CHƯƠNG 4

**CẤU TRÚC DỮ LIỆU NGĂN XẾP (STACK) VÀ HÀNG ĐỢI (QUEUE)**

* **Mục tiêu học tập:** Sau khi học xong chương này, người học có thể:
  + Vận dụng và cài đặt được các phép toán trên ngăn xếp.
  + Vận dụng và cài đặt được các phép toán trên hàng đợi.

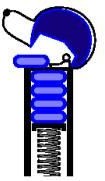
## 4.1. NGĂN XẾP (STACK)

Stack là một vật chứa (container) các đối tượng làm việc theo cơ chế LIFO (Last In First Out) nghĩa là việc thêm một đối tượng vào stack hoặc lấy một đối tượng ra khỏi stack được thực hiện theo cơ chế "Vào sau ra trước".

Các đối tượng có thể được thêm vào stack bất kỳ lúc nào nhưng chỉ có đối tượng thêm vào sau cùng mới được phép lấy ra khỏi stack.

Thao tác thêm 1 đối tượng vào stack thường được gọi là "Push". Thao tác lấy 1 đối tượng ra khỏi stack gọi là "Pop".

Trong tin học, CTDL stack có nhiều ứng dụng: khử đệ qui, tổ chức lưu vết các quá trình tìm kiếm theo chiều sâu và quay lui, vét cạn, ứng dụng trong các bài toán tính toán biểu thức, .



Một hình ảnh một stack

Ta có thể định nghĩa CTDL stack như sau: stack là một CTDL trừu tượng (ADT) tuyến tính hỗ trợ 2 thao tác chính:

Push(o): Thêm đối tượng o vào đầu stack

Pop(): Lấy đối tượng ở đầu stack ra khỏi stack và trả về giá trị của nó. Nếu stack rỗng thì lỗi sẽ xảy ra.

Ngoài ra, stack cũng hỗ trợ một số thao tác khác:

isEmpty(): Kiểm tra xem stack có rỗng không.

Top(): Trả về giá trị của phần tử nằm ở đầu stack mà không hủy nó khỏi stack. Nếu stack rỗng thì lỗi sẽ xảy ra.

Các thao tác thêm, trích và huỷ một phần tử chỉ được thực hiện ở cùng một phía của Stack do đó hoạt động của Stack được thực hiện theo nguyên tắc LIFO (Last In First Out - vào sau ra trước).

Ðể biểu diễn Stack, ta có thể dùng mảng 1 chiều hoặc dùng danh sách liên kết.

***4.1. 1. Biểu diễn Stack dùng mảng:***

Ta có thể tạo một stack bằng cách khai báo một mảng 1 chiều với kích thước tối đa là N (ví dụ, N có thể bằng 1000).

Như vậy stack có thể chứa tối đa N phần tử đánh số từ 0 đến N -1. Phần tử nằm ở đầu stack sẽ có chỉ số t (lúc đó trong stack đang chứa t+1 phần tử)

Ðể khai báo một stack, ta cần một mảng 1 chiều S, biến nguyên t cho biết chỉ số của đầu stack và hằng số N cho biết kích thước tối đa của stack.



Tạo stack S và quản lý đỉnh stack bằng biến t:

Data S [N]; Int t;

Việc cài đặt stack thông qua mảng một chiều đơn giản và khá hiệu quả.

Tuy nhiên, hạn chế lớn nhất của phương án cài đặt này là giới hạn về kích thước của stack N. Giá trị của N có thể quá nhỏ so với nhu cầu thực tế hoặc quá lớn sẽ làm lãng phí bộ nhớ.

***4.1. 2. Biểu diễn Stack dùng danh sách:***

Ta có thể tạo một stack bằng cách sử dụng một danh sách liên kết đơn (danh sách liên kết). Có thể nói, danh sách liên kết có những đặc tính rất phù hợp để dùng làm stack vì mọi thao tác trên stack đều diễn ra ở đầu stack.

Sau đây là các thao tác tương ứng cho list-stack:

**Tạo Stack S rỗng**

LIST \* S;

Lệnh S.pHead=l.pTail= NULL tạo ra một Stack S rỗng.

