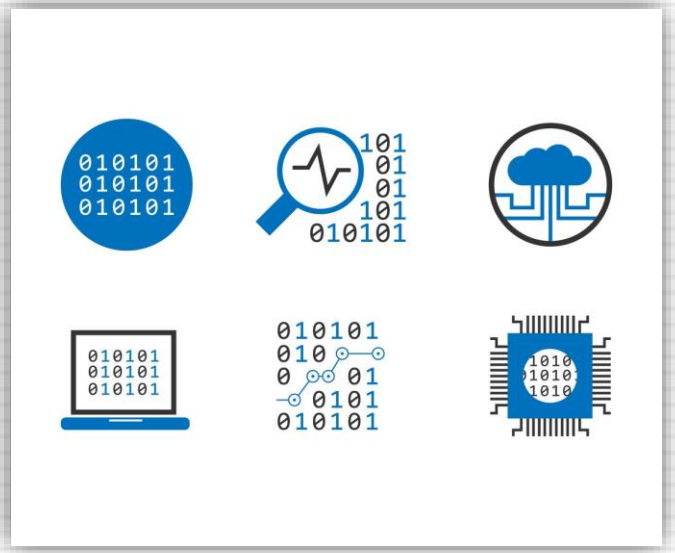
Đại Học Quốc Gia Tp. Hồ Chí Minh
Trường ĐH Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin

KỸ THUẬT LẬP TRÌNH



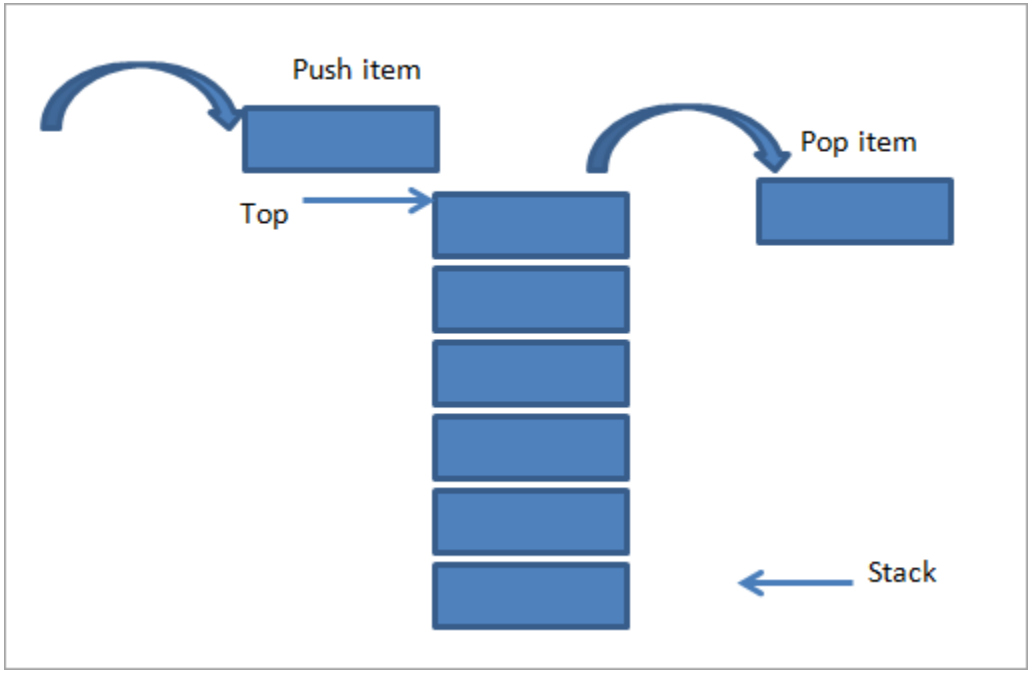
CHƯƠNG 5: NGĂN XẾP & HÀNG ĐỢI
(STACK & QUEUE)

GV: Phạm Nguyễn Sơn Tùng

Email: pnstung@fit.hcmus.edu.vn

GIỚI THIỆU VỀ NGĂN XẾP

Định nghĩa: là một cấu trúc dữ liệu tuân thủ theo quy tắc vào sau ra trước (Last In First Out)



CÁCH CÀI ĐẶT VÀ SỬ DỤNG

Cài đặt: Bạn có thể dùng mảng một chiều hoặc danh sách liên kết để cài đặt Stack.

