STACK

- Stack ( ngăn xếp )là một cấu trúc dữ liệu trong lập trình máy tính, được sử dụng để lưu trữ và quản lý các đối tượng. Nó hoạt động dựa trên cơ chế "Last In First Out" (LIFO), có nghĩa là phần tử cuối cùng được thêm vào là phần tử đầu tiên được lấy ra khỏi stack. Các thao tác phổ biến với stack bao gồm đẩy (push) một phần tử vào stack và lấy (pop) phần tử trên đầu stack ra để sử dụng.

- Infix là bài toán thông thường như : 1 + 2 + 3 - ( 2 -1 ) \* ( 5 + 6 ) / 9. chỉ là một bài toán thông thường thôi nhưng trong bài thường sẽ bắt chuyển đổi Infix thành postfix trong code hoặc bài thi thôi nên chỉ cần nhớ Infix là gì thôi.

- (Postfix hoặc gọi là Sunfix )Phép tính giá trị của một biểu thức dạng hậu tố (postfix expression), còn được gọi là Notation Ba Lan Đảo (Reverse Polish Notation - RPN), liên quan đến việc tính toán kết quả của một biểu thức toán học sử dụng cách ghi chú cụ thể không đòi hỏi việc sử dụng dấu ngoặc hoặc quy tắc ưu tiên của các phép toán. Trong biểu thức dạng hậu tố, các toán tử đặt sau các toán hạng của chúng. Để tính toán biểu thức dạng hậu tố, bạn có thể sử dụng thuật toán dựa trên ngăn xếp. Dưới đây là cách nó hoạt động:

1. Bắt đầu với một ngăn xếp trống.
2. Quét biểu thức dạng hậu tố từ trái qua phải.
3. Đối với mỗi phần tử (toán hạng hoặc toán tử):

* Nếu đó là toán hạng (một số), đẩy nó lên đỉnh của ngăn xếp.
* Nếu đó là toán tử, lấy ra hai phần tử đầu tiên từ ngăn xếp, áp dụng toán tử lên chúng, và đẩy kết quả trở lại ngăn xếp.

1. Tiếp tục quá trình này cho đến khi bạn đã quét toàn bộ biểu thức dạng hậu tố.
2. Kết quả cuối cùng sẽ là phần tử duy nhất còn lại trên ngăn xếp.

Dưới đây là một ví dụ về cách tính toán một biểu thức dạng hậu tố:

Giả sử bạn có biểu thức dạng hậu tố: "2 3 4 \* +"

1. Bắt đầu với một ngăn xếp trống.
2. Bắt đầu quét biểu thức từ trái qua phải:

* "2" là toán hạng, đẩy nó lên ngăn xếp.
* "3" là một toán hạng khác, đẩy nó lên ngăn xếp.
* "4" cũng là một toán hạng, đẩy nó lên ngăn xếp.
* "\*" là một toán tử; lấy ra hai toán hạng đầu tiên (4 và 3), tính toán 4 \* 3, và đẩy kết quả (12) trở lại ngăn xếp.
* "+" cũng là một toán tử; lấy ra hai toán hạng đầu tiên (12 và 2), tính toán 12 + 2, và đẩy kết quả (14) trở lại ngăn xếp.

1. Bây giờ bạn đã quét toàn bộ biểu thức, và phần tử duy nhất còn lại trên ngăn xếp là kết quả cuối cùng, là 14.

Vậy, biểu thức dạng hậu tố "2 3 4 \* +" có giá trị là 14.

Biểu thức dạng hậu tố rất hữu ích trong lĩnh vực khoa học máy tính và lập trình vì nó không đòi hỏi quy tắc phức tạp về ưu tiên của toán tử và bạn có thể dễ dàng tính toán biểu thức sử dụng ngăn xếp, điều này làm cho nó hiệu quả cho việc triển khai máy tính và trình tính toán biểu thức.

