ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# SLIDE BÀI GIẢNG

## MÔN

# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, KHU PHỐ 6, PHƯỜNG LINH TRUNG, QUẬN THỦ ĐỨC, TP. HỒ CHÍ MINH

[T] 08 3725 2002 101 | [F] 08 3725 2148 | [W] www.uit.edu.vn | [E] info@uit.edu.vn

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

## CHƯƠNG III

# STACK VÀ QUEUE

Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn
www.uit.edu.vn

# MỤC TIÊU CHƯƠNG III

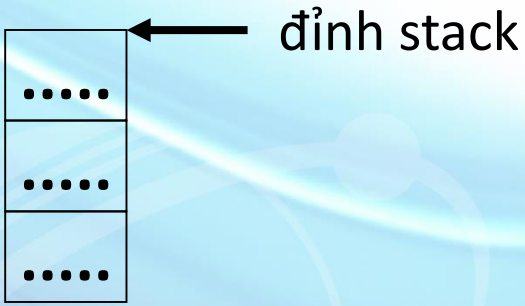
- Hiểu các khái niệm Stack và Queue
- Hiểu các thao thác trên Stack và Queue
- Áp dụng các cấu trúc đã biết để cài đặt Stack và Queue
- Áp dụng Stack và Queue để giải quyết bài toán khử đệ quy, tính giá trị biểu thức, …

# STACK VÀ QUEUE

- STACK
- CÁC THAO TÁC TRÊN STACK
- QUEUE
- CÁC THAO TÁC TRÊN QUEUE

# STACK

## ❖KHÁI NIỆM




- Là danh sách.
- Quản lý theo nguyên tắc LIFO (Last In First Out)
- Có đỉnh Stack luôn chỉ vào phần tử được thêm sau cùng.

# STACK

## ❖ CÁC THAO TÁC ĐẶC TRƯNG

- push: thêm phần tử dữ liệu x vào stack.
- pop: lấy phần tử tại đỉnh ra khỏi stack.
- isEmpty: kiểm tra stack có rỗng hay không.
- top: lấy giá trị của phần tử tại đỉnh stack mà không hủy nó.

# STACK

## ❖CẤU TRÚC DỮ LIỆU CHO STACK

- Mảng

```
#define MAX 100
struct Stack {
  TenDulieu data[MAX];
  int sp; //stack pointer
};
```

# STACK

## ❖CẤU TRÚC DỮ LIỆU CHO STACK

- Danh sách đơn

```
struct Stack {
  List data;
};
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

- Tạo Stack rỗng

```
void createStack(Stack &s) {
  s.sp = -1;
}
```

- Kiểm tra Stack có rỗng hay không

```
int isEmpty(Stack &s) {
  return (s.sp == -1);
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

- Kiểm tra Stack đầy (do cài đặt trên mảng)

```
int isFull(Stack &s) {
  return (s.sp >= MAX);
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

- Đưa phần tử vào Stack

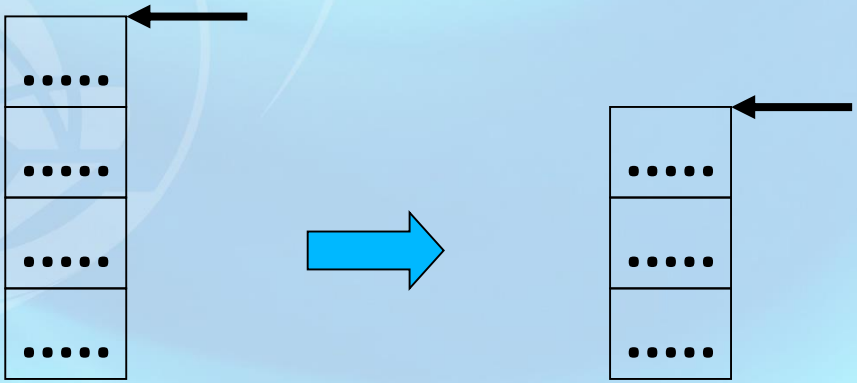
```
int push(Stack &s, TenDulieu x) {
  if (IsFull(s))
     return 0;
  s.sp++;
  s.data[s.sp] = x;
  return 1;
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