Ngoài cách tự cài đặt, bạn còn có thể dùng thư viện STL (Standard Template Library) để sử dụng khi cần thiết.

CÁC THAO TÁC CỦA STACK

- **InitStack:** Khởi tạo Stack rỗng.
- **IsEmpty:** Kiểm tra Stack rỗng.
- **IsFull:** Kiểm tra Stack đầy.
- **Size:** Số lượng phần tử trong Stack.
- **Push:** Thêm 1 phần tử vào đỉnh Stack.
- **Pop:** Xóa 1 phần tử từ đỉnh Stack.
- **Top:** Xem phần tử đầu Stack (không xoá)

CÀI ĐẶT STACK BẰNG MẢNG 1 CHIỀU

CÁC THAO TÁC CỦA STACK

Thao tác: Khai báo cấu trúc Stack.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Stack
{
    int* list;
    int max;
    int size;
};
```

CÁC THAO TÁC CỦA STACK

Thao tác: InitStack Khởi tạo Stack rỗng.

```
bool InitStack(Stack& s, int n)
{
    s.list = new int[n];
    if (s.list == NULL)
        return false;
    s.max = n;
    s.size = 0;
    return true;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: IsEmpty Kiểm tra Stack rỗng.

```
bool IsEmpty(const Stack& s)
{
    if (s.size == 0)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: IsFull Kiểm tra Stack đầy.

```
bool IsFull(const Stack& s)
{
    if (s.size == s.max)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: Push Thêm 1 phần tử vào đỉnh Stack (thêm vào cuối).

```
bool Push(Stack& s, int newitem)
{
    if (IsFull(s) == true)
        return false;
    s.list[s.size] = newitem;
    s.size++;
    return true;
}
```

CÁC THAO TÁC CỦA STACK

- **Thao tác: Pop** xóa 1 phần tử từ đỉnh Stack (xóa cuối)

```
bool Pop(Stack& s)
{
    if (IsEmpty(s) == true)
        return false;
    s.size--;
    return true;
}
```

CÁC THAO TÁC CỦA STACK

- **Thao tác: Top** Xem phần tử đầu Stack (không xoá)

```
int Top(const Stack s)
{
    if (IsEmpty(s)==true)
        return INT_MIN;
    int topitem = s.list[s.size-1];
    return topitem;
}
```

CÀI ĐẶT STACK BẰNG DANH SÁCH LIÊN KẾT

CÁC THAO TÁC CỦA STACK

Thao tác: Khai báo cấu trúc Stack.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Node
{
    int data;
    struct Node* pNext;
};

struct Stack
{
    Node* Head;
    Node* Tail;
    int max;
    int size;
};
```

CÁC THAO TÁC CỦA STACK

Thao tác: InitStack Khởi tạo Stack rỗng.

```
void InitStack(Stack& s, int n)
{
    s.Head = NULL;
    s.Tail = NULL;
    s.max = n;
    s.size = 0;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: IsEmpty Kiểm tra Stack rỗng.

```
bool IsEmpty(const Stack& s)
{
    if (s.size == 0)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: IsFull Kiểm tra Stack đầy.

```
bool IsFull(const Stack& s)
{
    if (s.size == s.max)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: CreatNoe tạo node mới.

```
Node* CreateNode(int x)
{
    Node* newNode = new Node;
    if (newNode == NULL)
    {
        cout << "Khong du vung nho";
        return NULL;
    }
    newNode->data = x;
    newNode->pNext = NULL;
    return newNode;
}
```

CÁC THAO TÁC CỦA STACK

Thao tác: Push Thêm 1 phần tử vào đỉnh Stack.

```
bool Push(Stack& s, int x)
{
    if (IsFull(s) == true)
        return false;
    else
    {
        if (IsEmpty(s) == true)
        {
            Node* newnode = CreateNode(x);
            s.Head = s.Tail = newnode;
        }
        else //Tương đương với hàm thêm cuối
        {
            Node* newnode = CreateNode(x);
            s.Tail->pNext = newnode;
            s.Tail = newnode;
        }
        s.size++;
    }
    return true;
}
```

