Chương 5

Stack & Queue

Nội dung trình bày

Stack

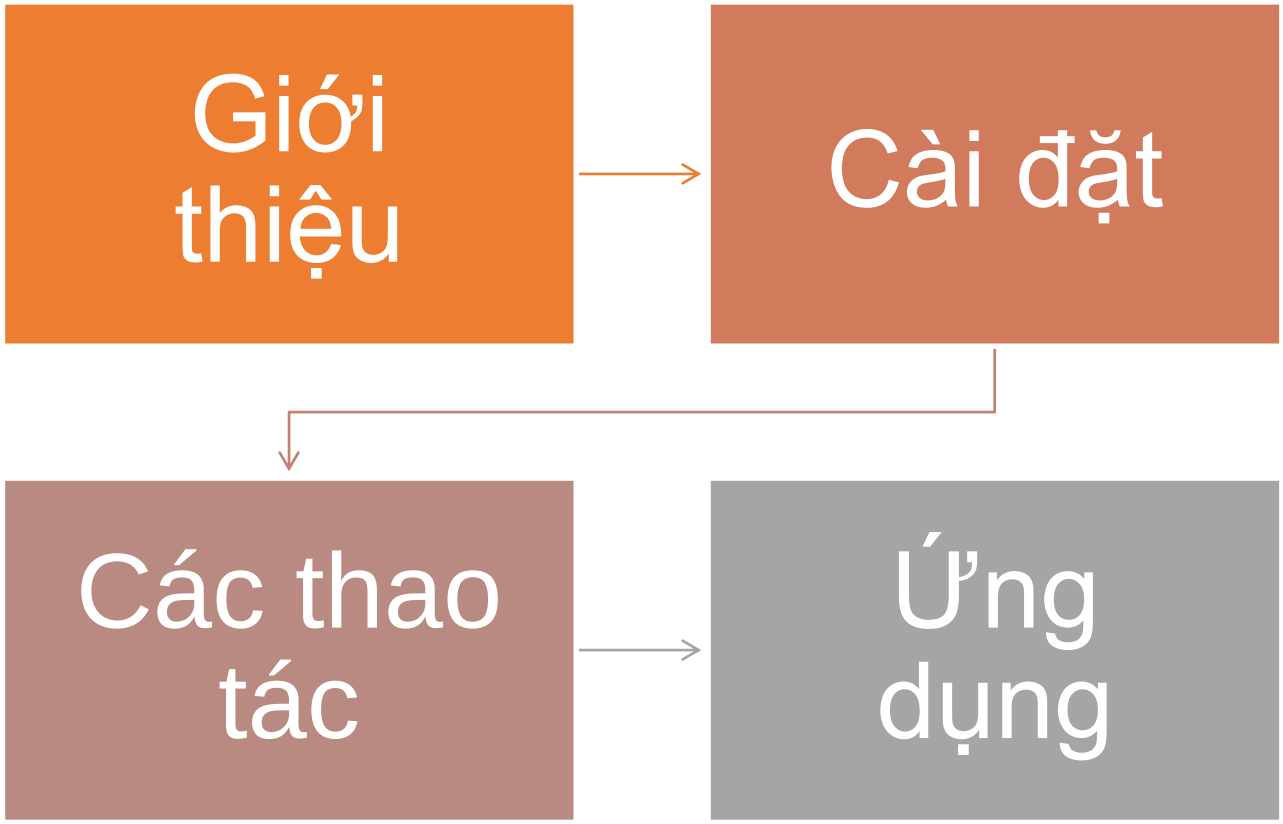
- Giới thiệu
- Cài đặt
- Các thao tác
- Ứng dụng

Queue

- Giới thiệu
- Cài đặt
- Các thao tác
- Ứng dụng

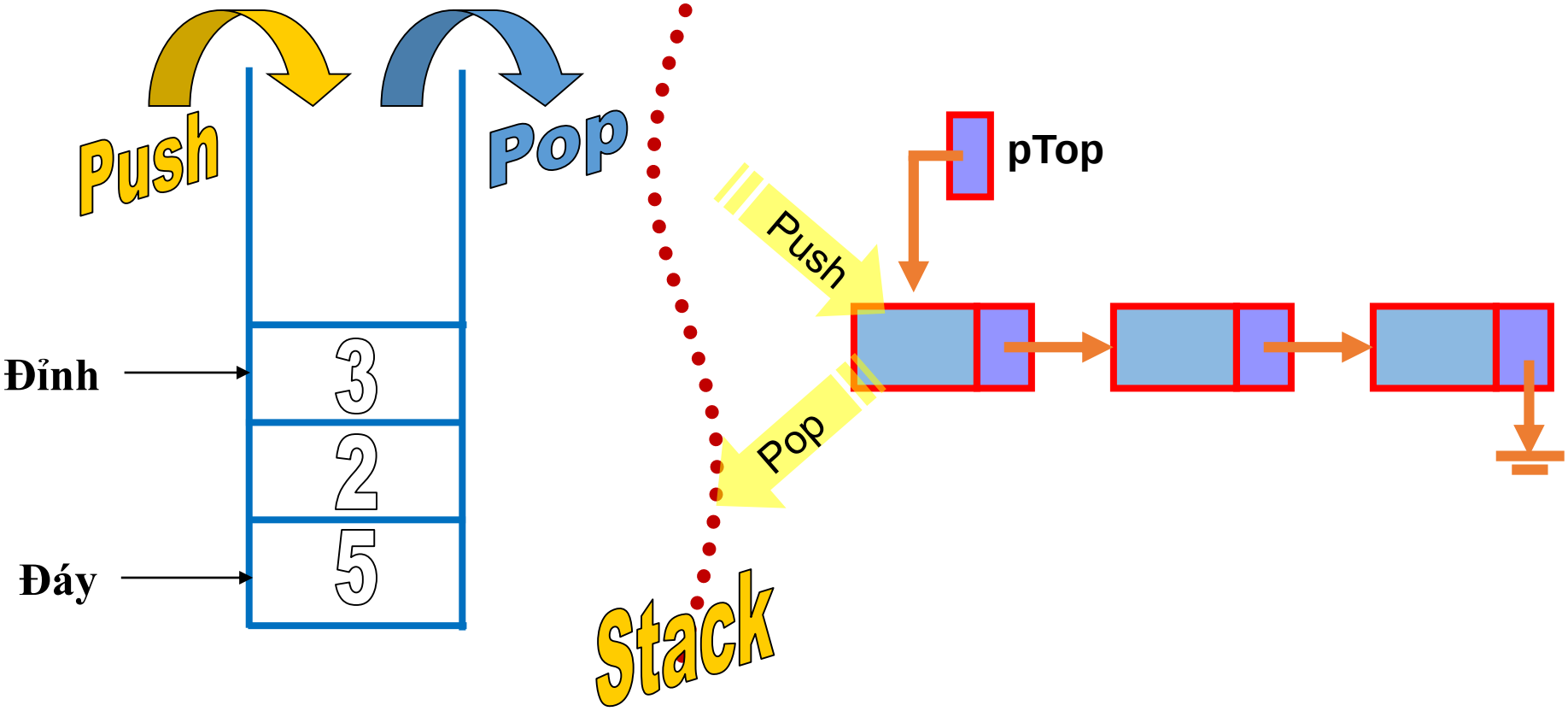
5.1.Stack

5.1 Stack



5.1.1 Giới thiệu

- Stack là một danh sách mà việc bổ sung và loại bỏ được thực hiện ở cùng 1 đầu
- LIFO: Last In First Out



