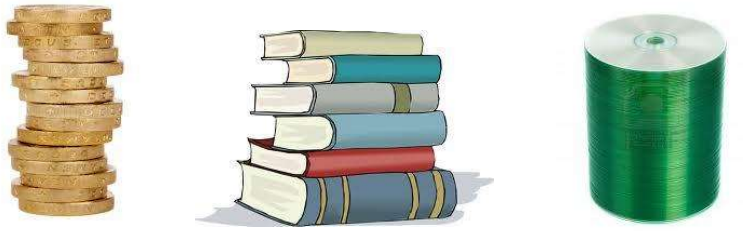
CHƯƠNG 3
NGĂN XẾP VÀ HÀNG ĐỢI

Biên soạn: ThS.Phạm Văn Đăng
Email: pvdang@ntt.edu.vn

Khái niệm Stack



Nội dung

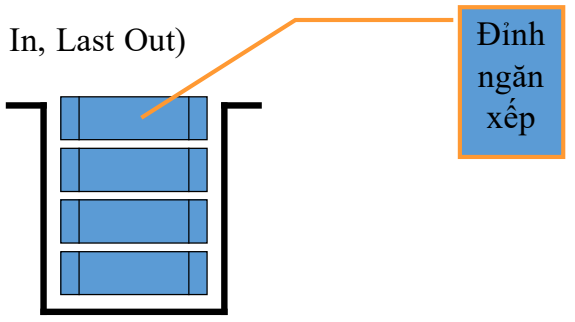
- Trình bày khái niệm Ngăn xếp (Stack) và Hàng đợi (Queue)
- Minh họa các ứng dụng
- Các phương pháp xây dựng Stack và Queue

Khái niệm Stack

- Gồm nhiều phần tử
- Hoạt động theo cơ chế "***Vào sau – Ra trước***"

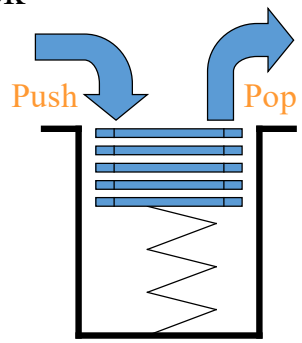
(LIFO – Last In, First Out) or

(FILO – First In, Last Out)

