HCMUTE

CHƯƠNG 5. STACK & QUEUE

HCMUTE

Nội dung

Nội dung 01 — STACK

Nội dung 02 — QUEUE

Nội dung 03 — THƯ VIỆN CLASS STACK, QUEUE

HCMUTE

Nội dung

Nội dung 01 **STACK**

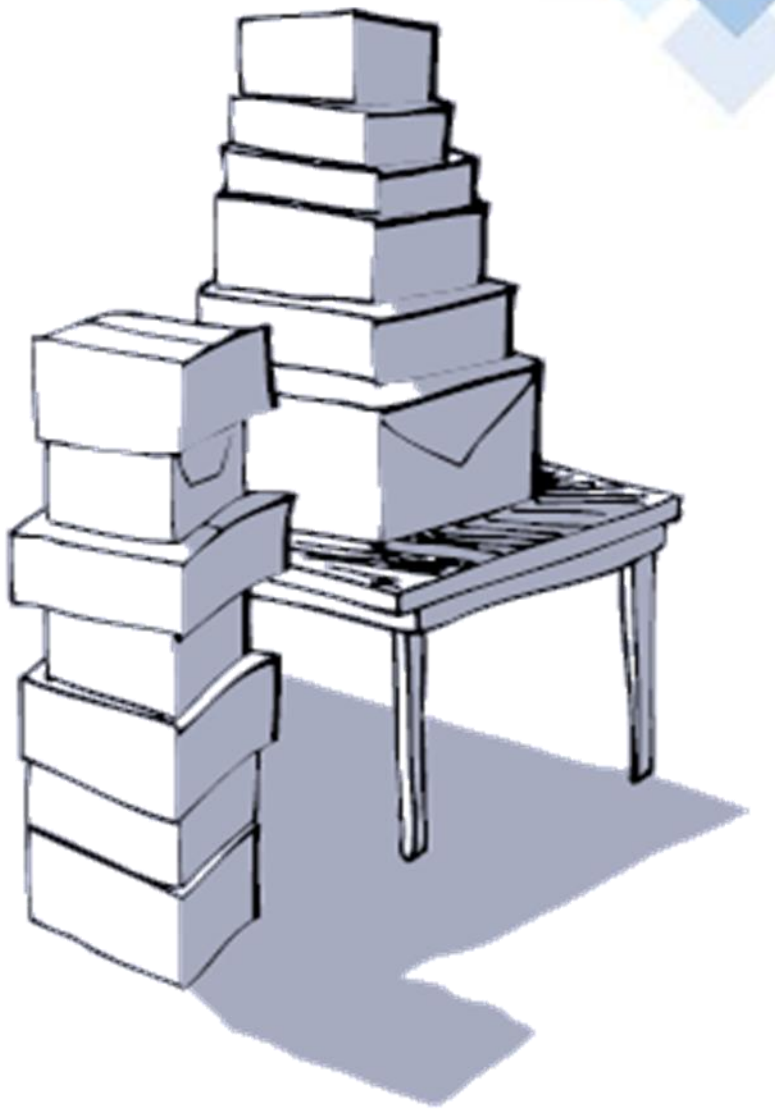
Nội dung 02 **QUEUE**

Nội dung 03 **THƯ VIỆN CLASS STACK, QUEUE**

HCMUTE

Nội Dung

1. Khái niệm stack
2. Stack ADT
3. Cài đặt stack
4. Ứng dụng của stack



1. Khái niệm Ngăn xếp (STACK)

- Một ngăn xếp (stack) là một cấu trúc dữ liệu đơn giản được sử dụng để lưu trữ dữ liệu. *Stack là một danh sách có thứ tự trong đó việc thêm phần tử vào hay lấy ra đều diễn ra tại một đầu, gọi là đỉnh (_top) của Stack. Phần tử cuối cùng đưa vào Stack chính là phần tử đầu tiên sẽ được lấy ra. Vì thế Stack được gọi là một danh sách làm việc theo cơ chế LIFO (Last in First out – vào sau ra trước) hoặc FILO (First in Last out – Vào trước ra sau).*

HCMUTE

2. Stack ADT

Khi xây dựng cấu trúc dữ liệu Stack, chúng ta sẽ xây dựng luôn bộ thao tác trên Stack, tạo thành Stack ADT (Abstract Data Type).