Để chuyển đổi một biểu thức dạng infix sang dạng postfix, bạn cần sử dụng một thuật toán dựa trên ngăn xếp. Dưới đây là các bước để chuyển đổi biểu thức dạng infix sang dạng postfix:

1. Khởi tạo một ngăn xếp trống để lưu trữ các toán tử.
2. Khởi tạo một danh sách trống để lưu trữ biểu thức postfix.
3. Bắt đầu quét biểu thức infix từ trái qua phải.
4. Đối với mỗi phần tử (toán hạng hoặc toán tử):
   * Nếu đó là một toán hạng (số), thêm nó vào danh sách biểu thức postfix.
   * Nếu đó là một toán tử, thực hiện các bước sau:
     + Trong khi ngăn xếp không trống và toán tử ở đỉnh ngăn xếp có độ ưu tiên cao hơn hoặc bằng (theo quy tắc ưu tiên toán tử), thì lấy toán tử ở đỉnh ngăn xếp ra và thêm nó vào danh sách biểu thức postfix.
     + Đẩy toán tử hiện tại vào ngăn xếp.
   * Nếu đó là dấu ngoặc trái "(", đẩy nó vào ngăn xếp.
   * Nếu đó là dấu ngoặc phải ")", lấy ra các toán tử từ ngăn xếp và thêm chúng vào danh sách biểu thức postfix cho đến khi bạn gặp dấu ngoặc trái. Sau đó, lấy ra và loại bỏ dấu ngoặc trái khỏi ngăn xếp.
5. Tiếp tục quá trình này cho đến khi bạn đã quét toàn bộ biểu thức infix.
6. Sau khi quét xong, lấy ra tất cả các toán tử còn lại từ ngăn xếp và thêm chúng vào danh sách biểu thức postfix.
7. Danh sách biểu thức postfix sẽ là kết quả cuối cùng.

Dưới đây là một ví dụ về cách chuyển đổi biểu thức infix "3 + 4 \* 5 - 6 / 2" thành biểu thức postfix:

Biểu thức Infix: "3 + 4 \* 5 - 6 / 2"

1. Khởi tạo một ngăn xếp trống và một danh sách trống để lưu trữ biểu thức postfix.
2. Bắt đầu quét biểu thức từ trái qua phải.
3. Xử lý các phần tử:
   * "3" là một toán hạng, thêm nó vào danh sách biểu thức postfix.
   * "+" là một toán tử, đẩy nó vào ngăn xếp.
   * "4" là một toán hạng, thêm nó vào danh sách biểu thức postfix.
   * "\*" là một toán tử có độ ưu tiên cao hơn "+" nên đẩy nó vào ngăn xếp.
   * "5" là một toán hạng, thêm nó vào danh sách biểu thức postfix.
   * "-" là một toán tử, lấy "\*" ra khỏi ngăn xếp (có độ ưu tiên cao hơn) và thêm nó vào danh sách biểu thức postfix. Sau đó, đẩy "-" vào ngăn xếp.
   * "6" là một toán hạng, thêm nó vào danh sách biểu thức postfix.
   * "/" là một toán tử, lấy "-" ra khỏi ngăn xếp (độ ưu tiên thấp hơn) và thêm nó vào danh sách biểu thức postfix. Sau đó, đẩy "/" vào ngăn xếp.
   * "2" là một toán hạng, thêm nó vào danh sách biểu thức postfix.
4. Sau khi quét xong, lấy ra mọi toán tử còn lại từ ngăn xếp và thêm chúng vào danh sách biểu thức postfix. Trong trường hợp này, bạn lấy "+" ra và thêm vào danh sách postfix.
5. Danh sách biểu thức postfix sẽ là: "3 4 5 \* + 6 2 / -"

Vậy là, biểu thức infix "3 + 4 \* 5 - 6 / 2" đã được chuyển đổi thành biểu thức postfix "3 4 5 \* + 6 2 / -." Biểu thức postfix này có thể được tính toán dễ dàng bằng cách sử dụng một thuật toán dựa trên ngăn xếp, như đã giải thích trong câu trả lời trước.