5.1.1 Giới thiệu

Mảng 1 chiều

Kích thước stack khi quá thiếu, lúc quá thừa

Push / Pop hơi phức tạp

Danh sách LK

Cấp phát động!

Push/Pop khá dễ dàng

5.1.2 Cài đặt

- Khai báo stack

```
typedef   struct    node
{
        DataType        info;
        node*           next;
} NODE;
typedef   NODE *      NodePtr;

NodePtr         pTop;
pTop    =  NULL;
```

pTop quản lý stack

Khởi tạo stack

5.1.3 Các thao tác

- Các thao tác trên stack

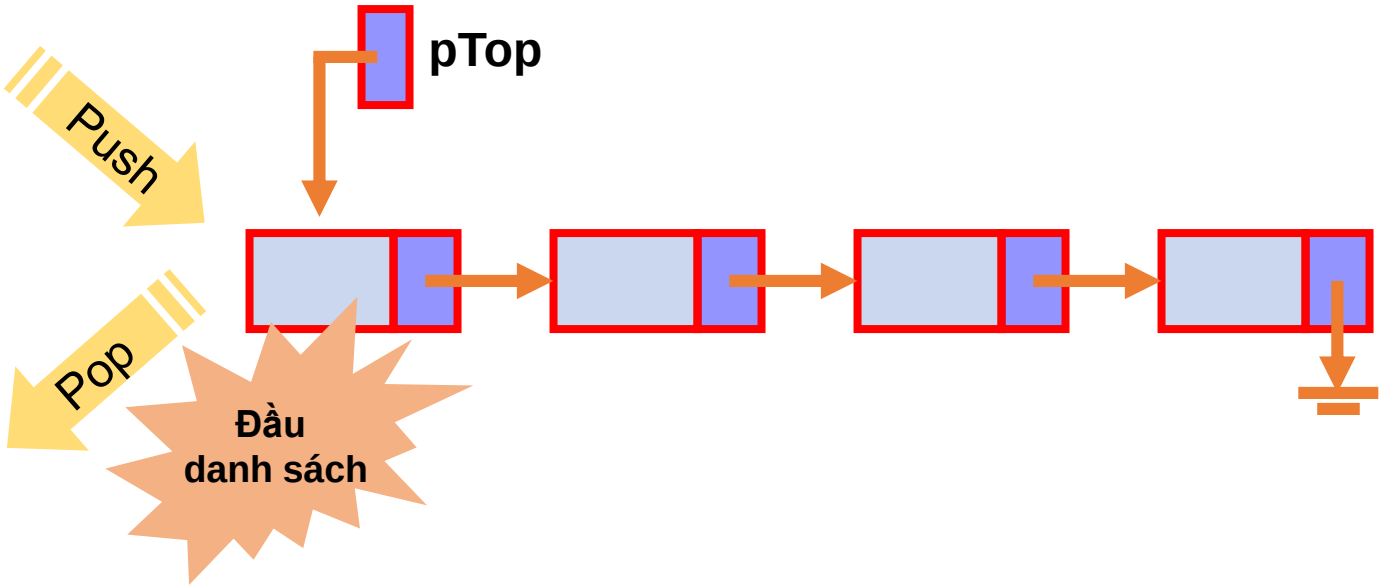
InitStack

IsEmpty

Pop

Push

Top **GetSize**



5.1.3 Các thao tác

- Khởi tạo stack

```
void Init(NodePtr &pTop)
{
      *pTop = NULL;
}
```

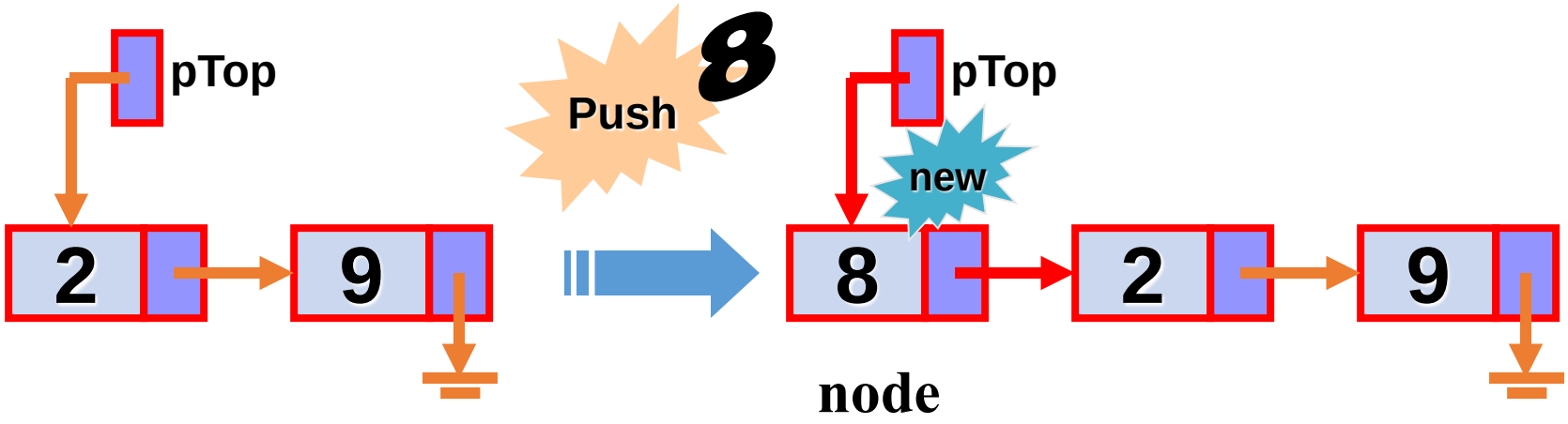
5.1.3 Các thao tác

- Kiểm tra stack có rỗng không?

```
1. int    IsEmpty(NodePtr &pTop)
2. {
3.    if (pTop == NULL)
4.          return   1;
5.    else
6.          return   0;
7. }
```