❖Thuộc tính hỗ trợ

- Count: Cho biết số phần tử chứa trong Stack.

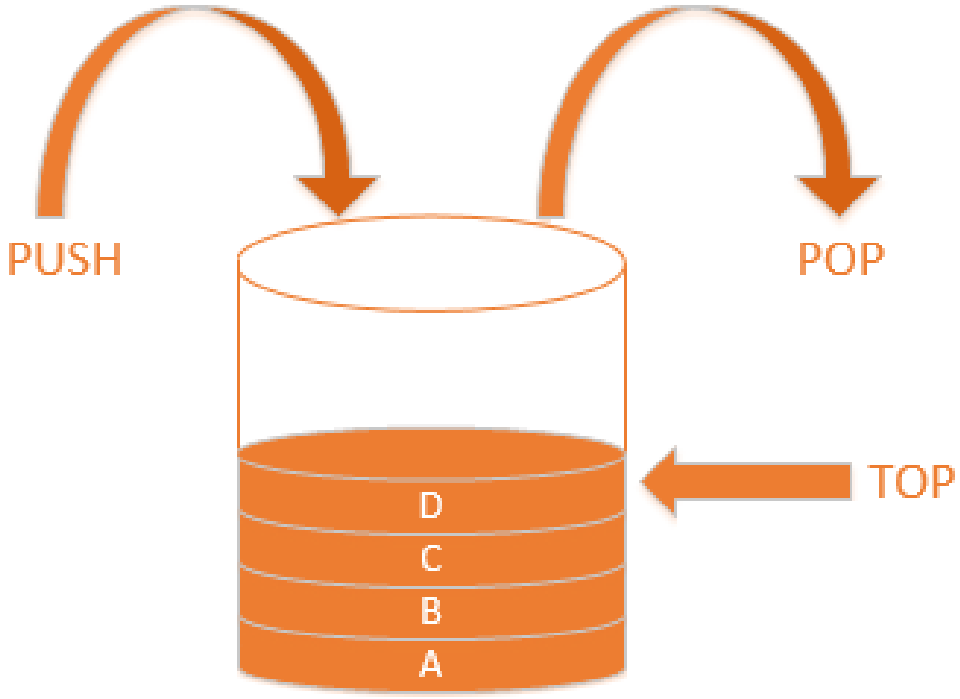
❖Các thao tác chính

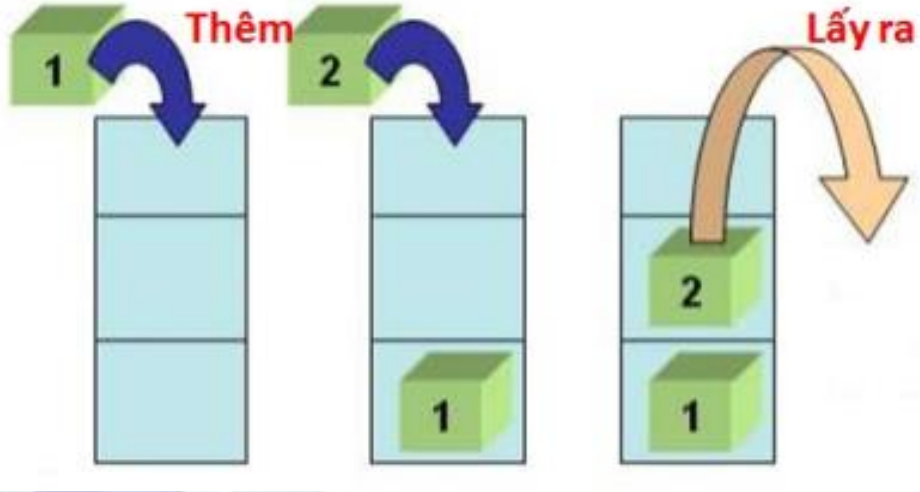
- void Push(int value): Chèn data vào Stack.
- int Pop(): Xóa và trả về phần tử trên đỉnh Stack.

