DANH SÁCH LIÊN KẾT ĐƠN (SINGLY LINKED LIST) STACK- QUEUE

DATA STRUCTURES AND ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

Nội dung

1. STACK

- Khái niệm ngăn xếp.
- Ứng dụng của ngăn xếp.
- Các thao tác trên ngăn xếp.
- Các phương án cài đặt ngăn xếp.

2. QUEUE

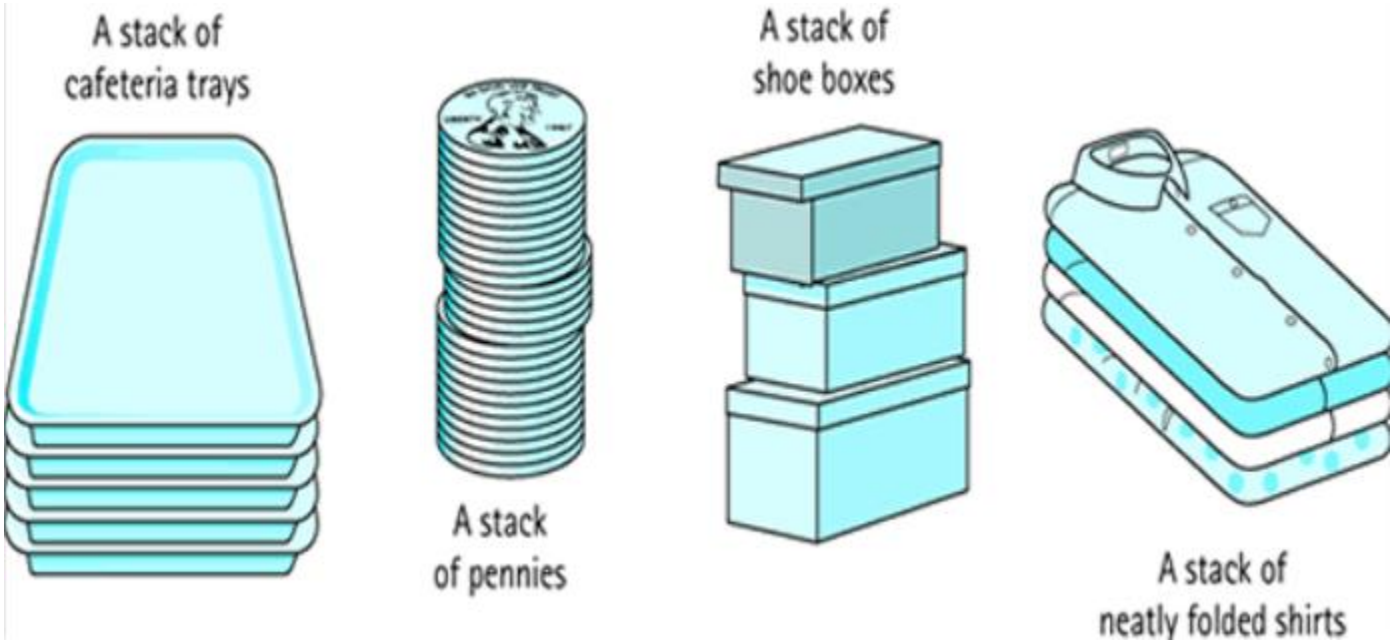
- Khái niệm Queue.
- Ứng dụng của hàng đợi.
- Các thao tác trên queue
- Các phương án cài đặt hàng đợi.

3. BÀI TOÁN

4. BÀI TẬP

1. STACK

- Stack (ngăn xếp): Là 1 vật chứa các đối tượng làm việc theo cơ chế **LIFO (Last In First Out)**, tc việc thêm 1 đối tượng vào Stack hoặc lấy 1 đối tượng ra khỏi Stack được thực hiện theo cơ chế "vào sau ra trước"