5.1.3 Các thao tác

- Push – Đưa một phần tử mới vào đỉnh stack
  - Tạo nút mới
  - Đưa vào đỉnh stack



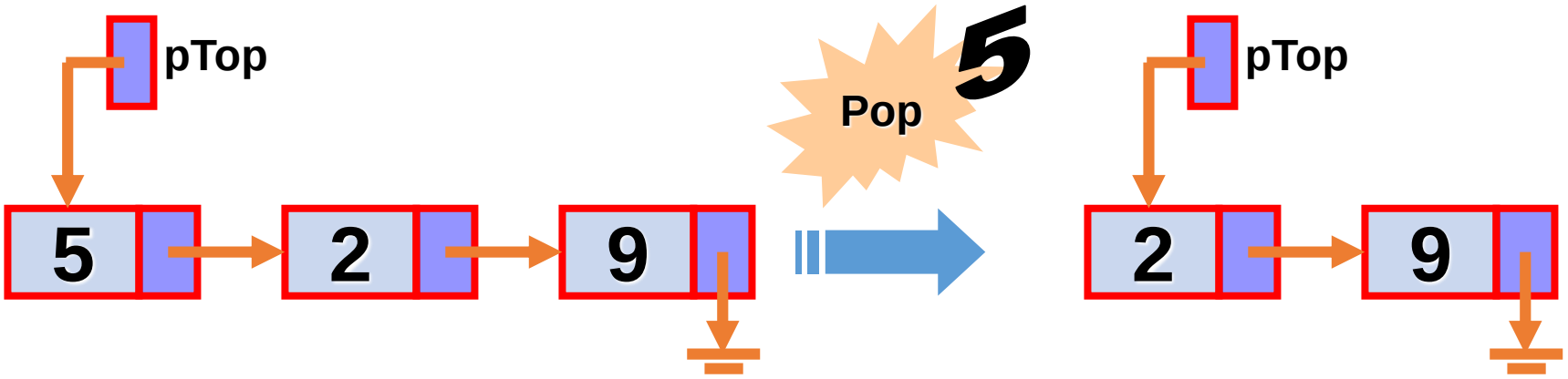
5.1.3 Các thao tác

- Push – Đưa một phần tử mới vào đỉnh stack

```
1. void Push(NodePtr &pTop, int x)
2. {  //tao nut moi
3.    NodePtr   node;
4.    node = newNODE;
5.    node->info = x;
6.    //dua vao dinh ngan xep
7.    node->next = pTop;
8.    pTop = node;
9. }
```

5.1.3 Các thao tac

- **Pop: Lấy một phần tử ra khỏi đỉnh Stack**
  - Lấy ra phần tử đầu danh sách
  - Trả về nội dung và giải phóng nút



5.1.3 Các thao tác

```
1. int Pop(NodePtr &pTop)
2. {
3.    NodePtr   p;          //trỏ nút loại bỏ
4.    int       value;      //lưu giá trị nút
5.    if (pTop == NULL) //ngăn xếp rỗng
6.    {     cout<<"Stack is empty!";
7.          return -1;
8.    }
9.    p = pTop;
10.   pTop = pTop->next;
11.   value = p->info;
12.   delete    p;
13.   return value;
14.}
```

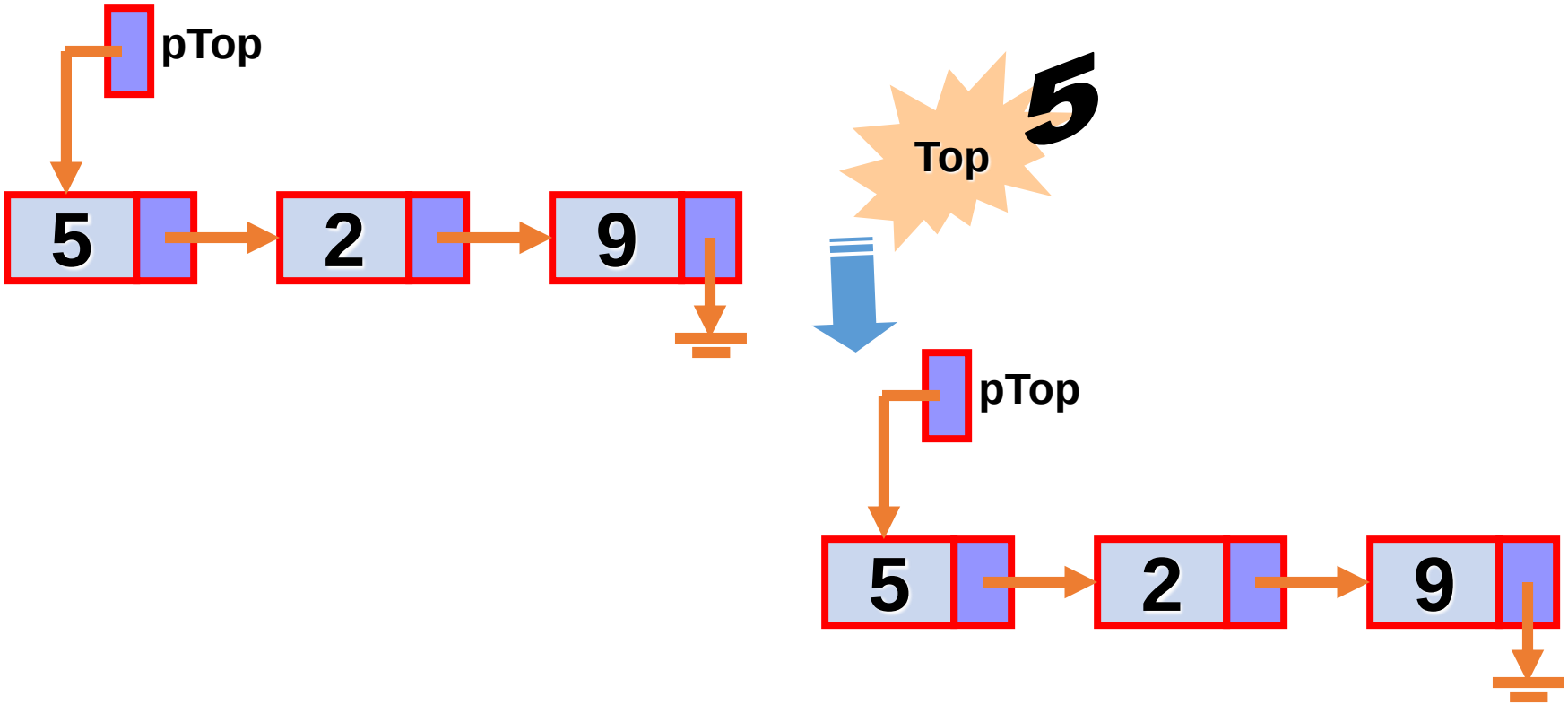
5.1.3 Các thao tác

GetSize - đếm số nút của stack

```
1. int      GetSize(NodePtr &pTop)
2. {  int   d = 0;
3.    NodePtr    p;    //con tro duyet
4.    p = pTop;        //duyet tu dau
5.    while (p!= NULL)
6.          {     d++;
7.                p = p -> next;
8.          }
9.    return     d;
10.}
```