❖Các thao tác hỗ trợ

- int Peek(): trả về phần tử trên đỉnh Stack mà không lấy ra khỏi Stack.
- int IsEmpty(): Kiểm tra Stack có rỗng hay không
- int IsFull(): Kiểm tra Stack có đầy hay không
- void Display(): Hiển thị Stack
- void Clear(): Xóa Stack

Push - Pop





HCMUTE

3. Cài đặt Stack

Mảng 1 chiều

Danh sách LK

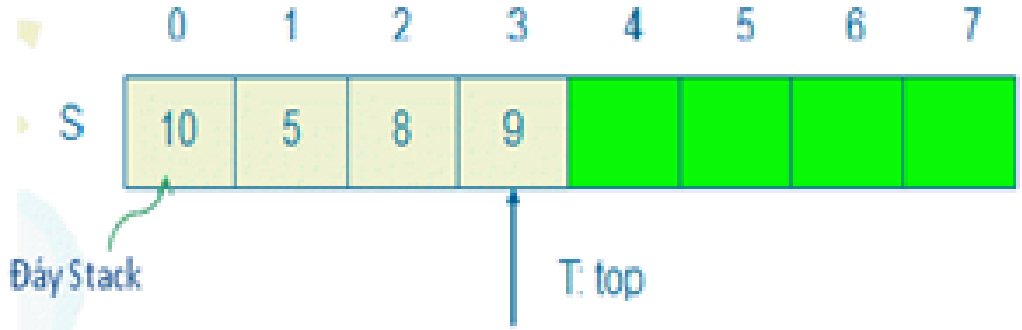
HCMUTE

3. Cài đặt Stack

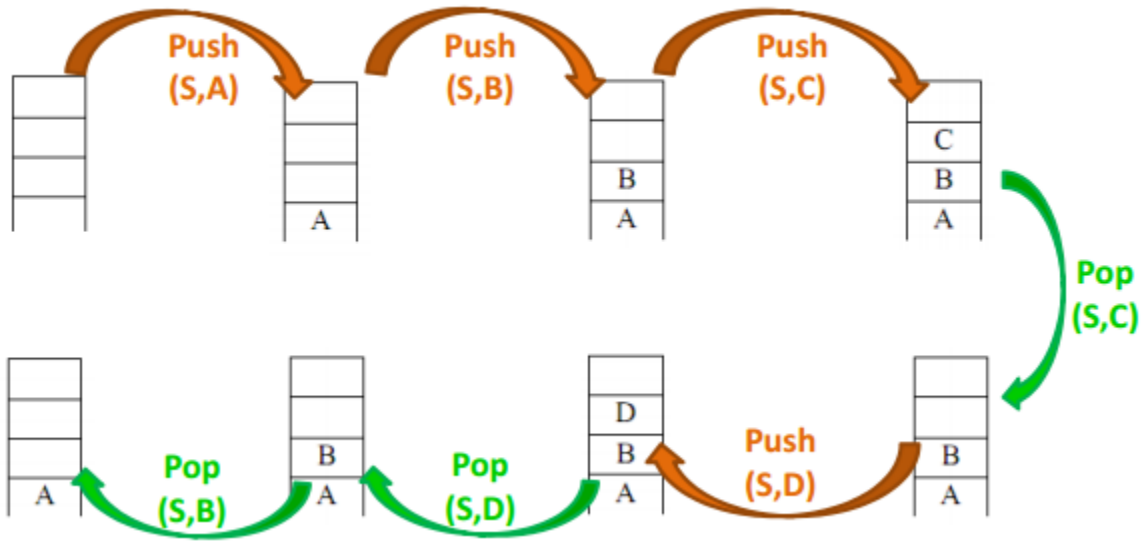
- Cài đặt Stack bằng mảng
- Cài đặt Stack bằng Linked List

Cài đặt Stack bằng mảng

Hãy xác định các thành phần lưu trữ của một Stack? Từ đó suy ra cấu trúc dữ liệu của Stack?