CÁC THAO TÁC CỦA STACK

- **Thao tác: Pop** xóa 1 phần tử từ đỉnh Stack.

```
bool Pop(Stack& s)
{
    if (IsEmpty(s) == true)
        return false;
    if (s.Tail != s.Head) // Nếu có 2 node trở lên thì xóa
cuối
    {
        Node* pDel = s.Tail;
        Node* pCur = s.Head;
        while (pCur->pNext != s.Tail)
            pCur = pCur->pNext;
        pCur->pNext = NULL;
        s.Tail = pCur;
        delete pDel;
    }
```

CÁC THAO TÁC CỦA STACK

- **Thao tác: Pop** xóa 1 phần tử từ đỉnh Stack.

```
    else //Nếu chỉ có 1 node
    {
        Node* pDel = s.Head;
        s.Head = s.Head->pNext;
        delete pDel;
        if (s.Head == NULL)
            s.Head = s.Tail = NULL;
    }
    s.size--;
    return true;
}
```

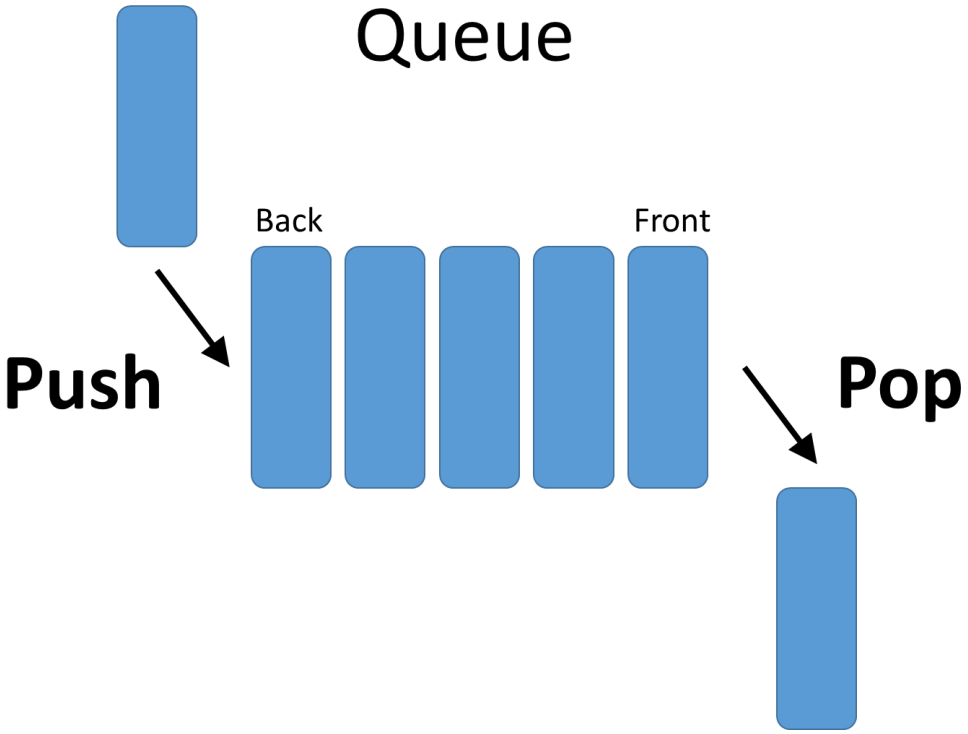
CÁC THAO TÁC CỦA STACK

- **Thao tác: Top** Xem phần tử đầu Stack (không xoá)

```
//Lấy phần tử ở cuối
int Top(const Stack s)
{
    if (IsEmpty(s) == true)
        return INT_MIN;
    int topitem = s.Tail->data;
    return topitem;
}
```

GIỚI THIỆU VỀ HÀNG ĐỢI

Định nghĩa: là một cấu trúc dữ liệu tuân thủ theo nguyên tắc vào trước ra trước (First In First Out).



