Ngăn Xếp và Hàng Đợi

Mục tiêu

Bài thực hành dưới đây sẽ giúp sinh viên nắm vững cách thức lập trình và vận hành của hai loại cấu trúc dữ liệu là ngăn xếp và hàng đợi. Các bài tập mở rộng của bài thực hành cũng giúp sinh viên xây dựng khả năng tư duy sáng tạo trong việc áp dụng hai loại cấu trúc dữ liệu này vào giải quyết các bài toán và liên tưởng đến các ứng dụng thực tế.

Nội dung

Câu 1: Hãy cài đặt lại một cấu trúc dữ liệu ngăn xếp (Stack) được biểu diễn bằng mảng một chiều (one-dimensional array) và viết các phương thức để hiện thực các thao tác cơ bản trên đó:

- Thêm một phần tử vào Stack (push)
- Lấy một phần tử ra khỏi Stack (pop)
- Xem giá trị của phần tử ở đỉnh Stack (peek)
- Cho biết số lượng phần tử trong Stack (getSize)
- Kiểm tra xem Stack có đầy hay không (isFull)
- Kiểm tra xem Stack có rỗng hay không (isEmpty)

Hướng dẫn: tham khảo cách xây dựng Stack bằng mảng 1 chiều đã được học trên lớp (không sử dụng CTDL Stack có sẵn của Java) và xây dựng Stack để lưu trữ các phần tử thuộc kiểu số nguyên (Integer)

Câu 2: Hãy cài đặt lại một cấu trúc dữ liệu hàng đợi (Queue) được biểu diễn bằng mảng một chiều (one-dimensional array) và viết các phương thức để hiện thực các thao tác cơ bản trên đó:

- Thêm một phần tử vào Queue (enQueue)
- Lấy một phần tử ra khỏi Queue (deQueue)
- Xem giá trị của phần tử ở đầu Queue (frontQueue)
- Xem giá trị của phần tử ở đuôi Queue (rearQueue)
- Cho biết số lượng phần tử có trong Queue (getSize)
- Kiểm tra xem Queue có đầy hay không (isFull)
- Kiểm tra xem Queue có rỗng hay không (isEmpty)

Hướng dẫn: tham khảo cách xây dựng Queue bằng mảng 1 chiều đã được học trên lớp (không sử dụng các lớp biểu diễn CTDL Queue có sẵn của Java) và xây dựng Queue để lưu trữ các phần tử thuộc kiểu số nguyên (Integer)

Câu 3: Bài toán "Đảo ngược chuỗi kí tự"

Cho một chuỗi có *n* kí tự. Ứng dụng cấu trúc dữ liệu ngăn xếp (Stack) trong thư viện java.util.Stack đã được học, hãy xây dựng thuật giải để thực hiện việc đảo ngược các kí tự của chuỗi trên.

Ví dụ: cho chuỗi "ENVIRONMENT" thì chuỗi đảo ngược của chuỗi trên sẽ là "TNEMNORIVNE"

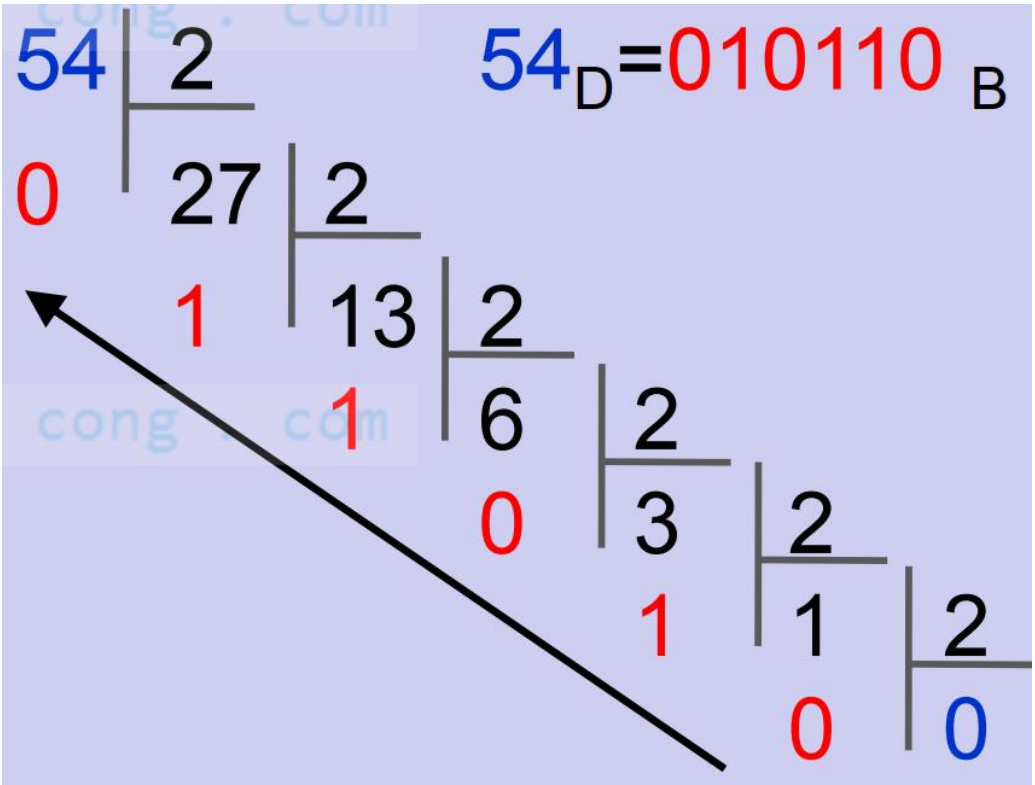
Dữ liệu đầu vào: chuỗi kí tự cần đảo ngược