5.1.3 Các thao tác

- Top: Lấy ra nội dung của phần tử đầu stack



5.1.3 Các thao tác

- **Top: Lấy ra nội dung của phần tử đầu stack**

```
1. int Top(NodePtr  pTop)
2. {
3.    if (pTop == NULL)
4.          return -1;
5.    else
6.          return  pTop -> info;
7. }
```

5.1.4 Ứng dụng stack

- Ứng dụng stack
  - Khử đệ quy:
    - Tháp Hanoi, QuickSort…
  - Áp dụng cho các bài toán dùng mô hình LIFO

Postfix

- Chuyển biểu thức trung tố sang hậu tố

Trung tố	Hậu tố
(6 / 2 + 3) * (7 - 4)	6 2 / 3 + 7 4 - *

Postfix

- Duyệt qua từng phần tử trong infix $\Rightarrow$ C
  - Nếu C là “(“ thì push $\Rightarrow$ stack.
  - Nếu C là “)” thì lấy tất cả phần tử trong stack cho đến khi gặp “(“. Xuất ra biểu thức hậu tố
  - Nếu C là toán tử: lấy trong stack ra tất cả toán tử có độ ưu tiên cao hơn C, xuất những phần tử này ra biểu thức hậu tố, và đưa C vào stack
  - Trường hợp C là toán hạng xuất C ra biểu thức hậu tố

Postfix

(2 * 3 + 8 / 8) * (5 – 1)

Đọc	Xử lý	Stack	Bt hậu tố
(	Đẩy vào stack	(	
2	Xuất ra bt hậu tố	(	2
*	Do '*' ưu tiên hơn '(' ở đỉnh stack nên đưa '*' vào stack	(*	2
3	Xuất ra bt hậu tố	(*	2 3
+	Do '+' ưu tiên thấp hơn '*' ở đỉnh stack nên ta lấy '*' ra. Tiếp tục so sánh '+' với '(' thì '+' ưu tiên cao hơn nên đưa vào stack	(+	2 3 *
8	Xuất ra bt hậu tố	(+	2 3 * 8
/	Do '/' có độ ưu tiên cao hơn '+' trên đỉnh stack nên đưa '/' vào stack.	(+ /	2 3 * 8
8	Xuất ra bt hậu tố	(+ /	2 3 * 8 8

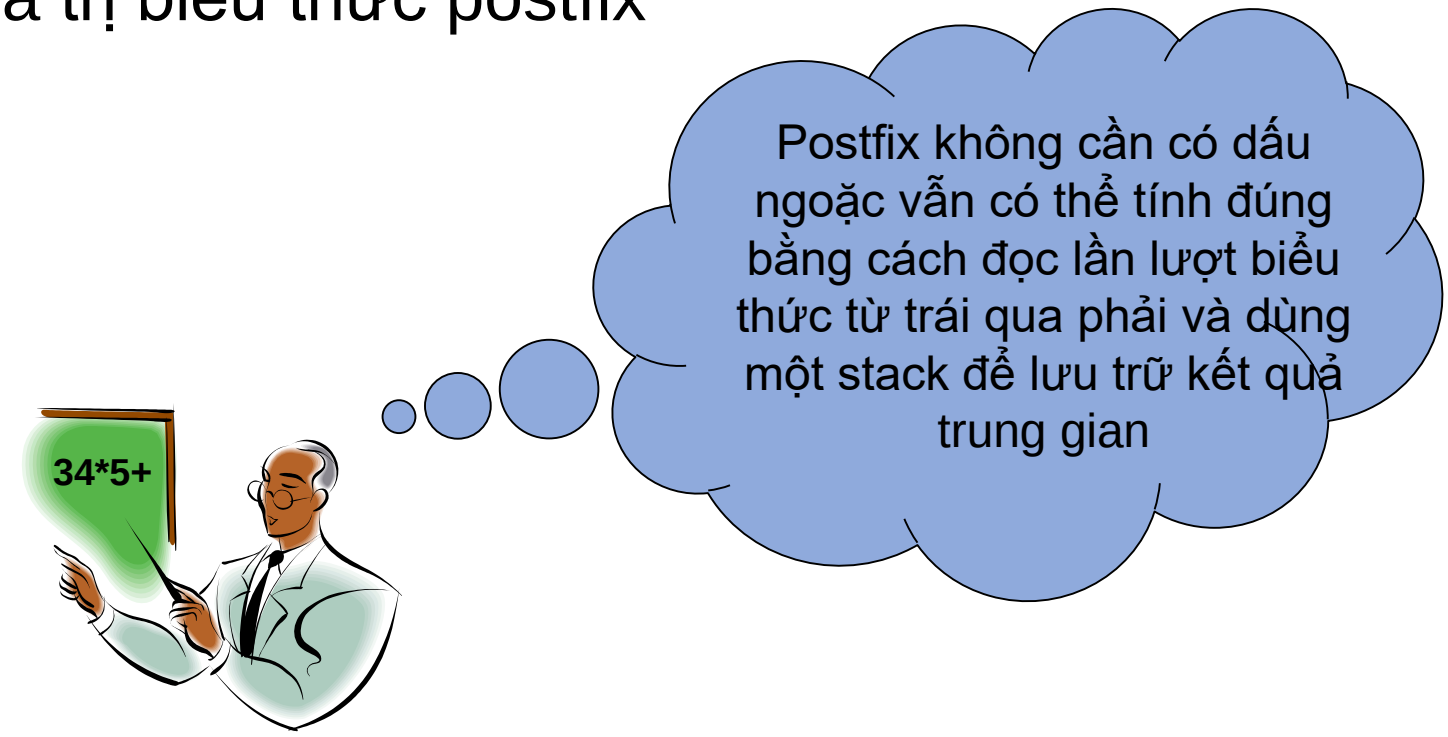
Postfix

(2 * 3 + 8 / 8) * (5 – 1)

