SINGLY LINKED LIST

APPLICATION of STACK

DATA STRUCTURES AND ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

Biểu thức trung tố và hậu tố (infix và postfix)

- Trong đa số các ngôn ngữ lập trình, các biểu thức được biểu diễn như trên gọi là ký pháp trung tố (infix). Nên khi xác định giá trị của một biểu thức số học ta đưa ra thuật toán sau. Thuật toán này gồm hai giai đoạn.
  - Chuyển biểu thức số học thông thường (dạng trung tố – infix) sang biểu thức số học dạng hậu tố (postfix – dạng ký pháp nghịch đảo Balan gọi tắt là biểu thức Balan).
  - Tính giá trị của biểu thức số học Ba Lan postfix

- Trong đó biểu thức ở dạng biểu thức Ba Lan thì phép toán được đặt sau các toán hạng.

- Chẳng hạn, các biểu thức a + b, a * b trong cách viết Ba Lan được viết là a b +, a b *.

Biểu thức trung tố và hậu tố (infix và postfix)

Infix:

- **2 + 3**
- **2 + 4 * 5**
- **((2 + 4) * 7) + 3* (9 – 5))**

Postfix

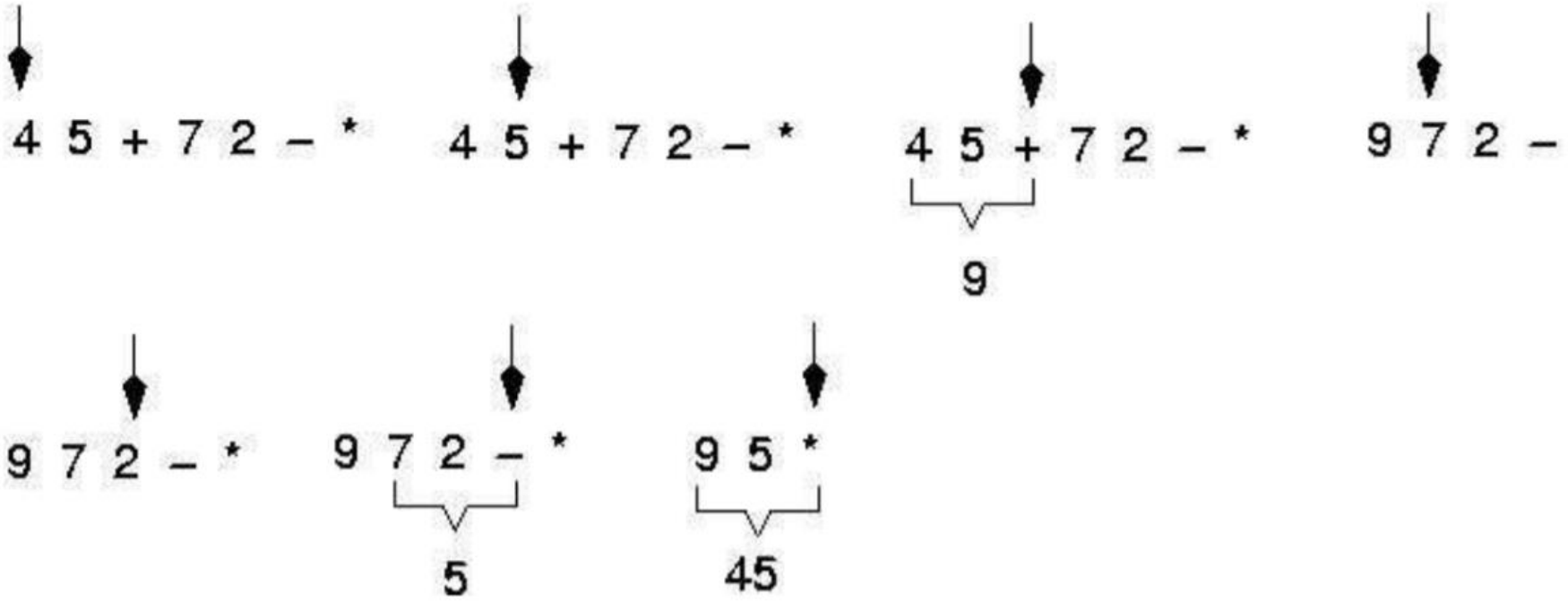