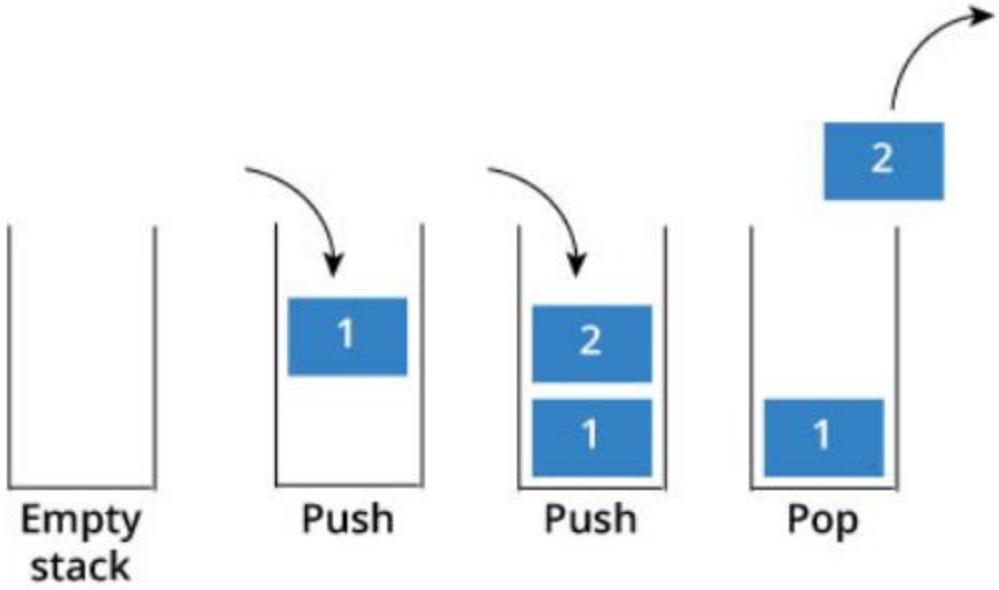


1. STACK

- Ứng dụng:
  - Undo trong Word, nút back trong trình duyệt web
  - Cài đặt / thực hiện lời gọi hàm trong trình biên dịch
  - Khử đệ quy đuôi
  - Lưu vết các quá trình quay lui, vét cạn
  - Các bài toán như: duyệt theo chiều sâu trong đồ thị, tính biểu thức, ...

Các thao tác trên Stack

- Push(o): Thêm đối tượng o vào Stack
- Pop(): Lấy đối tượng từ Stac
- isEmpty(): Kiểm tra Stack có rỗng hay không
- isFull(): Kiểm tra Stack có đầy hay không
- Top(): Trả về giá trị của phần tử nằm đầu Stack mà không hủy nó khỏi Stack.

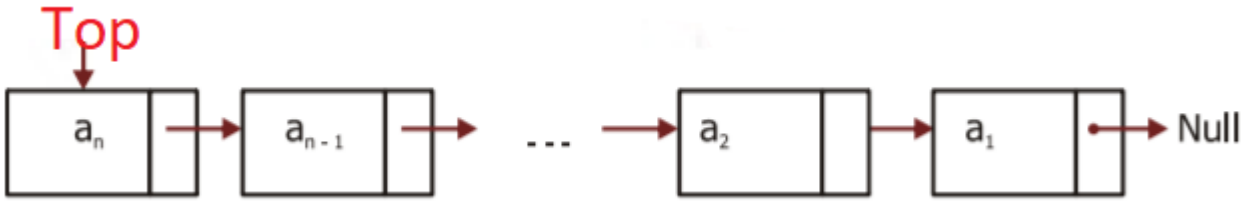


Cài đặt Stack

- Dùng mảng 1 chiều

Data S [N];
int t;

- Dùng danh sách liên kết đơn



Stack S;

- Chú ý: Thêm và hủy cùng phía

Cài Stack bằng mảng 1 chiều

- Cấu trúc dữ liệu của Stack

```
struct Stack {
    int a[MAX];
    int t;
};
```

- Khởi tạo Stack

```
void CreateStack(Stack &s) {
    s.t = -1;
}
```

Kiểm tra tính rỗng và đầy của Stack

```
bool isEmpty(Stack s) { //Stack có rỗng hay không
    if (s.t == -1)
        return 1;
    return 0;
}

bool isFull(Stack s) { //Kiểm tra Stack có đầy hay không
    if (s.t >= MAX)
        return 1;
    return 0;
}
```