Đọc	Xử lý	Stack	Bt hậu tố
)	Lấy trong stack ra cho đến khi gặp ngoặc (.		**2 3 * 8 8 / +**
*	Đưa vào stack	*	**2 3 * 8 8 / +**
(	Đưa vào stack	* (	**2 3 * 8 8 / +**
5	Xuất ra bt hậu tố	* (	**2 3 * 8 8 / + 5**
-	Độ ưu tiên của '-' cao hơn '(' trong đỉnh stack nên đưa '-' vào stack	* (-	**2 3 * 8 8 / + 5**
1	Xuất ra bt hậu tố	* (-	**2 3 * 8 8 / + 5 1**
)	Lấy trong stack ra cho đến khi gặp ngoặc đóng	*	**2 3 * 8 8 / + 5 1 -**
	Lấy những phần tử còn lại trong stack và hiển thị		**2 3 * 8 8 / + 5 1 - ***

Postfix - Tính giá trị biểu thức hậu tố

- Tính giá trị biểu thức postfix



Jan Lukasiewicz

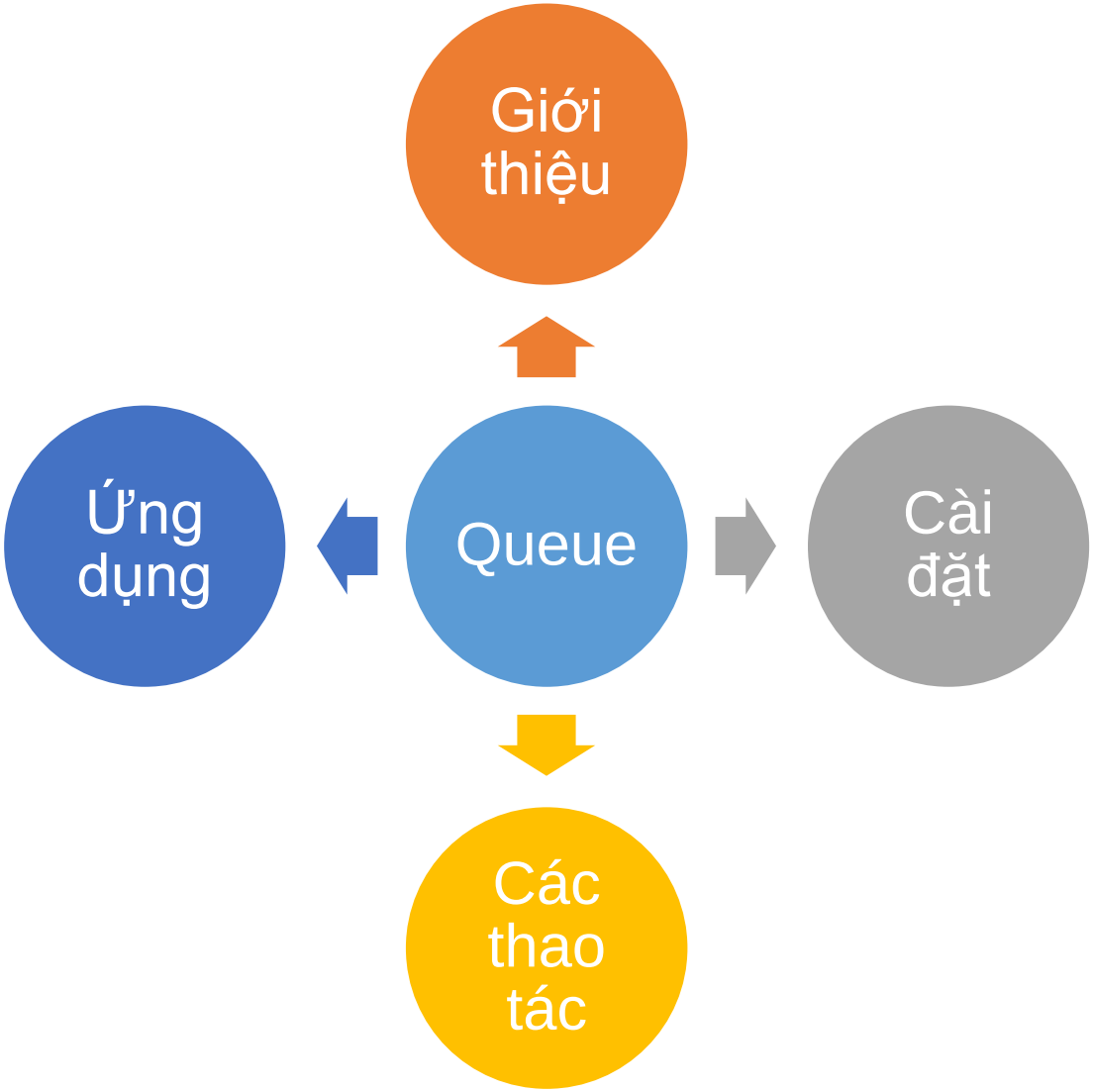
Postfix

Ý tưởng

- Khởi tạo stack = {Ø}
- Đọc lần lượt các phần tử từ trái, kiểm tra
  - Nếu toán hạng: Push $\Rightarrow$ stack
  - Nếu toán tử: lấy hai toán hạng, thực hiện phép toán, kết quả Push vào stack
- Sau khi đọc xong, trong stack còn duy nhất một phần tử $\Rightarrow$ kết quả!