Ví dụ Stack lưu trữ kí tự



Cấu trúc dữ liệu Stack

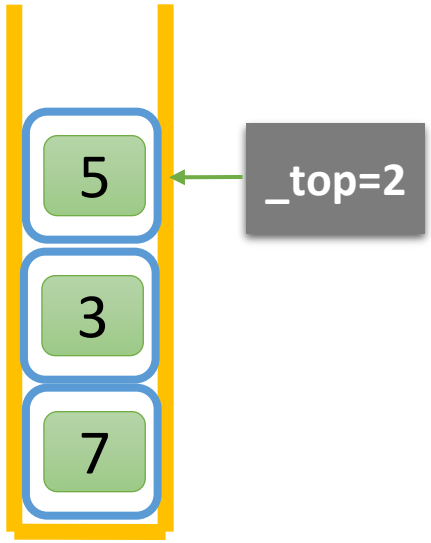
❖ **Để tạo ra một Stack thì chúng ta cần có:**

- Một mảng 1 chiều với kích thước tối đa (Capacity) là MAXSIZE(ví dụ: 100)
- Stack có thể chứa tối đa MAXSIZE phần tử đánh số từ 0 đến MAXSIZE - 1
- Phần tử nằm ở đỉnh Stack sẽ có chỉ số là **_top**

Khai báo cấu trúc dữ liệu

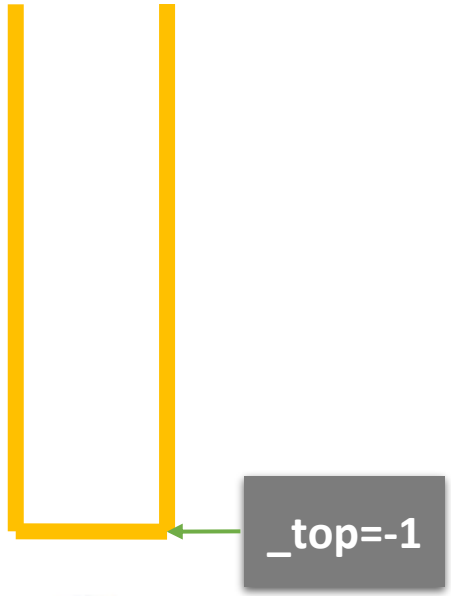
```
struct ArrayStack
{
    private int[] _array;     //mảng chứa các phần tử của stack
    private int _capacity;    //số phần tử tối đa của stack
    private int _top;         //chỉ số của phần tử ở đỉnh stack

    //thuộc tính Count cho biết số phần tử có trong Stack
    public int Count
    {
        get
        {
            return (_top + 1);
        }
    }
    ...
}
```



Các thao tác

- Khởi tạo Stack




```
/*
* Hàm tạo và khởi tạo ban đầu cho 1 stack.
*/
public ArrayStack(int maxSize)
{
    if (maxSize < 0)
        throw new Exception("MaxSize < 0");
    _capacity = maxSize;
    _array = new int[_capacity];
    _top = -1;
}
```

Hàm xử lý ngoại lệ

IsEmpty() : Kiểm tra xem Stack đã rỗng chưa, vì khi Stack đã rỗng, việc gọi đến hàm Pop() sẽ phát sinh ra lỗi

```
/*
* Kiểm tra stack có rỗng hay không
* return: true nếu stack rỗng, false
nếu stack khác rỗng
*/
public bool IsEmpty()
{
    return (_top == -1);
}
```

Hàm xử lý ngoại lệ

IsFull(): Kiểm tra xem Stack có đầy chưa, vì khi Stack đầy, việc gọi đến hàm Push() sẽ phát sinh ra lỗi

```
/*
* Kiểm tra Stack có đầy hay không
* return: true nếu stack đầy, flase nếu stack chưa đầy
*/
public bool IsFull()
{
    return (_top == _capacity - 1);
}
```

Push – Đẩy một phần tử vào đỉnh stack

```
/*
* Đẩy một phần tử vào đỉnh stack
* Tham số value: giá trị của phần tử cần thêm,
nếu stack đầy thì thông báo ngoại lệ
*/
public void Push(int value)
{
    if (_top == _capacity - 1)
        throw new Exception("Stack day");
    _array[++_top] = value;
}
```

Pop – Lấy một phần tử trên đỉnh ra khỏi Stack

```
/*
* Lấy một phần tử ra khỏi đỉnh stack
* return: giá trị phần tử vừa lấy ra khỏi stack,
nếu stack rỗng thì thông báo ngoại lệ
*/
public int Pop()
{
    if (_top == -1)
        throw new Exception("Stack rong");
    int temp = _array[_top];
    Array.Clear(_array, _top, 1);
    _top--;
    return temp;
}
```