- Lấy phần tử ra khỏi Stack

```
int pop(Stack &s, TenDulieu &x) {
  if (IsEmpty(s))
     return 0;
  x = s.data[s.sp]
  s.sp--;
  return 1;
}
```



# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

- Lấy giá trị phần tử ở đỉnh stack

```
int top(Stack &s, TenDulieu &x) {
  if (IsEmpty(s)) return 0;
  x = s.data[sp];
  return 1;
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

Ví dụ: Cho định nghĩa cấu trúc stack NameStack

```
#define MAX 30
struct Name {
  char dat[40];
};
struct NameStack {
  Name data[MAX];
  int sp;
};
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG MẢNG

Cho biết kết quả của đoạn chương trình

```
Name n1 = {"mot"}, n2 = {"hai"},
        n3 = {"ba"}, n4 = {"bon"}, n;
NameStack s; createStack(s);
push(s, n1); push(s, n2);
if (pop(s, n)) cout << n.dat << endl;
push(s, n3);
if (pop(s, n)) cout << n.dat << endl;
if (pop(s, n)) cout << n.dat << endl;
push(s, n4);
if (pop(s, n)) cout << n.dat << endl;
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG DANH SÁCH ĐƠN

- Tạo Stack rỗng

```
void createStack(Stack &s) {
  createList(s.data);
}
```

- Kiểm tra Stack rỗng

```
int isEmpty(Stack &s) {
  return s.data.pHead == NULL;
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG DANH SÁCH ĐƠN

- Đưa dữ liệu vào stack

```
void push(Stack &s, Node *p) {
  addHead(s.data, p);
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG DANH SÁCH ĐƠN

- Lấy dữ liệu ra khỏi stack

```
int pop(Stack &s, TenDulieu &x) {
  return removeHead(s.data, x);
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖DÙNG DANH SÁCH ĐƠN

- Lấy dữ liệu của phần tử tại đỉnh stack

```
int top(Stack &s, TenDulieu &x) {
  if (isEmpty(s)) return 0;
  x = s.data.pHead->info;
  return 1;
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖ ỨNG DỤNG CỦA STACK

- Khử đệ quy đuôi (Gọi hàm đệ quy ở cuối cùng trong hàm đệ quy)

Ví dụ: khử đệ quy cho hàm sau:

# CÁC THAO TÁC TRÊN STACK

## ❖ỨNG DỤNG CỦA STACK

```
void thapHaNoi(int n,int a,int b,int t) {
  if (n == 1) {
     cout<<"chuyen 1 dia tu "<<a<<"-"<<b<<endl;
     return;
  }
  thapHaNoi(n-1, a, t, b);
  thapHaNoi(1, a, b, t);
  thapHaNoi(n-1,t, b, a);
}
```

# CÁC THAO TÁC TRÊN STACK

```
struct Chuyen {
  int n, a, b, t;
};
struct Node {
  Chuyen data;
  Node * pNext;
}
struct List {
  Node *pHead, *pTail;
};
```

# CÁC THAO TÁC TRÊN STACK

```
struct Stack {
  List data;
};

//cài đặt các hàm cơ bản của danh sách
//đơn và stack
```

# CÁC THAO TÁC TRÊN STACK

```
void thapHaNoi(int n, int a, int b, int t){
  Stack stk;
  Chuyen dl;
  createStack(stk);
  Node *p = createNode(n, a, b, t);
  if (!p) return;
  push(stk, p);
```

# CÁC THAO TÁC TRÊN STACK

```
  while (!isEmpty(stk)) {
     pop(stk, dl);
     if (dl.n == 1)
     {  cout <<"chuyen 1 dia tu "<<a<<"-"<<b
              <<endl;   continue; }
     p = createNode(dl.n-1,dl.t,dl.b,dl.a);
     push(stk, p);
     p = createNode(1,dl.a,dl.b,dl.t);
     push(stk, p);
     p = createNode(dl.n-1,dl.a,dl.b,dl.t);
     push(stk, p);
  }
}
```

# CÁC THAO TÁC TRÊN STACK

## ❖ ỨNG DỤNG CỦA STACK

- Lưu vết các quá trình:
  - Quay lui,
  - Vét cạn,
  - Tìm kiếm theo chiều sâu.

# QUEUE

## ❖QUEUE - HÀNG ĐỢI

- Là danh sách.
- Quản lý theo nguyên tắc FIFO (First In First Out)

# QUEUE

## ❖CÁC THAO TÁC ĐẶC TRƯNG

- enQueue: thêm phần tử vào hàng đợi
- deQueue: lấy phần tử ra khỏi hàng đợi
- isEmpty: kiểm tra hàng đợi có rỗng không
- front: lấy dữ liệu của phần tử đầu hàng đợi mà không hủy nó.

# QUEUE

## ❖CẤU TRÚC DỮ LIỆU CHO QUEUE

- Mảng