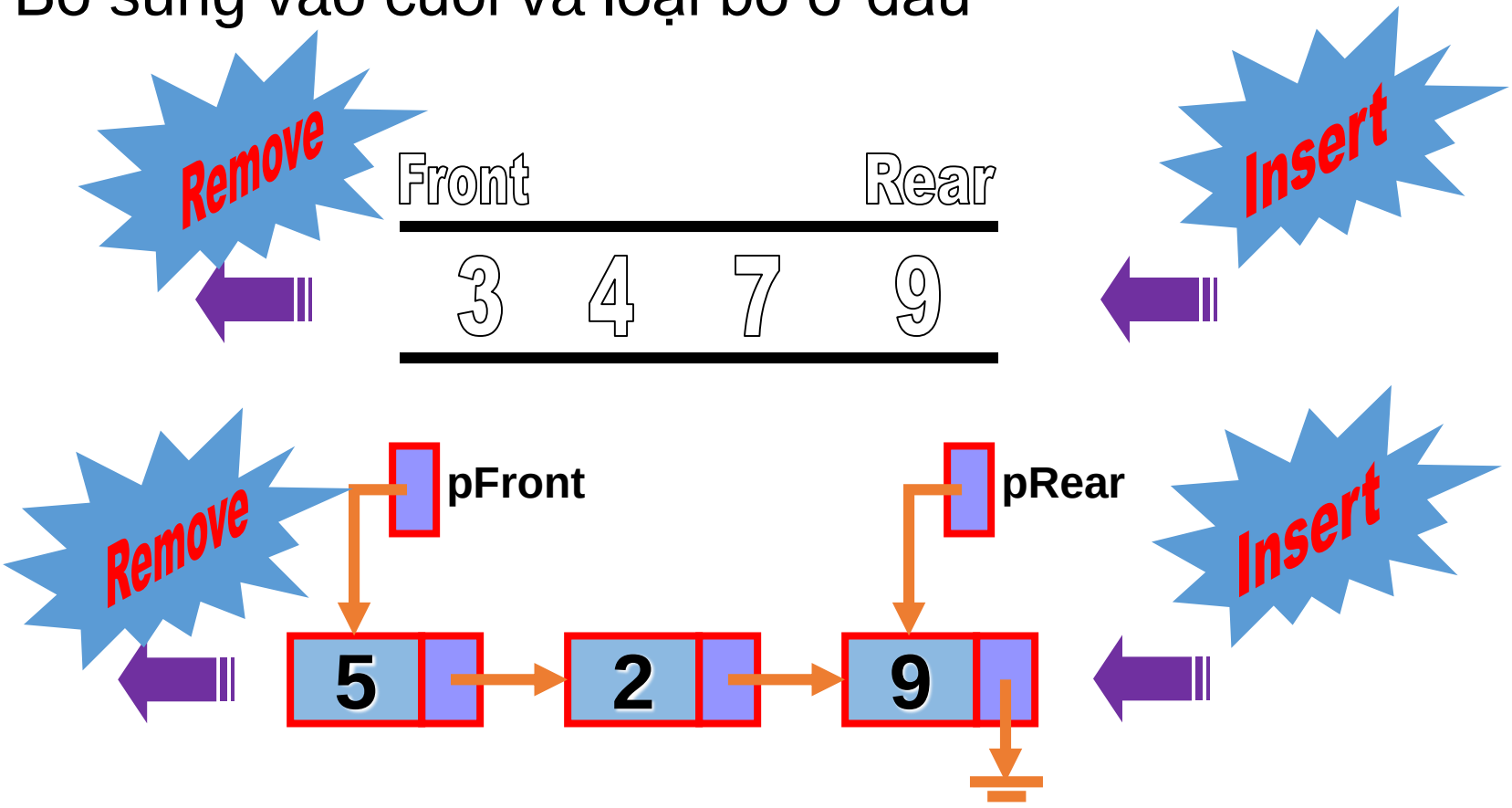
5.2.Queue

5.2 Queue



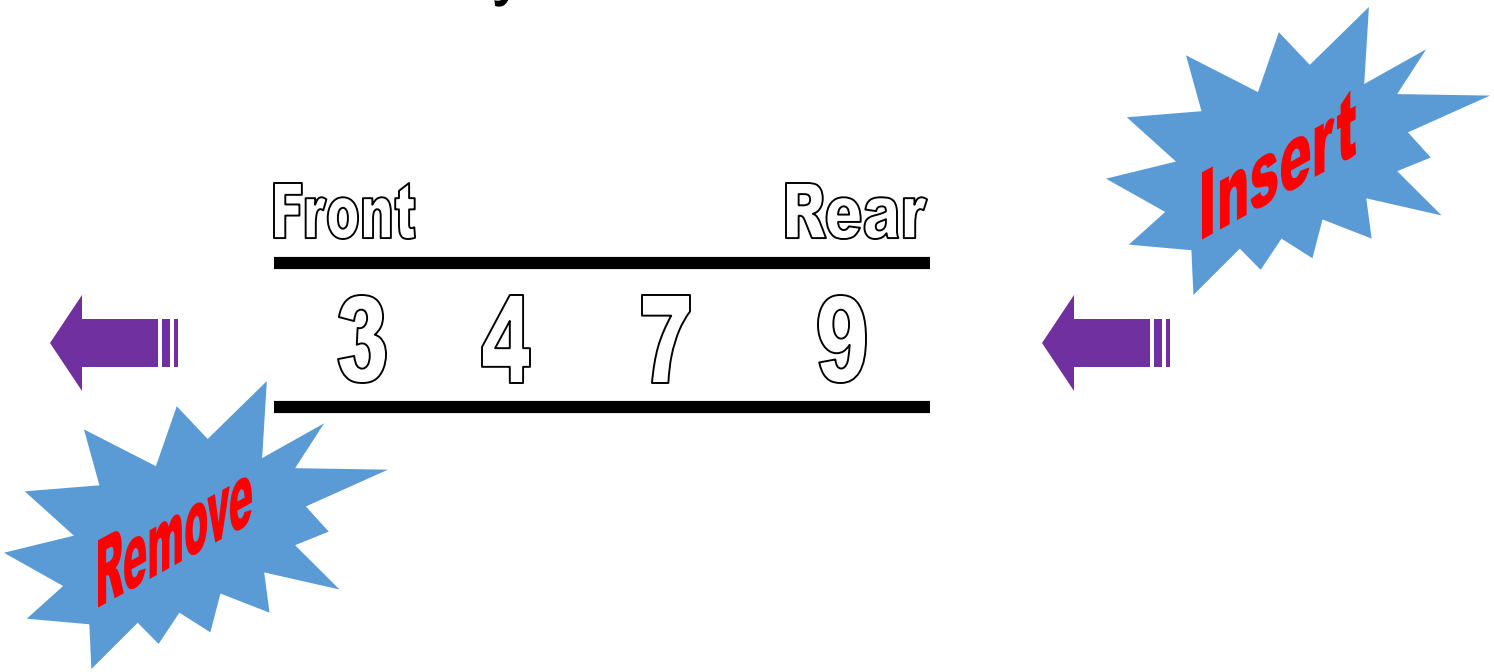
5.2.1 Giới thiệu

- Queue là một danh sách mà việc bổ sung và loại bỏ thực hiện ở hai đầu khác nhau
- Bổ sung vào cuối và loại bỏ ở đầu



5.2.1 Giới thiệu

- FIFO: First In First Out
- Thêm vào cuối và lấy ra ở đầu



5.2.1 Giới thiệu

- Queue dùng DSLK
  - Con trỏ pFront trỏ đầu danh sách
  - Con trỏ pRear trỏ đến cuối danh sách
  - Thao tác Remove diễn ra ở pFront
  - Tháo tác Insert diễn ra ở pRear
  - Thao tác thêm xoá dễ dàng ở hai đầu

5.2.2 Cài đặt

- Tạo cấu trúc Node cho Queue

```
typedef struct node
{
      DataType        info;
      struct  node*   next;
}NODE;
typedef  NODE*  NodePtr;
struct Queue
{
      NodePtr   pFront;
      NodePtr   pRear;
}
```