Chuyển đổi một biểu thức từ dạng postfix sang dạng infix có thể khó hơn so với chuyển từ infix sang postfix vì bạn cần phải theo dõi quy tắc ưu tiên toán tử và sử dụng dấu ngoặc để xác định rõ ràng thứ tự của các phép tính. Dưới đây là cách bạn có thể thực hiện quy trình chuyển đổi này:

Khởi tạo một ngăn xếp để lưu trữ các phần tử tạm thời.

Bắt đầu quét biểu thức postfix từ trái qua phải.

Đối với mỗi phần tử (operand hoặc operator):

* + Nếu phần tử là một operand (số), đẩy nó vào ngăn xếp.
  + Nếu phần tử là một toán tử, lấy ra hai phần tử từ ngăn xếp (ví dụ: "A" và "B") và tạo một biểu thức con theo dạng "A operator B" (ví dụ: "A + B"). Sau đó, đẩy biểu thức con này vào ngăn xếp.

Sau khi bạn đã quét toàn bộ biểu thức postfix, ngăn xếp sẽ chỉ còn một biểu thức infix duy nhất.

Lấy ra biểu thức infix từ ngăn xếp, và đó là kết quả cuối cùng.

Dưới đây là một ví dụ để chuyển đổi biểu thức postfix "3 4 5 \* + 6 2 / -" thành biểu thức infix:

Khởi tạo một ngăn xếp.

Bắt đầu quét biểu thức từ trái qua phải.

Xử lý các phần tử:

* + "3" là một toán hạng, đẩy nó vào ngăn xếp.
  + "4" cũng là một toán hạng, đẩy nó vào ngăn xếp.
  + "5" cũng là một toán hạng, đẩy nó vào ngăn xếp.
  + "\*" là một toán tử, lấy ra hai toán hạng cuối cùng ("4" và "5") và tạo biểu thức con "4 \* 5". Đẩy biểu thức con này vào ngăn xếp.
  + "+" là một toán tử, lấy ra hai toán hạng cuối cùng ("3" và biểu thức con "4 \* 5") và tạo biểu thức con "3 + (4 \* 5)". Đẩy biểu thức con này vào ngăn xếp.
  + "6" cũng là một toán hạng, đẩy nó vào ngăn xếp.
  + "2" cũng là một toán hạng, đẩy nó vào ngăn xếp.
  + "/" là một toán tử, lấy ra hai toán hạng cuối cùng ("6" và "2") và tạo biểu thức con "6 / 2". Đẩy biểu thức con này vào ngăn xếp.
  + "-" là một toán tử, lấy ra hai toán hạng cuối cùng (biểu thức con "3 + (4 \* 5)" và biểu thức con "6 / 2") và tạo biểu thức con "3 + (4 \* 5) - (6 / 2)". Đẩy biểu thức con này vào ngăn xếp.

Sau khi quét xong, ngăn xếp chỉ còn một biểu thức infix duy nhất: "3 + (4 \* 5) - (6 / 2)".

Lấy ra biểu thức infix từ ngăn xếp, và kết quả cuối cùng là "3 + (4 \* 5) - (6 / 2)".

Chuyển đổi từ postfix sang infix có thể mất thời gian và đòi hỏi kiên nhẫn, đặc biệt đối với các biểu thức phức tạp.

QUEUE

Trong lập trình và khoa học máy tính, "queue" (hàng đợi) là một cấu trúc dữ liệu được sử dụng để lưu trữ và quản lý các phần tử theo nguyên tắc "First-In-First-Out" (FIFO), nghĩa là phần tử đầu tiên được đưa vào hàng đợi sẽ là phần tử đầu tiên được lấy ra khỏi hàng đợi.