Ký pháp Ba Lan ngược

- **2 3 +**
- **2 4 5 *+**
- **2 4 + 7 * 3 9 5 - * +**

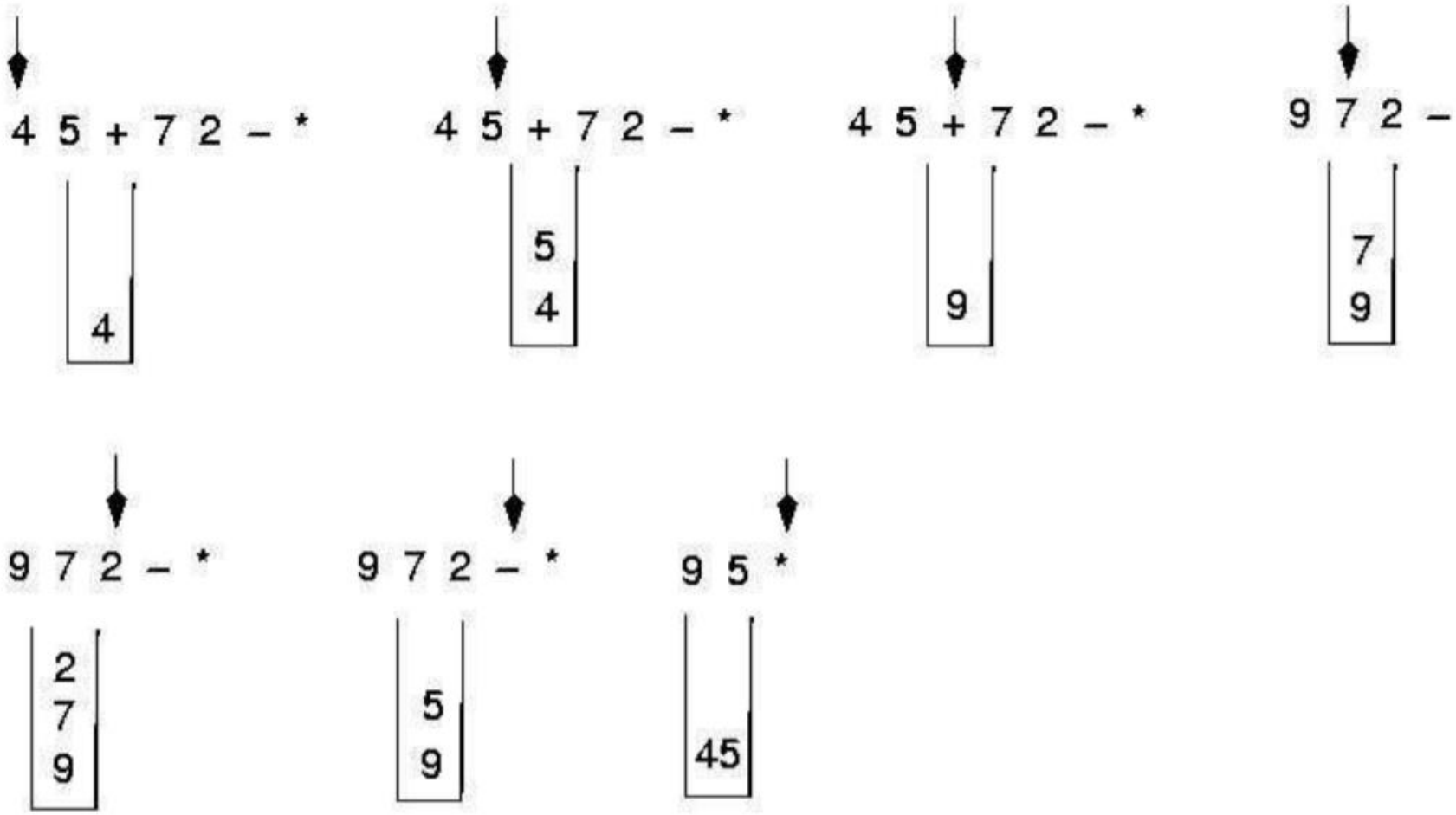
Chú ý: Cần lưu ý rằng, biểu thực số học Ba Lan không chứa các dấu ngoặc, nó chỉ gồm các toán hạng và các dấu phép toán.

Việc tính giá trị của biểu thức ở dạng biểu thức Ba Lan được thực hiện: biểu thức được đọc từ trái sang phải cho đến khi tìm được một toán tử. Tại thời điểm đó, hai toán hạng cuối cùng được đọc kết hợp với toán tử này.

Biểu thức trung tố và hậu tố (infix và postfix)



Biểu thức trung tố và hậu tố (infix và postfix)



Biểu thức trung tố và hậu tố (infix và postfix)

? Tại sao cần chuyển từ dạng biểu thức trung tố sang hậu tố

Độ ưu tiên của các toán tử

- Xét biểu thức chỉ chứa: toán hạng, dấu ngoặc đơn và các toán tử : +, -, *, /, %

Ví dụ: ((2 + 4) * 7) + 3 * (9 - 5))

- *, /, % có độ ưu tiên cao hơn +, -

Tiền xử lý biểu thức infix

- Tách biểu thức infix thành các token dựa vào các khoảng trắng.