5.2.3 Các thao tác

- Các thao tác trên Queue
  - Init : Khởi tạo hàng đợi
  - IsEmpty : Kiểm tra hàng đợi rỗng
  - **Insert** : Bổ sung 1 ptử vào hàng đợi
  - **Remove**: Loại bỏ 1 ptử khỏi hàng đợi
  - QueueFront : Lấy giá trị của nút pFront
  - QueueRear : Lấy giá trị của nút pRear
  - QueueSize : Đếm số nút của hàng đợi
  - Clear : Xóa toàn bộ hàng đợi

5.2.3 Các thao tác

- Init – Khởi tạo Queue

```
void Init(Queue &q)
{
      *q.pFront = NULL;
      *q.pRear  = NULL;
}
```

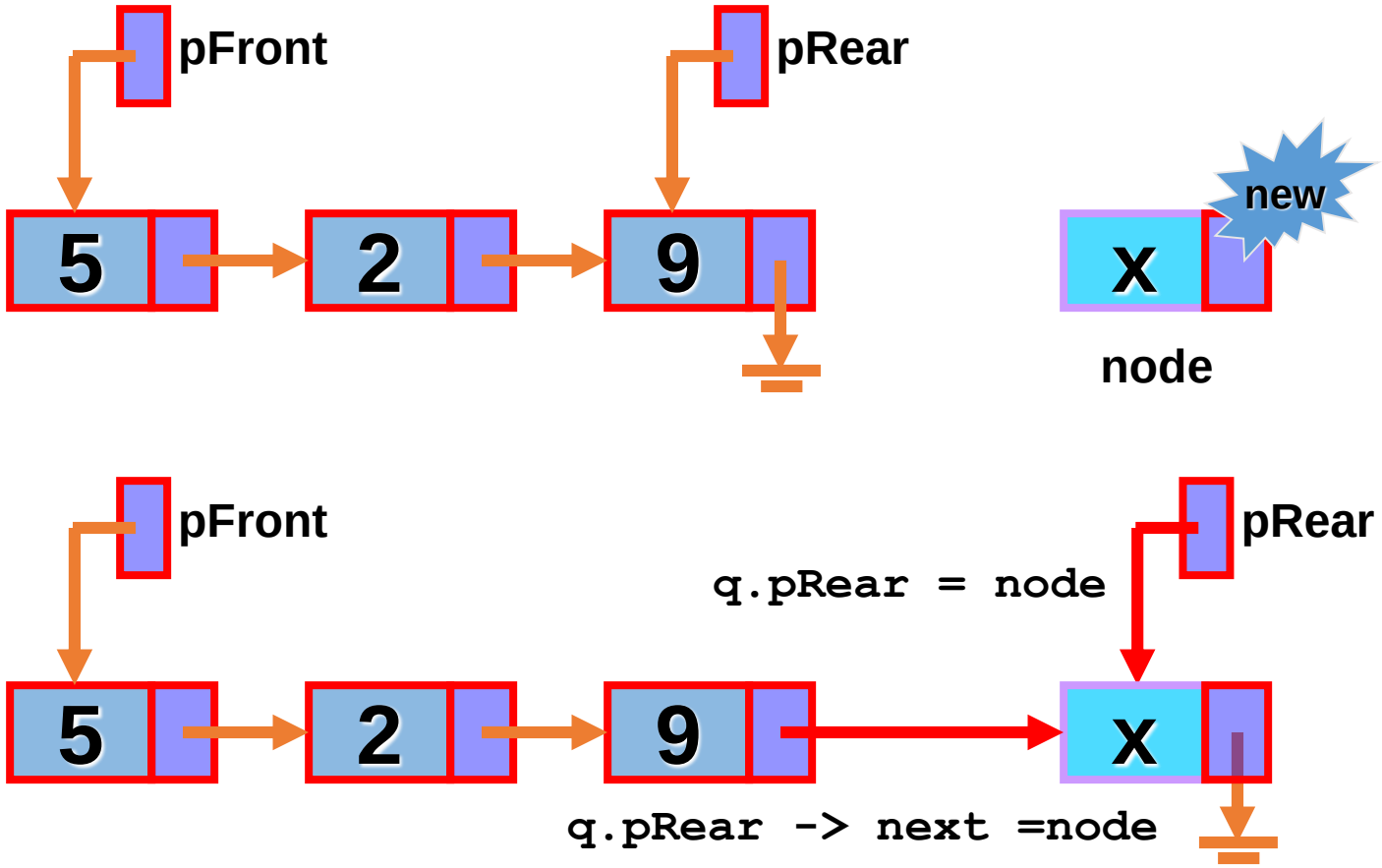
5.2.3 Các thao tác

- Init – Khởi tạo Queue

```
int  IsEmpty(Queue &q)
{
     if ( q.pFront == NULL)
          return   1;
     else
          return   0;
}
```

