Chương2

Cấu trúc dữ liệu động

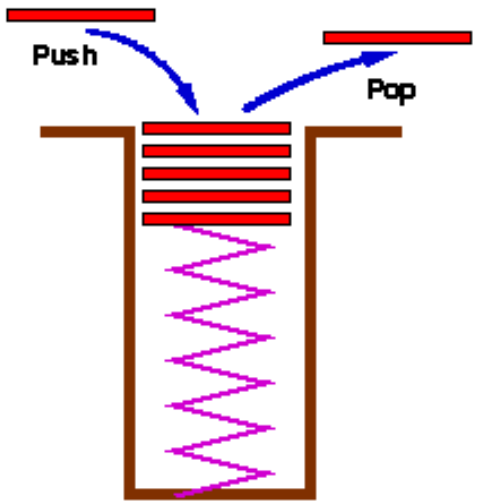
Nội dung

1. Ngăn xếp - Stack
2. Hàng đợi - Queue

Ngăn xếp _ Stack

- Ngăn xếp thường được sử dụng để lưu trữ dữ liệu tạm thời trong quá trình chờ xử lý theo nguyên tắc: **vào sau ra trước (Last In First Out - LIFO)**
- **Khai báo Cấu trúc dữ liệu (dùng xâu đơn)**

```
typedef <Định nghĩa kiểu T>;
typedef struct Node
{       T  Data;
        struct Node *Next; //Con trỏ tới nút kế
} NodeType;
typedef NodeType *StackPtr;
```



- **Khai báo con trỏ đầu Stack**

```
StackPtr  Stack;
```

Ngăn xếp - Stack

Các thao tác trên Stack (dùng xâu đơn)

- **Tạo Stack Rỗng:** Stack = NULL;
- **Kiểm tra Ngăn xếp Rỗng:** if (Stack == NULL)..
- **Thêm 1 phần tử X vào đầu Stack:**

```
void Push(DataType X , StackPtr &Stack )
{
    NodePtr  P;
    P = CreateNode(x);
    P->Next = Stack; /*InsertFirst(P, Stack);*/
    Stack = P;
}
```

Các thao tác trên Stack (dùng xâu đơn)

- **Lấy phần tử ở đỉnh Stack**

```
KieuT  Pop(StackPtr &Stack)
{   DataType  x;
    if (Stack!=NULL)
    {       x = (Stack)->Data; /*Xoa nut dau*/
            P = Stack;
            Stack = P->Next;
            free(P);
            return x;
    }
}
```

Ngăn xếp _ Stack

- **Khai báo Cấu trúc dữ liệu (dùng mảng)**
  - #define MaxSize 100 /*Kích thước Stack*/
  - typedef <Định nghiã kiểu T >
- **Khai báo kiểu mảng:**
  - typedef KiểuT StackArray[MaxSize];
- **Khai báo một Stack:**
  - StackArray Stack; int top; //chỉ mục phần tử đầu Stack

Các thao tác trên Stack (dùng mảng)

- **Khởi tạo 1 Stack rỗng:** top = -1
- **Kiểm tra ngăn xếp rỗng:** if (top == -1)...
- **Kiểm tra ngăn xếp đầy:** if(top == MaxSize-1)...
- **Thêm 1 phần tử có nội dung x vào đầu Stack:**

```
void Push(KieuT x, StackArray Stack, int &top)
{
    if (top < MaxSize-1)
    {       top++;  Stack[top]= x;
    }
}
```

Các thao tác trên Stack (dùng mảng)

- **Lấy phần tử ở đỉnh Stack**

```
KieuT Pop(StackArray Stack, int top)
{
    KieuT  Item;
    if (top != -1)
    {       Item = Stack[top];  top--;
            return Item;
    }
}
```

Ứng dụng của Stack

- Chuyển đổi các hệ thống số
  - Thập phân → nhị phân
  - Nhị phân → thập phân
  - ....
- Xử lý biểu thức hậu tố
  - Chuyển đổi biểu thức ngoặc toàn phần sang biểu thức tiền tố, trung tố, hậu tố
  - Ước lượng giá trị các biểu thức
  - ...
- ...

Hàng đợi _ Queue

- Loại danh sách này có hành vi giống như việc xếp hàng chờ mua vé, với qui tắc **Đến trước - Mua trước. (First in First Out - FIFO)**

  Ví dụ: Bộ đệm bàn phím, tổ chức công việc chờ in trong Print Manager của Windows
- Hàng đợi là một kiểu danh sách đặc biệt có
  - Các thao tác chèn thêm dữ liệu đều thực hiện ở cuối danh sách
  - Các thao tác lấy dữ liệu được thực hiện ở đầu danh sách.