Peek – Xem giá trị phần tử trên đỉnh Stack

```
/*
* Xem giá trị phần tử trên đỉnh stack
* return: giá trị phần tử trên đỉnh stack, nếu
stack rỗng thì thông báo ngoại lệ
*/
public int Peek()
{
    if (_top == -1)
        throw new Exception("Stack rong");
    return _array[_top];
}
```

Display – Hiển thị Stack

```
/*
* Hiển thị toàn bộ Stack
*/
public void Display()
{
    if (IsEmpty())
        Console.WriteLine("Stack rong!");
    else
    {
        for (int i = _top; i >= 0; i--)
            Console.WriteLine(_array[i]);
    }
}
```

Hủy Stack

```
/*
* Hủy bỏ Stack. Kết quả: stack rỗng
*/
public void Clear()
{
    Array.Clear(_array, 0, Count);
    _top = -1;
}
```

HCMUTE

Nhận xét

❖**Hạn chế của phương pháp này:** Kích thướctối đa của stack phải được xác định trước và không thể thay đổi. Nếu cấp dư gây lãng phí, nếu cấp ít thì sẽ có thể bị thiếu hụt.

HCMUTE

3. Cài đặt Stack

- Cài đặt Stack bằng mảng
- Cài đặt Stack bằng Linked List

HCMUTE

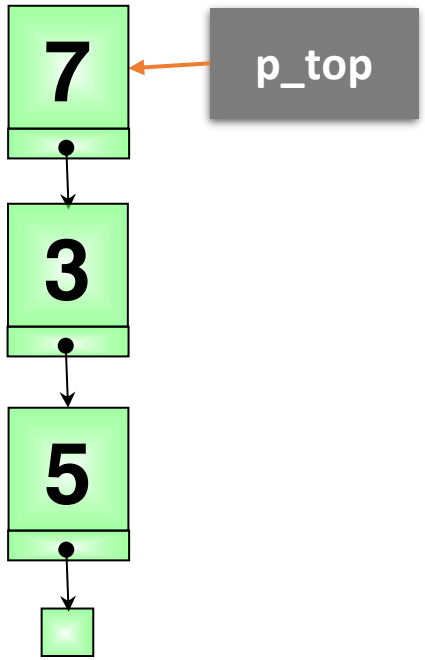
Cài đặt Stack bằng Linked List

- Việc thêm phần tử vào Stack (Push) như là việc thêm một phần tử tại đầu danh sách (AddFirst).
- Việc lấy ra một phần tử (Pop) như là việc xóa một phần tử ở đầu danh sách (RemoveFirst).

Cấu trúc dữ liệu Stack

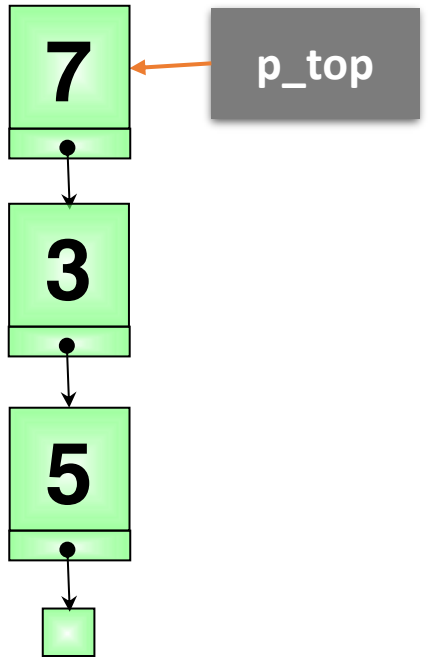
```
//struct chứa một phần tử của stack
struct Node
{
    internal int Data; //giá trị của phần tử
    internal Node Next;//con trỏ Next trỏ tới phần tử kế tiếp

    //hàm khởi tạo
    public Node(int value)
    {
        Data = value;
        Next = null;
    }
}
```



Cấu trúc dữ liệu Stack

```
//struct Stack
struct ListStack
{
    private Node _top;//con trỏ _top trỏ tới đỉnh Stack
    //thuộc tính Count cho biết số phần tử có trong Stack
    private int _size;
    public int Count
    {
        get
        {
            return _size;
        }
    }
    ...
}
```