5.2.3 Các thao tác

- Insert – Bổ sung phần tử vào hàng đợi

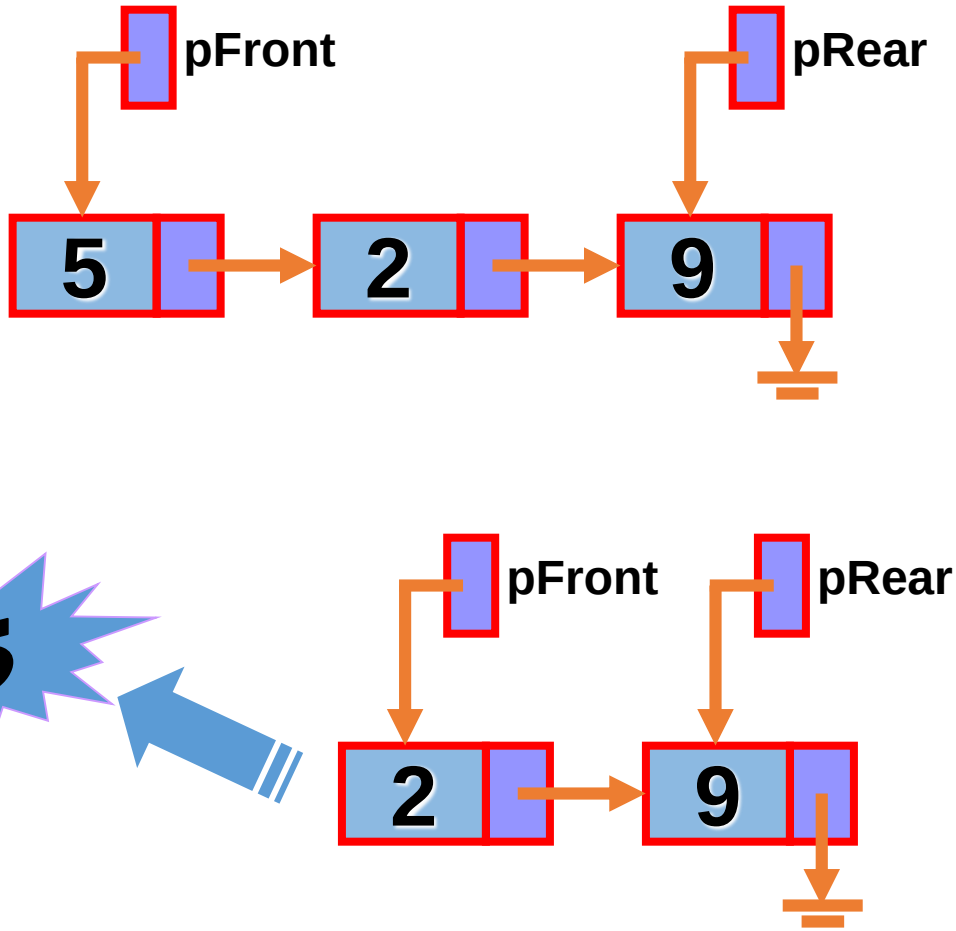


5.2.3 Các thao tác

```
1. void Insert(Queue &q, int x)
2. {  NodePtr    node;
3.    node = new NODE;
4.    node -> info = x;
5.    node -> next = NULL;
6.    if (q.pRear == NULL)
7.    {     q.pRear = node;
8.          q.pFront = node;
9.    }
10.   else
11.   {     q.pRear->next = node;
12.         q.pRear = node;
13.   }
14.}
```

5.2.3 Các thao tác

- Remove - Loại bỏ một nút khỏi hàng đợi



5.2.3 Các thao tác

```
1. int Remove(Queue  &q)
2. {  NodePtr     p;
3.    int         value;
4.    if (q.pFront == NULL)          //hang doi rong
5.        return -1;
6.    if (q.pFront == q.pRear)       //chi co 1 nut
7.    {     p = q.pFront;
8.          q.pFront = q.pRear = NULL;
9.    }
10.   else                           //co nhieu nut
11.   {     p =q.pFront;
12.         q.pFront = p->next;            }
13.   value = p->info;
14.   delete  p;
15.   return value;
16.}
```

5.2.3 Các thao tác

- QueueSize - Đếm số nút của hàng đợi

```
1. int      QueueSize(Queue    &q)
2. {  int   d = 0;              //bien dem d
3.    NodePtr    p;             //con tro de duyet
4.    p = q.pFront;             //duyet tu dau
5.    while (p != NULL)
6.          {     d++;
7.                p = p -> next;
8.          }
9.    return     d;
10.}
```

5.3.2 Các thao tác

- QueueFront – Lấy ra giá trị của nút trỏ bởi pFront

```
1. int QueueFront(Queue  &q)
2. {
3.    if (q.pFront == NULL)
4.          return  -1;
5.    else
6.          return  q.pFront->info;
7. }
```

5.3.2 Các thao tác

- QueueRear – Lấy ra giá trị của nút trỏ bởi pRear

```
1. int QueueRear(Queue  &q)
2. {
3.    if (q.pRear == NULL)
4.          return  -1;
5.    else
6.          return  q.pRear->info;
7. }
```

5.2.3 Các thao tác

- ClearQueue – Xóa hàng đợi

```
1. void ClearQueue(Queue  &q)
2. {  NodePtr   p;
3.    while (q.pFront != NULL)
4.    {     p =   q.pFront;
5.          q.pFront = p -> next;
6.          delete  p;
7.    }
8. }
```

5.2.4 Ứng dụng

- Ứng dụng Queue
  - Trong bài toán hàng đợi "Vào trước ra trước" FIFO:
    - Hệ thống print server
    - Cơ chế thông điệp, bộ đệm, hàng đợi xử lý sự kiện…
    - Các ứng dụng đặt vé tàu lửa, máy bay…
    - Các hệ thống rút tiền…

Câu hỏi củng cố bài

1. Thao tác nào dưới đây thực hiện trên ngăn xếp (stack):

A. Thêm và loại bỏ phần tử luôn thực hiện ởđầu danh sách

B. Thêm phần tử vào vị trí bất kỳ

C. Loại b ỏphần tử tại vị trí bất kỳ

D. Thêm phần tử luôn thực hiện tại vị trí cuối danh sách

Câu hỏi củng cố bài

2. Thao tác nào dưới đây thực hiện trên hàng đợi (queue):

A. Thêm và loại bỏ phần tử luôn thực hiện tại lối trước (đầu danh sách)

B. Thêm phần tử vào vị trí bất kỳ

C. Loại bỏ phần tử tại vị trí bất kỳ

D. Thêm phần tử luôn thực hiện tại lối sau (cuối danh sách)

Câu hỏi củng cố bài

3. Hoạt động của ngăn xếp là _____ để thêm một phần vào đỉnh ngăn xếp và _____ đ ểloại b ỏphần t ở ửđỉnh ngăn xếp.

A. malloc (), free ()

B. pop (), push ()

C. push (), pop ()

D. new (), delete ()

Câu hỏi củng cố bài

4. Cơ chế nào dưới đây được cài đặt cho hàng đợi

   A. Cơ chế vào trước ra trước (FIFO)

   B. Cơ chế Round Robin

   C. Cơ chế tuần tự

   D. Cơ chế vào sau ra trước (LIFO)

Câu hỏi củng cố bài

5. Cho một hàng đợi lưu trữ dưới dạng một danh sách liên kết, có một con trỏ trỏ vào lối trước và một con trỏ trỏ vào lối sau của danh sách. Các biến con trỏ sẽ thay đổi thế nào khi bổ sung một phần tử vào một hàng đợi rỗng?

A. Không thay đổi

B. Cả hai thay đổi.

C. Chỉ có con trỏ lối trước thay đổi.

D. Chỉ có con trỏ lối sau thay đổi.

Câu hỏi củng cố bài

6. Cho một hàng đợi lưu trữ dưới dạng một danh sách liên kết, có một con trỏ trỏ vào lối trước và một con trỏ trỏ vào lối sau của danh sách. Các biến con trỏ sẽ thay đổi thế nào khi bổ sung một phần tử vào một hàng đợi không rỗng?

A. Không thay đổi

B. Cả hai thay đổi.

C. Chỉ có con trỏ lối trước thay đổi.

D. Chỉ có con trỏ lối sau thay đổi.

Bài tập

1. Cho một danh sách nối kép có nút đầu được trỏ bởi pHead. Trường info của các nút chứa giá trị nguyên. Viết giải thuật bổ sung và loại bỏ để danh sách hoạt động như một Queue.
2. Viết chương trình C++ chứa tất cả các thao tác trên stack.
3. Viết chương trình C++ chứa tất cả các thao tác trên queue.

Tài liệu tham khảo

[1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013

[2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.

[3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018

[4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .

[5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014