CÁC THAO TÁC CỦA QUEUE

- **InitQueue:** Khởi tạo Queue rỗng.
- **IsEmpty:** Kiểm tra Queue rỗng.
- **IsFull:** Kiểm tra Queue đầy.
- **Size:** Số lượng phần tử trong Queue.
- **Push:** Thêm 1 phần tử vào đỉnh Queue.
- **Pop:** Lấy ra 1 phần tử từ đỉnh Queue.
- **Front:** Xem phần tử đầu Queue (không xoá)

CÀI ĐẶT QUEUE BẰNG MẢNG 1 CHIỀU

CÁC THAO TÁC CỦA QUEUE

Thao tác: Khai báo cấu trúc Queue.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Queue
{
    int* list;
    int max;
    int size;
};
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: InitQueue Khởi tạo Queue rỗng.

```
bool InitQueue(Queue& q, int n)
{
    q.list = new int[n];
    if (q.list == NULL)
        return false;
    q.max = n;
    q.size = 0;
    return true;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: IsEmpty Kiểm tra Queue rỗng.

```
bool IsEmpty(const Queue& q)
{
    if (q.size == 0)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: IsFull Kiểm tra Queue đầy.

```
bool IsFull(const Queue& q)
{
    if (q.size == q.max)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: Push Thêm 1 phần tử vào đỉnh Queue (thêm cuối)

```
bool Push(Queue& q, int x)
{
    if (IsFull(q) == true)
        return false;
    q.list[q.size] = x;
    q.size++;
    return true;
}
```

CÁC THAO TÁC CỦA QUEUE

- **Thao tác: Pop** xóa 1 phần tử từ đỉnh Queue (xóa đầu)

```
bool Pop(Queue& q)
{
    if (IsEmpty(q) == true)
        return false;
    for (int i = 0; i < q.size; i++)
    {
        q.list[i] = q.list[i + 1];
    }
    q.size--;
    return true;
}
```

CÁC THAO TÁC CỦA QUEUE

- **Thao tác: Front** Xem phần tử đầu Queue (không xoá)

```
int Front(const Queue q)
{
    if (IsEmpty(q) == true)
        return INT_MIN;
    int topitem = q.list[0];
    return topitem;
}
```

CÀI ĐẶT QUEUE BẰNG DANH SÁCH LIÊN KẾT

CÁC THAO TÁC CỦA QUEUE

Thao tác: Khai báo cấu trúc Queue.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
struct Node
{
    int data;
    struct Node* pNext;
};
struct Queue
{
    Node* Head;
    Node* Tail;
    int max;
    int size;
};
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: InitQueue Khởi tạo Queue rỗng.

```
void InitQueue(Queue& q, int n)
{
    q.Head = NULL;
    q.Tail = NULL;
    q.max = n;
    q.size = 0;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: IsEmpty Kiểm tra Queue rỗng.

```
bool IsEmpty(const Queue& q)
{
    if (q.size == 0)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: IsFull Kiểm tra Queue đầy.

```
bool IsFull(const Queue& q)
{
    if (q.size == q.max)
        return true;
    return false;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: CreateNode tạo một node mới.

```
Node* CreateNode(int x)
{
    Node* newNode = new Node;
    if (newNode == NULL)
    {
        cout << "Khong du vung nho";
        return NULL;
    }
    newNode->data = x;
    newNode->pNext = NULL;
    return newNode;
}
```

CÁC THAO TÁC CỦA QUEUE

Thao tác: Push Thêm 1 phần tử vào đỉnh Queue.

```
bool Push(Queue& q, int x)
{
    if (IsFull(q) == true)
        return false;
    else
    {
        if (IsEmpty(q) == true)
        {
            Node* newnode = CreateNode(x);
            q.Head = q.Tail = newnode;
        }
        else //Tương đương với hàm thêm cuối
        {
            Node* newnode = CreateNode(x);
            q.Tail->pNext = newnode;
            q.Tail = newnode;
        }
        q.size++;
    }
    return true;
}
```

CÁC THAO TÁC CỦA QUEUE

- **Thao tác: Pop** xóa 1 phần tử từ đỉnh Queue (xóa đầu)

```
int Pop(Queue& q)
{
    if (IsEmpty(q) == true)
        return false;
    Node* pDel = q.Head;
    q.Head = q.Head->pNext;
    delete pDel;
    if (q.Head == NULL)
        q.Head = q.Tail = NULL;
    q.size--;
    return true;
}
```

CÁC THAO TÁC CỦA QUEUE

- **Thao tác: Front** Xem phần tử đầu Queue (không xoá)

```
int Front(const Queue q)
{
    if (IsEmpty(q) == true)
        return INT_MIN;
    int topitem = q.Head->data;
    return topitem;
}
```

Hỏi đáp