Thêm 1 phần tử vào Stack

```
bool Push(Stack &s, int x) {
    if (isFull(s) == 0)
        s.a[++s.t] = x;
        return 1;
    }
    return 0;
}
```

Lấy 1 phần tử từ Stack

```
int Pop(Stack &s, int &x) {
    if (isEmpty(s) == 0) {
        x = s.a[s.t--];
        return 1;
    }
    return 0;
}
```

Cài Stack bằng danh sách liên kết

- Kiểm tra tính rỗng của Stack

```
// Cấu trúc của một node
struct NODE {
  int info;
  NODE* pNext;
};
// Cấu trúc của một Stack
struct STACK { NODE* pHead;};


int isEmpty(STACK &s) {
    if (s.pHead == NULL)//Stack rong
          return 1;
    return 0;
}
```

Thêm 1 phần tử vào Stack

```
void Push(STACK &s, NODE *p) { // AddHead
    if (s.pHead == NULL) {
        s.pHead = p;
        s.pTail = p;
    }
    else {
        p->pNext = s.pHead;
        s.pHead = p;
    }
}
```

Lấy 1 phần tử từ Stack

```
bool Pop(STACK &s, int &x) {
    NODE *p;
    if (isEmpty(s) != true) {
        if (s.pHead != NULL) {
            p = s.pHead;
            x = p->info;
            s.pHead = s.pHead->pNext;
            if (s.pHead == NULL)
                s.pTail = NULL;
            return 1;
        }
    }
    return 0;
}
```

Nội dung

1. STACK

- Khái niệm ngăn xếp.
- Ứng dụng của ngăn xếp.
- Đặc tả ngăn xếp.
- Các phương án cài đặt ngăn xếp.

2. QUEUE

- Khái niệm Queue.
- Ứng dụng của hàng đợi.
- Đặc tả hàng đợi.
- Các phương án cài đặt hàng đợi.

Nội dung

1. STACK
   - Khái niệm ngăn xếp.
   - Ứng dụng của ngăn xếp.
   - Các thao tác trên ngăn xếp.
   - Các phương án cài đặt ngăn xếp.

2. QUEUE
   - Khái niệm Queue.
   - Ứng dụng của hàng đợi.
   - Các thao tác trên queue
   - Các phương án cài đặt hàng đợi.

3. BÀI TOÁN

4. BÀI TẬP

2. QUEUE

- Queue (hàng đợi): Là 1 vật chứa các đối tượng làm việc theo cơ chế FIFO (First In First Out), tức việc thêm 1 đối tượng vào hàng đợi hay lấy 1 đối tượng ra khỏi hàng đợi thực hiện theo cơ chế "vào trước ra trước".

Một số dạng Hàng đợi

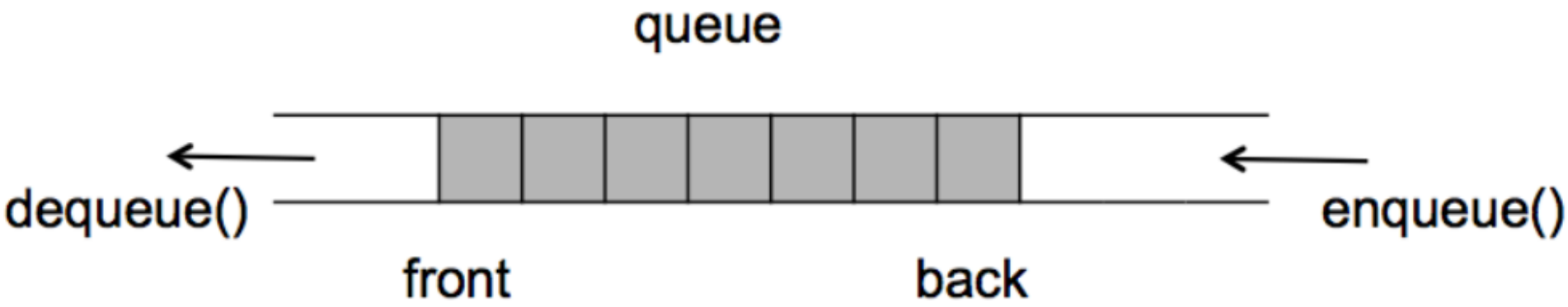
- Hàng đợi tuyến tính - Linear Queues
  - Tổ chức hàng đợi theo nghĩa thông thường.
- Hàng đợi vòng - Circular Queues
  - Giải quyết việc thiếu bộ nhớ khi sử dụng hàng đợi.
- Hàng đợi ưu tiên - Priority Queues
  - Mỗi phần tử có kết hợp thêm thông tin về độ ưu tiên.
  - Khi chương trình cần lấy một phần tử khỏi hàng đợi, nó sẽ xét những phần tử có độ ưu tiên cao trước.