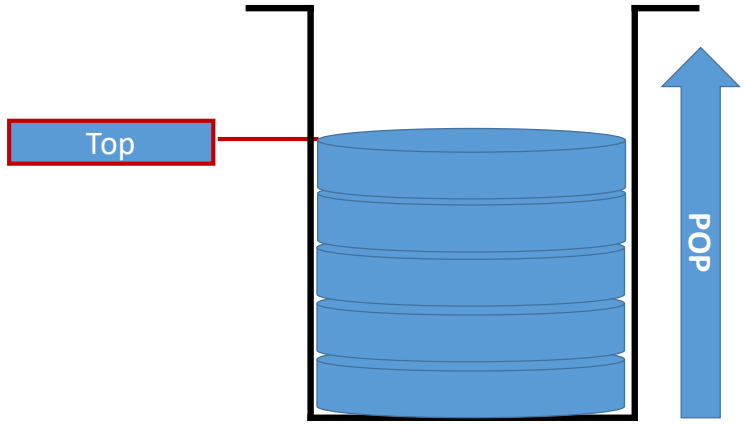


Các thao tác cơ bản trên Stack

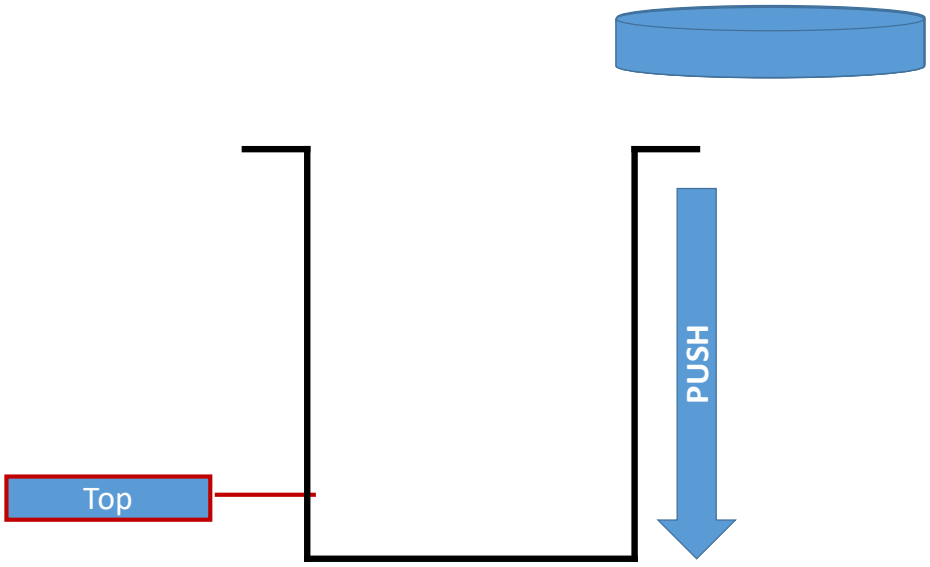
- *getSize*: lấy kích thước trong Stack
- *isEmpty*: kiểm tra Stack rỗng?
- *isFull*: kiểm tra Stack đầy?
- *peek*: xem phần tử trên đỉnh (top)
- *push*: thêm 1 phần tử vào Stack
- *pop*: lấy ra 1 phần tử khỏi Stack
- *display*: duyệt Stack



Thao tác Pop khỏi stack



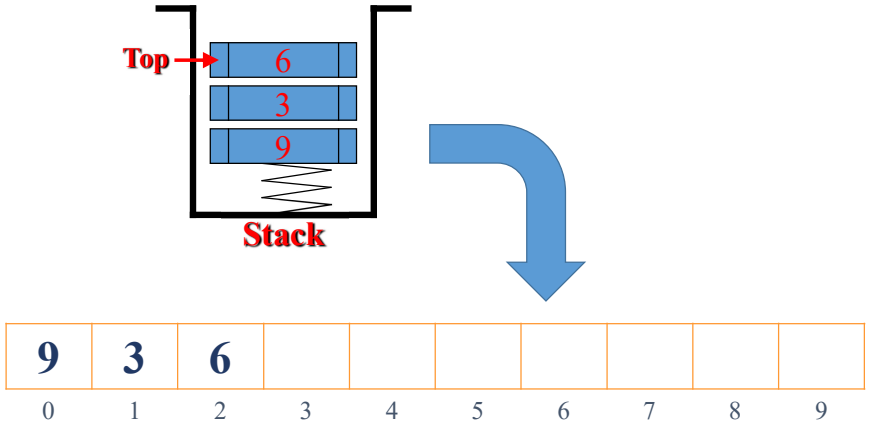
Thao tác Push vào Stack



Cách xây dựng Stack

Mảng 1 chiều	Danh sách liên kết
▪ *Viết chương trình dễ dàng, nhanh chóng* ▪ *Bị hạn chế do số lượng phần tử cố định* ▪ *Tốn chi phí tái cấp phát và sao chép vùng nhớ nếu sử dụng mảng động*	▪ *Phức tạp khi triển khai chương trình* ▪ *Không bị cố định về số phần tử, phụ thuộc vào bộ nhớ*

Stack – Sử dụng mảng



Các thao tác trên Stack

Kiểm tra Stack rỗng

```
public boolean isEmpty()
{
      return top == -1;
}
```

```
if(top==-1)
        return   true;
else
        return  false;
```

Kiểm tra Stack đầy

```
public boolean isFull()
{
      return top == size -1 ;
}
```

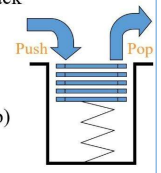
```
if(top==size-1)
        return   true;
return  false;
```


Stack số nguyên – Sử dụng mảng

```
class  arrayStack
{
      protected int arr[];              //Mảng chứa các phần tử
      protected int top, size, len;     //Số phần tử tối đa
      public  arrayStack(int   n)       //Hàm khởi tạo Stack
      {
            size = n;
            len = 0;
            arr = new  int [size];
            top = -1;
      }
}
```