Các thao tác trên hàng đợi

- Khai báo hang đợi
- Khởi tạo hàng đợi rỗng
- Kiểm tra hàng đợi rỗng
- Chèn dữ liệu X vào cuối hàng đợi
- Lấy dữ liệu từ đầu hàng đợi

Hàng đợi _ Queue

- **Khai báo Cấu trúc dữ liệu (dùng xâu đơn)**

```
typedef        <Định nghĩa kiểu T>;
typedef struct Node
{        T  Data;
         struct Node *Next; //Con trỏ tới nút kế
} NodeType;
typedef NodeType *QueuePtr;
```

- **Khai báo con trỏ**

```
QueuePtr  Head, Tail;
```

Các thao tác trên Queue (dùng xâu đơn)

- **Khởi tạo hàng đợi rỗng:** Head = NULL; Tail = NULL;
- **Kiểm tra hàng đợi rỗng:** if (Head == NULL)...
- **Chèn dữ liệu X vào cuối hàng đợi:**

```
void Push( KieuT x, QueuePtr &Head, QueuePtr &Tail )
{   QueuePtr P;
    P = CreateNode(x);
    if (Head == NULL){  Head = P; Tail = Head;      }
    else   {          Tail->Next = P; Tail = P;  }
}
```

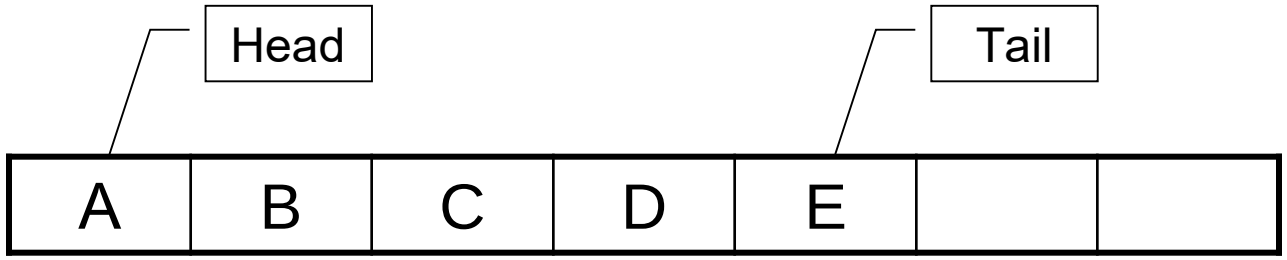
Các thao tác trên Queue (dùng xâu đơn)

- **Lấy dữ liệu từ đầu hàng đợi**

```
KieuT Pop( QueuePtr  &Head, QueuePtr &Tail)
{   QueuePtr P; KieuT x;
    if (Head != NULL)
    {       x = Head->Data;
            P = Head;                /* DeleteFirst(Head);*/
            Head = P->Next;
            free(P);
            If (Head == NULL) Tail = NULL;
    }
    return x;
}
```

Cài đặt Queue dùng mảng

- Sử dụng kỹ thuật xác định chỉ số vòng tròn để định vị trí đầu và cuối hàng đợi.



- Head là vị trí phần tử đầu hàng đợi. Tail là vị trí phần tử cuối hàng đợi
  - Hàng đợi rỗng: Head = Tail
  - Vị trí đầu mới = (Head + 1) mod Maxsize
  - Vị trí cuối mới = (Tail + 1) mod Maxsize
  - Hàng đợi đầy: Vị trí cuối mới = Head

Cài đặt Queue dùng mảng

- **Khai báo kích thước Queue**
  - **#define** MaxSize 100
  - typedef /* khai báo kiểu T*/
- **Khai báo cấu trúc Queue**

```
typedef struct
{           int Head, Tail;
            KiểuT  Node[MaxSize] ;
} QueueType;
QueueType  Queue ;
```

Các thao tác trên Queue (dùng mảng)

- **Khởi Tạo Queue rỗng:**

```
void CreateQ(QueueType &queue)
{    queue.Head = 0;
     queue.Tail = 0;
}
```

- **Kiểm tra hàng đợi rỗng: Head == Tail**

```
int EmptyQ(QueueType  queue )
{
     return (queue.Head == queue.Tail ? 1 : 0));
}
```

Các thao tác trên Queue (dùng mảng)

- **Thêm phần tử vào cuối hàng đợi:**

```
void AddQ(KieuT item, Queuetype &q)
{   int Vitri;
    Vitri = (q.Tail + 1)% maxsize;
    if (Vitri == q.Head)
            printf("\nHang đoi da day");  /*Day hang doi*/
    else
    {       q.Node[Tail]=Item;
            q.Tail = Vitri;
    }
}
```

Các thao tác trên Queue (dùng mảng)

- **Lấy ra 1 phần tử ở đầu hàng đợi:**

```
KieuT GetQ(QueueType  &q)
{   KieuT   Item;
    int Vitri;
    if ( ! EmptyQ(q))
    {       Vitri = (q.Head + 1) % MaxSize;
            Item  = q.Node[Vitri];
            q.Head = Vitri;
            return Item;
    }
}
```