2. QUEUE

- Ứng dụng:
  - 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, …
  - …

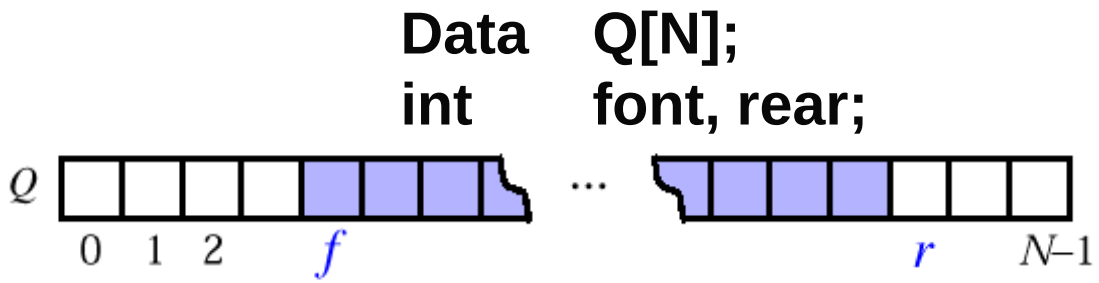
Các thao tác trên Queue

- EnQueue(O): Thêm đối tượng O vào cuối hàng đợi.
- DeQueue(): Lấy đối tượng ở đầu hàng đợi
- isEmpty(): Kiểm tra xem hàng đợi có rỗng hay không?
- Front(): Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó.