```
struct Queue {
  TenDulieu data[MAX];
  int front, rear;
};
```

# QUEUE

## ❖CẤU TRÚC DỮ LIỆU CHO QUEUE

- Danh sách liên kết

```
struct Queue {
  List data;
};
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG MẢNG

- Tạo hàng đợi rỗng

```
void createQueue(Queue &q) {
    q.front = -1; q.rear = -1;
}
```

- Kiểm tra hàng đợi có rỗng hay không

```
int isEmpty(Queue &q) {
    return q.front == -1;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG MẢNG

- Thêm phần tử vào hàng đợi

```
int enQueue(Queue &q, TenDulieu x) {
  if ( (q.front == q.rear+1) ||
     (q.rear-q.front+1 == MAX))
     return 0; // hàng đợi đã đầy
  if (q.front == -1) q.front = 0;
  q.rear++;
  if (q.rear == MAX) q.rear = 0;
  q.data[q.rear] = x;
  return 1;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG MẢNG

- Lấy phần tử ra khỏi hàng đợi

```
int deQueue(Queue &q, TenDulieu &x) {
  if (isEmpty(q))return 0;
  x = q.data[q.front];
  if (q.front == q.rear)
  { q.front = -1; q.rear = -1; }
  else
  {  q.front++;
     if (q.front >= MAX) q.front = 0; }
  return 1;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG MẢNG

- Lấy dữ liệu phần tử đầu hàng đợi

```
int front(Queue &q, TenDulieu &x) {
  if (isEmpty(q)) return 0;
  x = q.data[q.front];
  return 1;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG DANH SÁCH ĐƠN

- Tạo hàng đợi rỗng

```
void createQueue(Queue &q) {
  createList(q.data);
}
```

- Kiểm tra hàng đợi có rỗng hay không

```
int isEmpty(Queue &q) {
  return q.data.pHead == NULL;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG DANH SÁCH ĐƠN

- Thêm một phần tử vào hàng đợi

```
void enQueue(Queue &q, Node *p) {
  addTail(q.data, p);
}
```

- Lấy một phần tử ra khỏi hàng đợi

```
void deQueue(Queue &q, TenDulieu &x) {
  removeHead(q.data, x);
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖DÙNG DANH SÁCH ĐƠN

- Lấy giá trị phần tử đầu hàng đợi

```
int front(Queue &q, TenDulieu &x) {
  if (isEmpty(q.data)) return 0;
  x = q.data.pHead->info;
  return 1;
}
```

# CÁC THAO TÁC TRÊN QUEUE

## ❖ ỨNG DỤNG CỦA QUEUE

- Tổ chức lưu vết quá trình:
  - Tìm kiếm theo chiều rộng,
  - Quay lui,
  - Vét cạn
- Tổ chức quản lý và phân phối công việc (quản lý các tiến trình trong hệ điều hành).

# BÀI TẬP

1) Cho biết quy tắc khi dùng stack để chuyển một biểu thức dạng trung tố sang dạng hậu tố

Áp dụng quy tắc trên, mô tả quá trình chuyển các biểu thức sau về dạng hậu tố:

a) a – b * c ^ d + f

b) 1 + 2 + 3 * 4 + 5 – 6 * 7 + 8

# BÀI TẬP

2) Cho biết quy tắc khi dùng stack để tính toán một biểu thức hậu tố và mô tả quá trình tính toán các biểu thức sau:

a) 1 2 + 3 1 + * 1 1 + 1 + /

b) 3 4 + 3 5 + * 7 + 8 *