Các thao tác cơ bản trên Stack

- *getSize*: lấy kích thước trong Stack
- *isEmpty*: kiểm tra Stack rỗng?
- *isFull*: kiểm tra Stack đầy?
- *peek*: xem phần tử trên đỉnh (top)
- *push*: thêm 1 phần tử vào Stack
- *pop*: lấy ra 1 phần tử khỏi Stack
- *display*: duyệt Stack



Các thao tác trên Stack

Lấy kích thước (số phần tử) trong Stack

```
public int getSize()
{
      return len ;
}
```

Xem phần tử trên đỉnh Stack

```
public int peek()
{
     if( isEmpty() )
        throw  new  NoSuchElementException("Stack rỗng");
     return  arr[top];
}
```

```
if(isEmpty()==true)
        throw  new  …
return   ….
```

Các thao tác trên Stack

Thêm một phần tử vào Stack

```
public  void  push(int  i)
{
      if(top + 1 >= size)   hoặc  if(isFull())
         throw new IndexOutOfBoundsException("Stack  full");
      if(top + 1 < size )
         arr[++top] = i;
      len++ ;
}
```


Các thao tác trên Stack

Lấy một phần tử ra khỏi Stack

```
public  int  pop()
{
      if(isEmpty() )
            throw new NoSuchElementException("Underflow  Exception");
      len-- ;
      return  arr[top--];
}
```


Các thao tác trên Stack

Duyệt Stack

```
public void display()
{
      System.out.print("\nStack = ");
      if (len == 0){
            System.out.print(“Stack Rỗng\n");
            return ;
      }
      for (int i = top; i >= 0; i--)
            System.out.print(arr[i]+" ");
      System.out.println();
}
```


Bài tập 1

1. Khai báo cấu trúc Stack
2. Kiểm tra Stack rỗng (isEmpty)
3. Khởi tạo top=-1 (isEmpty)
4. Thêm một 1 phần tử vào Stack (push)
5. Lấy một phần tử ra khỏi Stack (pop)
6. Viết hàm main gọi các thao tác trên

Bài tập 1

```java
package stack;
/**
 * @author PHAM VAN DANG
 */
public class Stack {
    static final int MAX = 1000; //Khai báo hằng số MAX
    int top;
    int a[] = new int[MAX];      // Maximum size of Stack

    boolean isEmpty()
    {
        return (top < 0);
    }
    Stack()
    {
        top = -1;
    }
```


Bài tập 1

```java
    boolean push(int x)
    {
        if (top >= (MAX-1))
        {
            System.out.println("Stack đầy");
            return false;
        }
        else
        {
            a[++top] = x;
            System.out.println(x + " đã push vào stack");
            return true;
        }
    }
```


Bài tập 1

```java
    int pop()
    {
        if (top < 0)
        {
            System.out.println("Stack rỗng!");
            return 0;
        }
        else
        {
            int x = a[top--];
            return x;
        }
    }
    public static void main(String[] args) {
        Stack s = new Stack();
        s.push(10);
        s.push(20);
        s.push(30);
        System.out.println(s.pop() + " Pop ra khỏi stack.");
    }
}
```


Bài tập 2

Viết chương trình khai báo Stack (dùng mảng) có chứa tối đa 20 số nguyên, với menu thực hiện như sau:

1. Push vào Stack
2. Pop ra khỏi Stack
3. Xem phần tử trên đỉnh Stack
4. Kiểm tra Stack rỗng
5. Kiểm tra Stack đầy
6. Số phần tử trong Stack
7. Duyệt Stack
8. Thoát
9. Nhập các phần tử vào Stack

Nhấn 1 số để chọn:

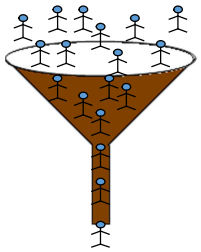
Bài tập 3

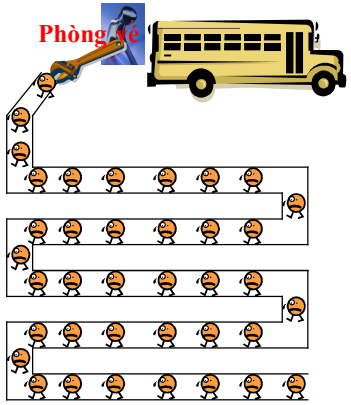
Viết chương trình khai báo Stack (dùng mảng) có chứa n số nhị phân, với menu thực hiện như sau:

1. Push vào Stack
2. Pop ra khỏi Stack
3. Xem phần tử trên đỉnh Stack
4. Kiểm tra Stack rỗng
5. Kiểm tra Stack đầy
6. Số phần tử trong Stack
7. Duyệt Stack
8. Thoát
9. Chuyển số thập phân sang nhị phân lưu vào Stack.
10. Nhập vào một số nguyên dương n, sử dụng stack để chứa các chữ số của n, in ra các chữ số của n và đồng thời tính tổng các chữ số của n.

Nhấn 1 số để chọn:

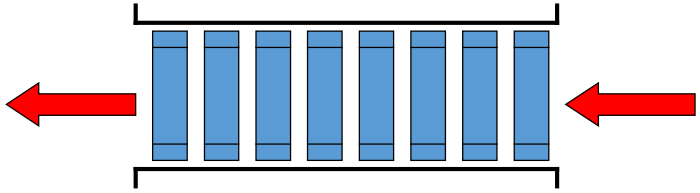
Queue





Queue – Định nghĩa

- Hàng đợi là một cấu trúc:
  - Gồm nhiều phần tử có thứ tự
  - Hoạt động theo cơ chế "Vào trước, ra trước" (FIFO - First In First Out)

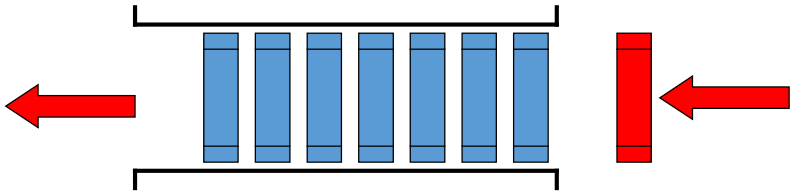


Queue – Định nghĩa

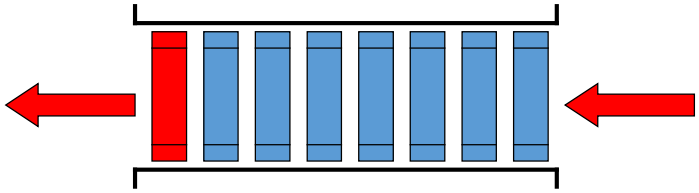
- Các thao tác cơ bản trên hàng đợi
  - *arrayQueue*: khởi tạo hàng đợi
  - *isEmpty*: kiểm tra hàng đợi rỗng
  - *isFull*: kiểm tra hàng đợi đầy
  - *getSize*: lấy kích thước hang đợi
  - *peek()*: xem phần tử Front của hàng đợi
  - *enQueue*: thêm 1 phần tử vào cuối hàng đợi, có thể làm hàng đợi đầy
  - *deQueue*: lấy ra 1 phần tử từ đầu Queue, có thể làm Queue rỗng
  - *display*: duyệt hàng đợi

Queue

- Minh họa thao tác EnQueue



- Minh họa thao tác DeQueue



Queue số nguyên – Sử dụng mảng

```
class arrayQueue
{
        protected int   Queue[];                //Khai báo mảng 1D
        protected int   front, rear, size, len; //Khai báo 4 thuộc tính
        public  arrayQueue(int  n) {            //Hàm tạo Queue
                size = n;
                len  = 0;
                Queue = new int[size];
                front = -1;
                rear = -1;
        }
};
```


Các thao tác trên Queue

Kiểm tra Queue rỗng

```
public boolean isEmpty()
{
        return  front == -1;
}
```

Kiểm tra Queue đầy

```
public boolean isFull()
{
        return  front==0 && rear == size -1;
}
```


Các thao tác trên Queue

Lấy kích thước của Queue

```
public int getSize() {  return len ;  }
```

Xem phần tử Front của Queue

```
public int peek()
{
        if (isEmpty())
                throw new NoSuchElementException("Queue rỗng!");
        return Queue[front];
}
```