**Kiểm tra stack rỗng :**

char IsEmpty(LIST &S)

{

if (S.pHead == NULL) // stack rỗng return 1;

else return 0;

}

**Thêm một phần tử p vào stack S**

void Push(LIST &S, Data x)

{

InsertHead(S, x);

}

**Trích huỷ phần tử ở đỉnh stack S**

Data Pop(LIST &S)

{ Data x;

if(isEmpty(S)) return NULLDATA; x = RemoveFirst(S);

return x;

}

**Xem thông tin của phần tử ở đỉnh stack S**

Data Top(LIST &S)

{ if(isEmpty(S)) return NULLDATA; return l.Head->Info;

}

* 1. ***3. Ứng dụng của Stack***
     + Cấu trúc Stack thích hợp lưu trữ các loại dữ liệu mà trình tự truy xuất ngược với trình tự lưu trữ, do vậy một số ứng dụng sau thường cần đến stack :
     + Trong trình biên dịch (thông dịch), khi thực hiện các thủ tục, Stack được sử dụng để lưu môi trường của các thủ tục.
     + Trong một số bài toán của lý thuyết đồ thị (như tìm đường đi), Stack cũng thường được sử dụng để lưu dữ liệu khi giải các bài toán này.

Ngoài ra, Stack cũng còn được sử dụng trong trường hợp khử đệ qui đuôi.

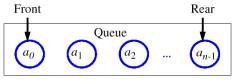
## HÀNG ĐỢI (QUEUE)

Hàng đợi là một vật chứa (container) các đối tượng làm việc theo cơ chế FIFO (First In First Out) nghĩa là việc thêm một đối tượng vào hàng đợi hoặc lấy một đối tượng ra khỏi hàng đợi được thực hiện theo cơ chế "Vào trước ra trước".

Các đối tượng có thể được thêm vào hàng đợi bất kỳ lúc nào nhưng chỉ có đối tượng thêm vào đầu tiên mới được phép lấy ra khỏi hàng đợi.

Thao tác thêm một đối tượng vào hàng đợi và lấy một đối tượng ra khỏi hàng đợi lần lượt được gọi là "enqueue" và "dequeue".

Việc thêm một đối tượng vào hàng đợi luôn diễn ra ở cuối hàng đợi và một phần tử luôn được lấy ra từ đầu hàng đợi.



Trong tin học, CTDL hàng đợi có nhiều ứng dụng: khử đệ qui, tổ chức lưu vết các quá trình tìm kiếm theo chiều rộng và quay lui, vét cạn, tổ chức quản lý và phân phối tiến trình trong các hệ điều hành, tổ chức bộ đệm bàn phím, .

Ta có thể định nghĩa CTDL hàng đợi như sau: hàng đợi là một CTDL trừu tượng (ADT) tuyến tính. Tương tự như stack, hàng đợi hỗ trợ các thao tác:

EnQueue(o): Thêm đối tượng o vào cuối hàng đợi

DeQueue(): Lấy đối tượng ở đầu queue ra khỏi hàng đợi và trả về giá trị của nó.

Nếu hàng đợi rỗng thì lỗi sẽ xảy ra.

IsEmpty(): Kiểm tra xem hàng đợi có rỗng không.

Front(): Trả về giá trị của phần tử nằm ở đầu hàng đợi mà không hủy nó. Nếu hàng đợi rỗng thì lỗi sẽ xảy ra.

Các thao tác thêm, trích và huỷ một phần tử phải được thực hiện ở 2 phía khác nhau của hàng đợi do đó hoạt động của hàng đợi được thực hiện theo nguyên tắc FIFO (First In First Out - vào trước ra trước).

Cũng như stack, ta có thể dùng cấu trúc mảng 1 chiều hoặc cấu trúc danh sách liên kết để biểu diễn cấu trúc hàng đợi.

* + 1. ***Biểu diễn dùng mảng:***

Ta có thể tạo một hàng đợi bằng cách sử dụng một mảng 1 chiều với kích thước tối đa là N (ví dụ, N có thể bằng 1000) theo kiểu xoay vòng (coi phần tử an-1 kề với phần tử a0).

Như vậy hàng đợi có thể chứa tối đa N phần tử. Phần tử nằm ở đầu hàng đợi (front element) sẽ có chỉ số f. Phần tử nằm ở cuối hàng đợi (rear element) sẽ có chỉ số r (xem hình).

Ðể khai báo một hàng đợi, ta cần một mảng một chiều Q, hai biến nguyên f, r cho biết chỉ số của đầu và cuối của hàng đợi và hằng số N cho biết kích thước tối đa của hàng đợi. Ngoài ra, khi dùng mảng biểu diễn hàng đợi, ta cũng cần một giá trị đặc biệt để gán cho những ô còn trống trên hàng đợi. Giá trị này là một giá trị nằm ngoài miền xác định của dữ liệu lưu trong hàng đợi. Ta ký hiệu nó là NULLDATA như ở những phần trước.