Khởi tạo Stack

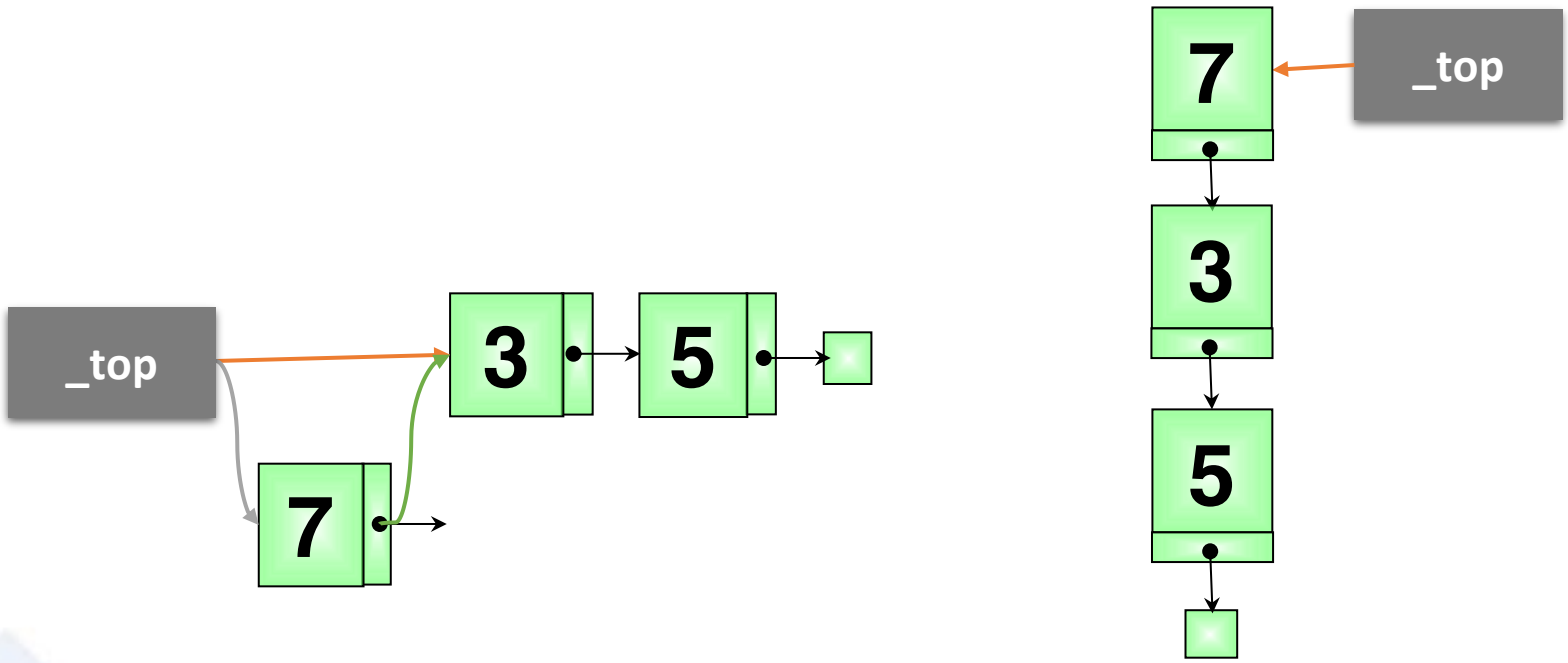
```
/*
* Khởi tạo Stack
*/
public ListStack()
{
    _top = null;
    _size = 0;
}
```

HCMUTE

Kiểm tra xem Stack có rỗng hay không

```
/*
* Kiểm tra stack rỗng hay không
* Return: true nếu stack rỗng, false nếu stack khác rỗng
*/
public bool IsEmpty()
{
    return (_top == null);
}
```