Ví dụ 1: ((2 + 4) * 7) + 3* (9 – 5))

=> Các token: (, (, 2, +, 4, *, 7,) , +, 3, *, (, 9, –, 5,),)

Ví dụ 2: A * B + C * ((D - E) + F)/G

=> Các token: A, *, B, +, C, *, (, (, D, -, E,), +, F,), /, G

Thuật toán chuyển từ infix sang postfix

Bước 1: Duyệt các token của infix từ trái sang phải.

Bước 1.1: Nếu là toán hạng: cho vào output.

Bước 1.2: Nếu là dấu mở ngoặc "(": cho vào stack

Bước 1.3: Nếu là dấu đóng ngoặc ")": lấy các toán tử trong stack S ra và cho vào output cho đến khi gặp dấu mở ngoặc "(" (Dấu mở ngoặc cũng phải được đưa ra khỏi stack)

Bước 1.4: Nếu là toán tử:

- Chừng nào ở **đỉnh stack là toán tử có độ ưu tiên lớn hơn hoặc bằng toán tử hiện tại** thì lấy toán tử ra khỏi stack để cho vào output.
- Đưa toán tử hiện tại vào stack

Bước 2: Sau khi duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các token trong đó ra và cho lần lượt vào output.

Algorithm to convert Infix To Postfix

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

1. Push “(“onto Stack, and add “)” to the end of X.

2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.

3. If an operand is encountered, add it to Y.

4. If a left parenthesis is encountered, push it onto Stack.

5. If an operator is encountered ,then:

    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.

    2. Add operator to Stack.
    [End of If]

6. If a right parenthesis is encountered ,then:

    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.

    2. Remove the left Parenthesis.
    [End of If]
    [End of If]

7. END.

https://www.includehelp.com/c/infix-to-postfix-conversion-using-stack-with-c-program.aspx

Ví dụ: Chuyển từ infix sang postfix

Infix: A*B+C*((D-E)+F)/G

Postfix: A B * C E - F + * G / +

Token	Stack	Output
A	Empty	A
*	*	A
B	*	A B
+	+	A B *
C	+	A B * C
*	+ *	A B * C
(	+ * (	A B * C
(	+ * ((	A B * C
D	+ * ((	A B * C D
-	+ * ((-	A B * C
E	+ * ((-	A B * C E
)	+ * (	A B * C E -
+	+ * (+	A B * C E -
F	+ * (+	A B * C E - F
)	+ *	A B * C E - F +
/	+ /	A B * C E - F + *
G	+ /	A B * C E - F + * G
	Empty	A B * C E - F + * G / +

Kiểu dữ liệu của S?

Thuật toán tính giá trị biểu thức postfix

- Duyệt các token của biểu thức postfix từ trái sang phải:
  - Nếu token là toán hạng thì dùng push đưa vào ngăn xếp S.
  - Nếu token là **toán tử** thì pop 2 toán hạng trong ngăn xếp S ra, sau đó tính toán giá trị của chúng dựa vào **toán tử**, sau đó push kết quả vào S.
- Kết quả của biểu thức chính là phần tử còn lại cuối cùng trong ngăn xếp S.

Ví dụ:

Biểu thức: ((2 + 4) * 7) + 3* (9 – 5))

=> postfix: 2, 4, +, 7, *, 3, 9, 5, -, *, +

Ví dụ: Chuyển từ infix sang postfix

Infix: ((2 + 4) * 7) + 3* (9 – 5))
Postfix: 2 4 + 7 * 3 9 5 - * +

STT	Token	Stack	Output
1	(	(	
2	(	((	
3	2	((	2
4	+	((+	2
5	4	((+	2 4
6	)	(	2 4 +
7	*	(*	2 4 +
8	7	(*	2 4 + 7
9	)		2 4 + 7 *
10	+	+	2 4 + 7 *
11	3	+	2 4 + 7 * 3
12	*	+ *	2 4 + 7 * 3
13	(	+ * (	2 4 + 7 * 3
14	9	+ * (	2 4 + 7 * 3 9
15	-	+ * (-	2 4 + 7 * 3 9
16	5	+ * (-	2 4 + 7 * 3 9 5
17	)	+ *	2 4 + 7 * 3 9 5 -
18	)	+	2 4 + 7 * 3 9 5 - *
19		Empty	2 4 + 7 * 3 9 5 - * +

Ví dụ: Tính giá trị từ postfix

Infix: ((2 + 4) * 7) + 3* (9 – 5))
Postfix: 2 4 + 7 * 3 9 5 - * +
⇒ Giá trị:

STT	Token	Stack
1	2	2
2	4	2 4
3	+	6
4	7	6 7
5	*	42
6	3	42 3
7	9	42 3 9
8	5	42 3 9 5
9	-	42 3 4
10	*	42 12
11	+	54
12		Empty

Kiểu dữ liệu của S?

Chúc các em học tốt!