr
def calculate_postfix(expr):
    s ← stack trống
    for token in expr:
        if token là số:
            s.push(token)
        if token là toán tử:
            n1 ← s.pop()
            n2 ← s.pop()
            a ← n2 token n1 // vd: nếu token là toán tử +, a ← n1 + n2
            s.push(a)
    return s.pop()
```

Để minh họa, chúng ta sẽ tính toán giá trị biểu thức $5\ 1\ 2\ +\ 4\ \times\ +\ 3\ -$.

Token	**Hành động**	**s**	**Ghi chú**
5	push 5 vào s	5	
1	push 1 vào s	5 1	
2	push 2 vào s	5 1 2	
$+$	pop s 2 lần	5	$n_1 = 2$ $n_2 = 1$
	tính a	5	$a = n_2 + n_1 = 3$
	push a vào s	5 3	
4	push 4 vào s	5 3 4	
$\times$	pop s 2 lần	5	$n_1 = 4$ $n_2 = 3$
	tính a	5	$a = n_2 \times n_1 = 12$
	push a vào s	5 12	
$+$	pop s 2 lần		$n_1 = 12$ $n_2 = 5$
	tính a		$a = n_2 + n_1 = 17$
	push a vào s	17	
3	push 3 vào s	17 3	
$-$	pop s 2 lần		$n_1 = 3$ $n_2 = 17$
	tính a		$a = n_2 - n_1 = 14$
	push a vào s	14	

Bài tập

Bài 1. Chỉnh sửa lại hàm chuyển biểu thức trung tố sang hậu tố để kết quả trả về là dạng mảng.

Bài 2. Bổ sung các toán tử sau vào phần thực hành trước:

- Toán tử %: phép chia nguyên, độ ưu tiên là 2.
- Toán tử $\wedge$: phép lũy thừa, độ ưu tiên là 3.

Bài 3. Viết chương trình để tính toán giá trị biểu thức hậu tố.