Các thao tác trên Queue

Thêm 1 phần tử vào Queue (Enqueue)

```
public void insert(int i)
{
    if (rear == -1)
    {   front = 0;
        rear = 0;
        Queue[rear] = i;
    }
    else if (rear + 1 >= size)
        throw new IndexOutOfBoundsException("Queue đầy!");
    else if ( rear + 1 < size)
        Queue[++rear] = i;
    len++ ;
}
```


Các thao tác trên Queue

Lấy 1 phần tử ra khỏi Queue (Dequeue)

```
public int remove() {
    if (isEmpty())
        throw new NoSuchElementException("Queue rỗng!");
    else {
        len-- ;
        int ele = Queue[front];
        if ( front == rear){
            front = -1;
            rear = -1;
        }
        else
            front++;
        return ele;
    }
}
```


Các thao tác trên Queue

Duyệt Queue

```
public void display()
{
    System.out.print("\nNội dung của Queue : ");
    if (len == 0)
    {
        System.out.print("Queue rỗng\n");
        return ;
    }
    for (int i = front; i <= rear; i++)
        System.out.print(Queue[i]+" ");
    System.out.println();
}
```


Bài tập 1

1. Khai báo Queue chứa số nguyên
2. Hàm tạo Queue
3. Hàm kiểm tra Queue đầy
4. Hàm kiểm tra Queue rỗng
5. Hàm thêm 1 phần tử vào Queue
6. Lấy 1 phần tử khỏi Queue
7. Hàm lấy phần tử đầu Queue (front)
8. Hàm lấy phần từ cuối Queue (rear)
9. Hàm main gọi thực hiện các hàm trên

Bài tập 1

```
1  package queue;
2  /**
3   * @author PHAM VAN DANG
4   */
5  public class Queue {
6      int front, rear, size;
7      int  capacity;
8      int array[];
9      public Queue(int capacity)
10     {
11         this.capacity = capacity;
12         front = this.size = 0;
13         rear = capacity - 1;
14         array = new int[this.capacity];
15     }
16     boolean isFull(Queue queue)
17     {
18         return (queue.size == queue.capacity);
19     }
```


Bài tập 1

```
20     boolean isEmpty(Queue queue)
21     {  return (queue.size == 0); }
22     void enqueue( int item)
23     {
24         if (isFull(this))
25             return;
26         this.rear = (this.rear + 1)%this.capacity;
27         this.array[this.rear] = item;
28         this.size = this.size + 1;
29         System.out.println(item+ " enqueued to queue");
30     }
31     int dequeue()
32     {
33         if (isEmpty(this))
34             return Integer.MIN_VALUE;
35         int item = this.array[this.front];
36         this.front = (this.front + 1)%this.capacity;
37         this.size = this.size - 1;
38         return item;
39     }
```


Bài tập 1

```
40     int front()
41     {
42         if (isEmpty(this))
43             return Integer.MIN_VALUE;
44
45         return this.array[this.front];
46     }
47     int rear()
48     {
49         if (isEmpty(this))
50             return Integer.MIN_VALUE;
51
52         return this.array[this.rear];
53     }
```


Bài tập 1

```
54     public static void main(String[] args)
55     {
56         Queue queue = new Queue(1000);
57         queue.enqueue(10);
58         queue.enqueue(20);
59         queue.enqueue(30);
60         queue.enqueue(40);
61         System.out.println(queue.dequeue() + " dequeued from queue\n");
62         System.out.println("Front item is " + queue.front());
63         System.out.println("Rear item is " + queue.rear());
64     }
65 }
```

Bài tập 2

Viết chương trình khai báo Queue (dùng mảng) có chứa tối đa 20 số nguyên, với menu thực hiện như sau:

1. Thêm vào Queue (EnQueue)
2. Lấy ra khỏi Queue (DeQueue)
3. Xem phần tử đầu Queue
4. Kiểm tra Queue rỗng
5. Kiểm tra Queue đầy
6. Kích thước Queue
7. Duyệt Queue
8. Thoát
9. Nhập vào dãy số nguyên, sử dụng queue để chứa các số lẻ riêng và chứa các số chẵn riêng

Nhấn 1 số để chọn

Chúc bạn thành công!