Dữ liệu đầu ra: chuỗi đã được đảo ngược

Câu 4: Tương tự như câu 1, hãy xây dựng một thuật giải sử dụng cấu trúc dữ liệu kiểu ngăn xếp (Stack) để thực hiện phép toán chuyển đổi một số thập phân (số ở hệ cơ số 10) ra thành một dãy số nhị phân

Ví dụ: số 54_{10} có thể được chuyển thành 010110_2

Cách thức chuyển: lấy số thập phân chia cho 2 để được một thương số, rồi lại lấy thương số đó chia tiếp cho 2. Cứ tiếp tục như vậy cho đến khi thương số bằng 0. Dãy số được tạo bởi các số dư từ cuối về đầu của các phép chia trên chính là dãy số nhị phân cần tìm.



Câu 5: Tính toán biểu thức bằng phương pháp nghịch đảo kí tự Ba Lan (Reserved Polish Notation)

Việc lập trình để cho máy tính tính được giá trị của một biểu thức toán học là một điều bình thường, nhưng việc trình bày làm sao để máy tính có thể đọc và thực hiện đúng qui trình tính toán thì là một điều không hề đơn giản. Với các biểu thức toán học đơn giản như tính a + b – c + d thì ta có thể lập trình cho máy tính thực hiện một cách đơn giản là lần lượt đọc các toán hạng và toán tử từ trái qua phải rồi thực hiện tính toán. Nhưng để giải quyết được các biểu thức có dấu ngoặc, chẳng hạn như a + b – (c + d) * e thì việc tính toán như vậy là không khả thi. Trong tình huống này, ta phải sử dụng đến một thuật toán có tên là Reserve Polish Notation – RPN, tiếng Việt được gọi là nghịch đảo kí tự Ba Lan, đây là thuật toán kinh điển trong lĩnh vực trình biên dịch hay xử lý ngôn ngữ tự nhiên.

Giả sử biểu thức cần tính toán là một chuỗi số được nhập từ bàn phím. Các kí tự trong chuỗi chỉ bao gồm: các dấu đóng “(“ hoặc mở ngoặc “)”; các toán tử +, -, *, / và các toán hạng là những con số nguyên được hình thành từ các chữ số 0 – 9.

Ví dụ: Chuỗi “5 + ((1 + 2) * 4) + 3” là một biểu thức nhập vào

Ta gọi các biểu thức mà toán tử nằm giữa 2 toán hạng là một biểu thức trung tố. Ví dụ: a + b là một biểu thức trung tố. Đây là dạng biểu thức được biểu diễn theo cách thông thường và tự nhiên đối với con người, nhưng lại khá khó chịu đối với máy tính. Để giải quyết vấn đề này giải thuật RPN đưa ra một cách thức để biến đổi biểu thức nói trên thành một dạng biểu thức có tên là biểu thức hậu tố - tức là biểu thức mà toán tử nằm sau 2 toán hạng.
Ví dụ: a + b → a b +

Quá trình tính toán biểu thức hậu tố này diễn ra khá tự nhiên và dễ dàng đối với máy tính. Ý tưởng là đọc biểu thức từ trái sang phải, nếu gặp một toán hạng (con số hoặc biến) thì push vào ngăn xếp (Stack); nếu gặp toán tử thì pop 2 toán hạng ra khỏi ngăn xếp (Stack), tính toán và đẩy kết quả trở lại ngăn xếp. Khi quá trình kết thúc thì giá trị còn lại cuối cùng trong ngăn xếp chính là giá trị của biểu thức cần tính toán.