Đẩy một phần tử vào đỉnh stack

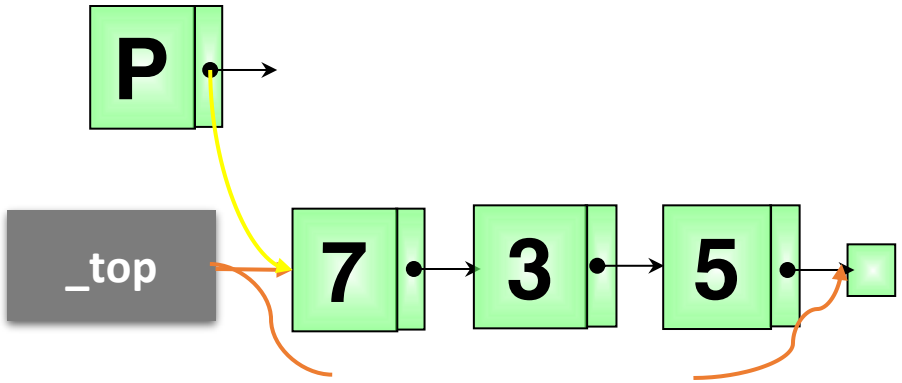


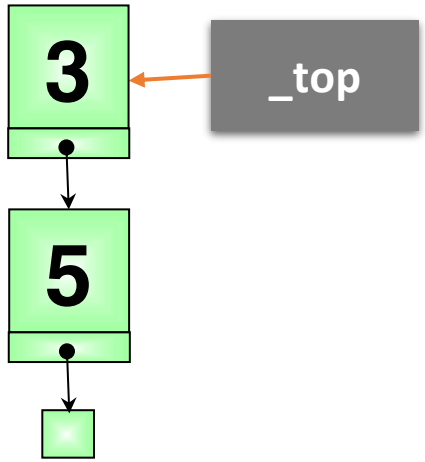
Đẩy một phần tử vào đỉnh stack

```
/*
* Hàm đẩy một phần tử vào đỉnh stack
*
* Tham số value: dữ liệu của phần tử cần thêm
* Nếu tạo node mới thất bại thì thông báo ngoại lệ
*/
```

```
public void Push(int value)
{
    //tạo node mới
    Node pNew = new Node(value);
    if (pNew == null)
        throw new Exception("Khong the them vao stack");
    //nếu stack rỗng
    if (_top == null)
        _top = pNew;
    else
    {
        pNew.Next = _top;
        _top = pNew;
    }
    _size++;
}
```

Lấy phần tử trên đỉnh stack





Lấy phần tử trên đỉnh stack

```
/*
* Lấy một phần tử ra khỏi đỉnh stack
* Return: Giá trị phần tử vừa lấy ra, nếu stack rỗng thì thông báo ngoại lệ
*/
```

```
public int Pop()
{
    if (_top == null)
        throw new Exception("Stack rong");
    //lấy giá trị và tách phần tử đỉnh stack ra khỏi stack
    Node pTemp = _top;
    _top = pTemp.Next;
    int temp = pTemp.Data;
    _size--;
    pTemp = null;
    return temp;
}
```

Xem giá trị phần tử trên đỉnh Stack

```
/*
* Xem giá trị phần tử trên đỉnh stack
* return: Giá trị phần tử đang ở trên đỉnh stack, nếu
stack rỗng thì thông báo ngoại lệ
*/
public int Peek()
{
    if (_top == null)
        throw new Exception("Stack rong");
    return _top.Data;
}
```

Display – Hiển thị Stack

```
/*
* Hiển thị toàn bộ Stack
*/
public void Display()
{
    if (IsEmpty())
        Console.WriteLine("Stack rong!");
    else
    {
        Node p = _top;
        while (p!=null)
        {
            Console.WriteLine(p.Data);
            p = p.Next;
        }
    }
}
```

Hủy Stack

```
/*
* Hủy stack
*/
public void Clear()
{
    Node tempNode;
    while (_top!=null)
    {
        tempNode = _top;
        _top = tempNode.Next;
        tempNode = null;
    }
    _size = 0;
}
```

Ứng dụng của Stack

Stack thích hợp lưu trữ các loại dữ liệu mà trình tự truy xuất ngược với trình tự lưu trữ.

❖***Một số ứng dụng của Stack:***

- *Trong trình biên dịch , khi thực hiện các thủ tục, Stack được sử dụng để lưu môi trường của các thủ tục.*
- *Lưu dữ liệu khi giải một số bài toán của lý thuyết đồ thị (như tìm đường đi)*
- *Khử đệ qui*
- *Chuyển đổi các cơ số (nhị phân, thập phân, bát phân,...)*
- *...*