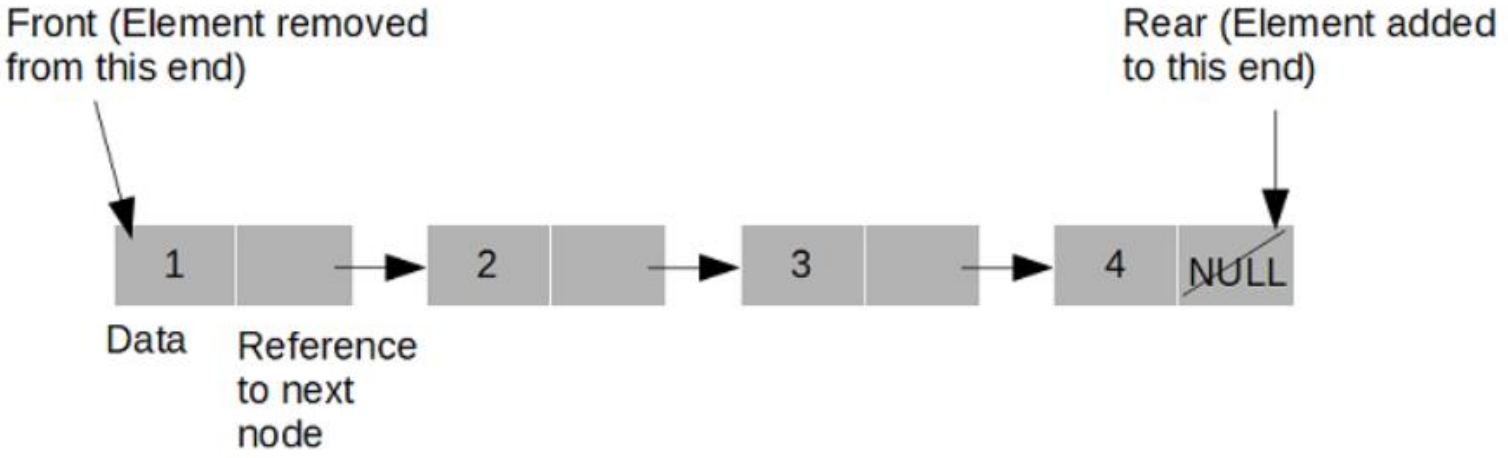


Cài đặt Queue

- Dùng mảng 1 chiều



- Dùng danh sách liên kết đơn



Cài đặt Queue bằng mảng 1 chiều

- *Cấu trúc dữ liệu:*

```
typedef struct tagQueue {
    int a[MAX];
    int Front; //chỉ số của phần tử đầu trong Queue
    int Rear; //chỉ số của phần tử cuối trong Queue
} Queue;
```

- *Khởi tạo Queue rỗng*

```
void CreateQueue(Queue &q) {
    q.Front = -1;
    q.Rear = -1;
}
```

Kiểm tra tính rỗng và đầy của Queue

```
bool isEmpty(Queue q) { // Queue có rỗng?
  if (q.Front == -1)
     return 1;
  return 0;
}

bool isFull(Queue q) { // Kiểm tra Queue có đầy?
  if (q.Rear - q.Front + 1 == MAX)
     return 1;
  return 0;
}
```

Thêm 1 phần tử vào Queue

```
void EnQueue(Queue &q, int x) {
    int f, r;
    if (isFull(q)) //queue bị đầy => không thêm được nữa
        printf("queue day roi khong the them vao duoc nua");
    else {
        if (q.Front == -1) {
            q.Front = 0;
            q.Rear = -1;
        }
        if (q.Rear == MAX - 1) { //Queue đầy ảo
            f = q.Front;
            r = q.Rear;
            for (int i = f; i <= r; i++)
                q.a[i - f] = q.a[i];
            q.Front = 0;
            q.Rear = r - f;
        }
        q.Rear++;
        q.a[q.Rear] = x;
    }
}
```

Lấy 1 phần tử từ Queue

```
bool DeQueue(Queue &q, int &x) {
    if (isEmpty(q)==0) { //queue khong rong
        x = q.a[q.Front];
        q.Front++;
        if (q.Front>q.Rear) { //truong hop co mot phan tu
          q.Front = -1;
          q.Rear = -1;
        }
        return 1;
    }
    //queue trong
    printf("Queue rong");
    return 0;
}
```

Cài đặt Queue bằng List

Kiểm tra Queue có rỗng?

```
// Cấu trúc của một node
struct NODE {
   int info;
   NODE* pNext;
};
// Cấu trúc của một DSLK
struct QUEUE {
       NODE* front;
       NODE* back;
};


bool isEmpty(QUEUE &Q) {
     if (Q.front == NULL) //Queue rỗng
              return 1;
     return 0;
}
```

Thêm 1 phần tử vào Queue

```
void EnQueue(QUEUE &Q, NODE *p) { // AddTail
    if (Q.front == NULL) {
        Q.front = p;
        Q.front = p;
    }
    else {
        Q.back->pNext = p;
        Q.back = p;
    }
}
```

Lấy 1 phần tử từ Queue

```
int DeQueue(QUEUE &Q, int &x) {
    NODE *p;
    if (isEmpty(Q) != 1) {
        if (Q.front != NULL) {
            p = Q.front;
            x = p->info;
            Q.front = Q.front ->pNext;
            if (Q.front == NULL)
                Q.back = NULL;
            return 1;
        }
    }
    return 0;
}
```

Nội dung

1. STACK
   - Khái niệm ngăn xếp.
   - Ứng dụng của ngăn xếp.
   - Các thao tác trên ngăn xếp.
   - Các phương án cài đặt ngăn xếp.
2. QUEUE
   - Khái niệm Queue.
   - Ứng dụng của hàng đợi.
   - Các thao tác trên queue
   - Các phương án cài đặt hàng đợi.
3. BÀI TOÁN
4. BÀI TẬP

