BÀI TẬP STACK & QUEUE - DIỄN GIẢI CHI TIẾT

# Bài 1. Quản lý Stack bằng danh sách liên kết đơn

Yêu cầu: Quản lý một ngăn xếp (Stack) chứa các số nguyên. Mỗi phần tử là một node trong danh sách liên kết đơn.

Cách làm:  
1. Stack là cấu trúc LIFO (Last In – First Out): phần tử vào sau ra trước.  
2. Mỗi phần tử là một node gồm dữ liệu (int) và con trỏ next.  
3. Con trỏ top trỏ tới phần tử trên cùng.  
4. Thao tác chính:  
 • Khởi tạo: top = NULL.  
 • Kiểm tra rỗng: top == NULL.  
 • Thêm (push): tạo node mới, nối vào đầu danh sách.  
 • Xóa (pop): xóa node đầu tiên, cập nhật top.  
5. Ứng dụng: Đổi số thập phân sang nhị phân bằng cách chia 2 liên tục, đưa phần dư vào stack, sau đó lấy ra ngược lại.

Kết quả minh họa:

13 → 1101  
10 → 1010  
8 → 1000

# Bài 2. Quản lý Queue bằng danh sách liên kết đơn

Yêu cầu: Cài đặt hàng đợi (Queue) có các phần tử kiểu số nguyên.

Cách làm:  
1. Queue là cấu trúc FIFO (First In – First Out): phần tử vào trước ra trước.  
2. Mỗi phần tử là node có data và con trỏ next.  
3. Có hai con trỏ: front (đầu hàng), rear (cuối hàng).  
4. Thao tác chính:  
 • Khởi tạo: front = rear = NULL.  
 • Kiểm tra rỗng: front == NULL.  
 • Thêm (enqueue): thêm node vào cuối hàng.  
 • Xóa (dequeue): xóa node ở đầu hàng.

Kết quả minh họa:

Enqueue 10 → 10  
Enqueue 20 → 10 → 20  
Enqueue 30 → 10 → 20 → 30  
Dequeue → 20 → 30

# Bài 3. Stack bằng mảng (Array Stack)

Yêu cầu: Cài đặt stack số nguyên bằng mảng có kích thước cố định (MAX = 100).

Cách làm:  
1. Sử dụng mảng data[] để lưu phần tử.  
2. Dùng biến top để đánh dấu phần tử trên cùng (khởi tạo -1).  
3. Thao tác chính:  
 • PUSH: tăng top và thêm phần tử.  
 • POP: giảm top để loại bỏ phần tử trên cùng.  
 • TOP: xem giá trị phần tử trên cùng.  
 • SIZE: đếm số phần tử = top + 1.  
 • Nếu đầy → FULL; nếu rỗng → EMPTY.

Kết quả minh họa:

Input:  
PUSH 10  
PUSH 20  
TOP  
POP  
SIZE  
END  
Output:  
20  
1

# Bài 4. Kiểm tra dấu ngoặc & thẻ HTML-like

Phần a) Dấu ngoặc:  
1. Duyệt chuỗi ký tự:  
 • Gặp ngoặc mở ((, [, {) → đưa vào stack.  
 • Gặp ngoặc đóng (), ], } → lấy phần tử trên cùng ra và so khớp.  
2. Nếu cuối cùng stack rỗng → hợp lệ.

Ví dụ:  
([{}]) → YES  
([)] → NO

Phần b) Thẻ HTML-like:  
1. Khi gặp <tag> → push tên thẻ vào stack.  
2. Khi gặp </tag> → lấy ra và so sánh.  
3. Nếu cuối cùng stack rỗng → các thẻ đóng mở đúng.

Ví dụ:  
<b><i>ok</i></b> → YES  
<b><i></b></i> → NO

# Bài 5. Chuyển trung tố → hậu tố & tính giá trị

Cách làm:  
1. Trung tố (infix): toán tử nằm giữa hai toán hạng (vd: 3 + 4).  
2. Hậu tố (postfix): toán tử nằm sau hai toán hạng (vd: 3 4 +).  
3. Dùng thuật toán Shunting-yard:  
 • Gặp toán hạng → ghi ra (đưa vào kết quả).  
 • Gặp toán tử → xét độ ưu tiên, đẩy vào stack.  
 • Gặp ngoặc → xử lý riêng (mở thì push, đóng thì pop ra đến mở).  
4. Khi hết chuỗi → đưa toàn bộ toán tử trong stack ra.  
5. Đánh giá biểu thức hậu tố bằng cách dùng stack tính toán.

Kết quả minh họa:

Trung tố: 3 + 4 \* (2 - 1)  
Hậu tố: 3 4 2 1 - \* +  
Giá trị: 7  
  
Trung tố: (5 + 2) \* 3  
Hậu tố: 5 2 + 3 \*  
Giá trị: 21