Ví dụ: biểu thức trung tố 6 + ((7 + 2) * 3) + 4
được biểu diễn lại dưới dạng hậu tố là 6 7 2 + 3 * + 4 +

Thuật toán để xây dựng biểu thức trung tố thành hậu tố:

B1: Khởi tạo một Stack rỗng và một Queue rỗng

B2: Dùng phương thức split() để cắt chuỗi theo dấu cách thành một mảng các chuỗi, đặt tên là array

B3: Cho vòng lặp duyệt tuần tự từng phần tử của array:

- Nếu gặp phần tử là toán hạng (0...9), ta push vào Queue
- Nếu gặp phần tử là toán tử (+, -, *, /) ta xét:
  - Cứ khi nào ở đỉnh Stack là toán tử có độ ưu tiên lớn hơn toán tử đang xét thì ta pop toán tử đó ra khỏi Stack và push vào Queue (Ví dụ: phép * có độ ưu tiên cao hơn phép +; phép + xuất hiện trước có độ ưu tiên cao hơn phép + xuất hiện sau;…)
  - Push toán tử hiện tại vào Stack
- Nếu gặp phần tử là dấu (ta push vào Stack
- Nếu gặp phần tử là dấu) thì pop từng toán tử trong Stack ra và push vào Queue cho tới khi gặp được dấu mở ngoặc (. Lưu ý: ta pop luôn cả dấu (ra khỏi Stack

B4: Sau khi kết thúc vòng lặp ở B3 nếu trong Stack vẫn còn phần tử thì ta sẽ lần lượt pop từng phần tử này ra và push vào Queue cho đến khi Stack rỗng

B5: Đảo ngược danh sách các phần tử trong Queue

Ví dụ: chuyển đổi biểu thức 6 + ((7 + 2) * 3) + 4 thành biểu thức hậu tố:

Ký tự	Thao tác	Stack	Queue
6	push vào queue		6
+	push vào stack	+	6
(	push vào stack	+(	6
(	push vào stack	+((	6
7	push vào queue	+((	7 6
+	push vào stack	+((+	7 6
2	push vào queue	+((+	2 7 6
)	pop dữ liệu từ stack ra và push vào queue cho tới khi gặp được dấu (	+(	+ 2 7 6
*	push vào stack	+(*	+ 2 7 6
3	push vào queue	+(*	3 + 2 7 6
)	pop dữ liệu từ stack ra và push vào	+	* 3 + 2 7 6

	queue cho tới khi gặp được dấu (		
+	toán tử + ở đỉnh Stack có độ ưu tiên cao hơn (do xuất hiện trước) nên pop ra đưa vào queue và push toán tử + hiện tại vào Stack	+	+ * 3 + 2 7 6
4	push vào queue	+	
	Stack vẫn còn phần tử nên ta pop ra và push vào queue		+ 4 + * 3 + 2 7 6

Như vậy sau khi thực hiện xong, danh sách các phần tử trong Queue được đảo ngược tạo thành một biểu thức hậu tố 6 7 2 + 3 * + 4 +

Để tính toán giá trị của biểu thức hậu tố trên, ta có thuật toán như sau:
Gọi A là mảng chứa các phần tử biểu diễn biểu thức hậu tố
B1: Khởi tạo một Stack rỗng
B2: Duyệt qua từng phần tử trên mảng A:

- Nếu gặp phần tử là toán hạng, ta push vào Stack
- Nếu gặp phần tử là toán tử, ta pop hai phần tử ở đỉnh Stack ra và tính giá trị của biểu thức với toán tử hiện tại, cuối cùng push kết quả vào Stack

B3: Phần tử còn lại duy nhất trong Stack chính là kết quả của biểu thức, ta pop phần tử này ra khỏi Stack.

Ví dụ: tính toán biểu thức hậu tố: 6 7 2 + 3 * + 4 +

Ký tự	Thao tác	Stack
6	push vào Stack	6
7	push vào Stack	6 7
2	push vào Stack	6 7 2

+	pop 2 phần tử 2 7 ở đỉnh Stack và thực hiện biểu thức 7 + 2 = 9. Push 9 vào Stack	6 9
3	push vào Stack	6 9 3
*	pop 2 phần tử 3 9 ở đỉnh Stack và thực hiện biểu thức 9 * 3 = 27. Push 27 vào Stack	6 27
+	pop 2 phần tử 27 6 ở đỉnh Stack và thực hiện biểu thức 6 + 27 = 33. Push 33 vào Stack	33
4	push vào Stack	33 4
+	pop 2 phần tử 4 33 ở đỉnh Stack và thực hiện biểu thức 33 + 4 = 37. Push 37 vào Stack	37
	pop phần tử 37 ra khỏi Stack và xuất kết quả	

Như vậy kết quả của biểu thức 6 + ((7 + 2) * 3) + 4 là 37

Yêu cầu: Nhập vào một chuỗi biểu thức, mỗi toán hạng và toán tử trong biểu thức cách nhau một khoảng trắng (spacebar/white space). Hãy tính kết quả của biểu thức đó và xuất ra màn hình.