3. BÀI TOÁN

3.1 Palindromes

3.2 Demerging

3.3 Tính giá trị của biểu thức

3.1 Palindromes

- Khái niệm: Một chuỗi được gọi là Palindrome nếu như đọc xuôi giống đọc ngược.
- Bài toán: Cho trước một chuỗi, kiểm tra xem chuỗi đó có phải là chuỗi palindrome hay không?
- Ví dụ về chuỗi palindrome: Able was I ere I saw Elba
- Giải pháp:
  - Để tránh ảnh hưởng tới chuỗi ban đầu, đọc chuỗi nói trên vào stack và queue.
  - So sánh từng phần tử của stack và queue, nếu giống nhau từng cặp thì đó là chuỗi Palindrome, ngược lại thì chuỗi trên không phải là chuỗi Palindrome.

3.2 Demerging

- Bài toán: Xem xét bài toán sau:
  - Giả sử, với một hệ thống quản lý nhân sự. Các bản ghi được lưu trên file.
  - Mỗi bản ghi gồm các trường: Họ tên, giới tính, ngày tháng năm sinh, ...
  - Dữ liệu trên đã được sắp theo ngày tháng năm sinh.
  - Cần tổ chức lại dữ liệu sao cho nữ được liệt kê trước nam nhưng vẫn giữ được tính đã sắp theo ngày tháng năm sinh.

3.2 Demerging

Cách giải quyết:

- **Ý tưởng không hiệu quả:**
  - Sử dụng thuật toán sắp xếp.
  - Độ phức tạp của thuật toán O(n log n) trong trường hợp tốt nhất.
- **Ý tưởng hiệu quả hơn:**
  - Sử dụng giải thuật demerging.
  - Độ phức tạp của giải thuật này là O(n).

3.2 Demerging

Giải thuật Demerging:

1. Tạo 2 queue rỗng, có tên lần lượt là NU và NAM.
2. Với mỗi bản ghi p, xem xét:
    2.1 Nếu p có giới tính nữ, đưa vào queue NU.
    2.2 Nếu p có giới tính nam, đưa vào queue NAM.
3. Xét queue NU, khi queue chưa rỗng:
    3.1 Lấy từng phần tử trong queue này.
    3.2 Ghi vào file output nào đó.
4. Xét queue NAM, khi queue chưa rỗng:
    4.1 Lấy từng phần tử trong queue này.
    4.2 Ghi tiếp vào file output trên.
5. Kết thúc giải thuật.

3. Tính giá trị biểu thức

Ví dụ: Tính giá trị của biểu thức sau:

$$(((2 + 3) * 5 / 8) – 2 + 3) / 3 + 2 * (5 - 1) = ?$$

Bài toán tìm đường đi ngắn nhất

Nội dung

1. STACK
   - Khái niệm ngăn xếp.
   - Ứng dụng của ngăn xếp.
   - Các thao tác trên ngăn xếp.
   - Các phương án cài đặt ngăn xếp.
2. QUEUE
   - Khái niệm Queue.
   - Ứng dụng của hàng đợi.
   - Các thao tác trên queue
   - Các phương án cài đặt hàng đợi.
3. BÀI TOÁN
4. BÀI TẬP

Bài tập

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

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

Với một ký tự 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?

2. Cài đặt chương trình cho phép thực hiện các phép tính +,-,*,/ trên các số có tối đa 30 chữ số, có chức năng nhớ (M+, M-, MC, MR).

3. Viết chương trình thực hiện các thao tác trên đa thức.

4. Hãy viết chương trình mô phỏng cho bài toán "Tháp Hà Nội" sử dụng ngăn xếp.

5. Viết chương trình tìm tất cả các cặp dấu ngoặc tương ứng trong một chương trình viết bằng ngôn ngữ C/C++.

Bài tập 2

Yêu cầu : Cài đặt một queue sử dụng 2 stack ?

Ứng dụng :

- Job interview
- Cài đặt queue ưu tiên
- …

Chúc các em học tốt!