Trạng thái hàng đợi lúc bình thường:



Trạng thái hàng đợi lúc xoay vòng:



Câu hỏi đặt ra: khi giá trị f=r cho ta điều gì ? Ta thấy rằng, lúc này hàng đợi chỉ có thể ở

một trong hai trạng thái là rỗng hoặc đầy. Coi như một bài tập các bạn hãy tự suy nghĩ tìm câu trả lời trước khi đọc tiếp để kiểm tra kết quả.

Hàng đợi có thể được khai báo cụ thể như sau:

Data Q[N] ; int f, r;

Cũng như stack, do khi cài đặt bằng mảng một chiều, hàng đợi có kích thước tối đa nên ta cần xây dựng thêm một thao tác phụ cho hàng đợi:

IsFull(): Kiểm tra xem hàng đợi có đầy chưa.

* + 1. ***Dùng danh sách liên kết:***

Ta có thể tạo một hàng đợi bằng cách sử dụng một danh sách liên kết đơn.

Phần tử đầu danh sách liên kết (head) sẽ là phần tử đầu hàng đợi, phần tử cuối danh sách liên kết (tail) sẽ là phần tử cuối hàng đợi.

Sau đây là các thao tác tương ứng cho hàng đợi sử dụng danh sách liên kết đơn:

**Tạo hàng đợi rỗng:**

Lệnh Q.pHead = Q.pTail = NULL tạo ra một hàng đợi rỗng.

**Kiểm tra hàng đợi rỗng :**

char IsEmpty(LIST Q)

{

if (Q.pHead == NULL) // stack rỗng return 1;

else return 0;

}

**Thêm một phần tử p vào cuối hàng đợi**

void EnQueue(LIST Q, Data x)

{

InsertTail(Q, x);

}

**Trích/Hủy phần tử ở đầu hàng đợi**

Data DeQueue(LIST Q)

{

Data x;

if (IsEmpty(Q)) return NULLDATA; x = RemoveFirst(Q);

return x;

}

**Xem thông tin của phần tử ở đầu hàng đợi**

Data Front(LIST Q)

{

if (IsEmpty(Q)) return NULLDATA; return Q.pHead->Info;

}

O(n).

Các thao tác trên đều làm việc với chi phí O(1).

Lưu ý, nếu không quản lý phần tử cuối danh sách, thao tác dequeue sẽ có độ phức tạp

* + 1. ***Ứng dụng của hàng đợi:***
* Hàng đợi có thể được sử dụng trong một số bài toán:
* Bài toán sản xuất và tiêu thụ (ứng dụng trong các hệ điều hành song song).
* Bộ đệm (ví dụ: Nhấn phím -> Bộ đệm -> CPU xử lý).
* Xử lý các lệnh trong máy tính (ứng dụng trong HÐH, trình biên dịch), hàng đợi các tiến trình chờ được xử lý, …
  + - * **Bài tập củng cố:**

1. Hãy cho biết nội dung của stack sau mỗi thao tác trong dãy :

EAS\*Y\*\*QUE\*\*\*ST\*\*\*I\*ON

Với một chữ cái tượng trưng cho thao tác thêm chữ cái tương ứng vào stack, dấu \* tượng trưng cho thao tác lấy nội dung một phần tử trong stack in lên màn hình.

Hãy cho biết sau khi hoàn tất chuỗi thao tác, những gì xuất hiện trên màn hình ?

1. Hãy cho biết nội dung của hàng đợi sau mỗi thao tác trong dãy :

EAS\*Y\*\*QUE\*\*\*ST\*\*\*I\*ON

Với một chữ cái tượng trưng cho thao tác thêm chữ cái tương ứng vào hàng đợi, dấu \* tượng trưng cho thao tác lấy nội dung một phần tử trong hàng đợi in lên màn hình.

Hãy cho biết sau khi hoàn tất chuỗi thao tác, những gì xuất hiện trên màn hình ?

1. Hãy dùng cấu trúc dữ liệu Stack để viết chương trình theo yêu cầu sau:
   1. Tính giá trị của biểu thức hậu tố (ab+: nghĩa là a+b)
   2. Tính giá trị của biểu thức tiền tố (+ab: nghĩa là a+b)
2. Một bệnh nhân khi đến khám bệnh ở bệnh viện phải khai báo các thông tin sau: họ tên, tuổi, địa chỉ, tình trạng bệnh vào tờ khai bệnh cho cán bộ tiếp nhận. Sau đó cán bộ tiếp nhận sẽ gán một số thứ tự cho bệnh nhân này theo thứ tự đến của họ.
   1. Hãy khai cấu trúc dữ liệu để đưa thông tin các bệnh nhân vào hàng đợi dùng cấu trúc danh sách liên kết.
   2. Viết hàm enqueu() để đưa một bệnh nhân vào hàng đợi theo cấu trúc trên.