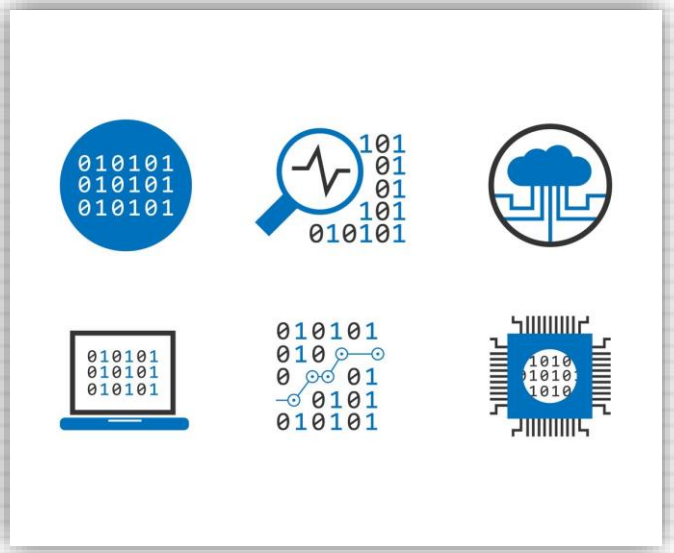
Đại Học Quốc Gia Tp. Hồ Chí Minh
Trường ĐH Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin

KỸ THUẬT LẬP TRÌNH



CHƯƠNG 4: DANH SÁCH LIÊN KẾT (LINKED LIST)

GV: Phạm Nguyễn Sơn Tùng
Email: pnstung@fit.hcmus.edu.vn

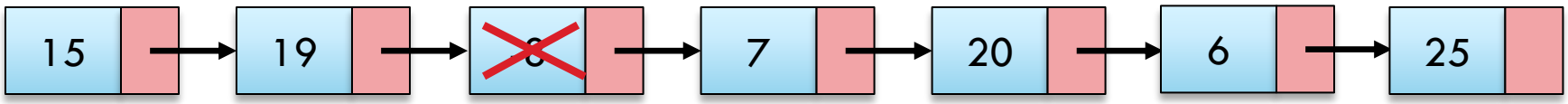
ĐẶT VẤN ĐỀ

Khi bạn sử dụng mảng một chiều, mọi thao tác đều ổn, cho đến khi bạn sử dụng thao tác:

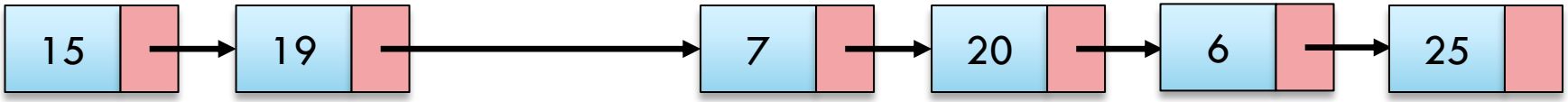
Dời các phần tử phía sau vị trí 2 lên một đơn vị.

ĐẶT VẤN ĐỀ

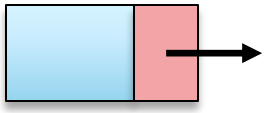
Sử dụng một cấu trúc dữ liệu khác để giảm bớt số lần thao tác khi thêm xóa một phần tử trong dãy.



Chỉ cần xóa phần tử cần xóa và trỏ lại con trỏ phần tử kia.



MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

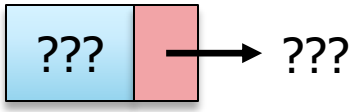


Mỗi một phần tử gọi là Node, mỗi node có 2 trường thông tin:

- Giá trị (chứa dữ liệu của node).
- Con trỏ (dùng để liên kết với phần tử tiếp theo trong danh sách).

Nếu con trỏ không trỏ đến phần tử nào thì trỏ đến NULL.

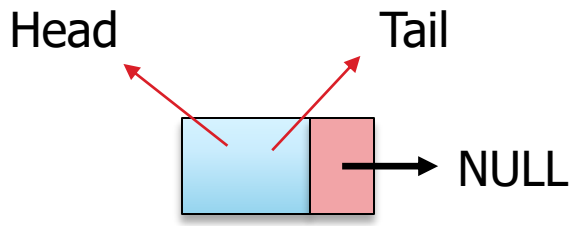
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




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

**** Lưu ý: data ở đây có thể là một số nguyên, nhưng cũng có thể là một kiểu dữ liệu khác hoặc một cấu trúc (struct) gồm nhiều thành phần.*

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




```
struct Linked_List
{
    Node* Head;
    Node* Tail;
};
```

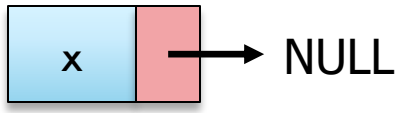
```
void Khoitao(Linked_List& list)
{
    list.Head = list.Tail = NULL;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Kiểm tra danh sách liên kết có rỗng hay không.

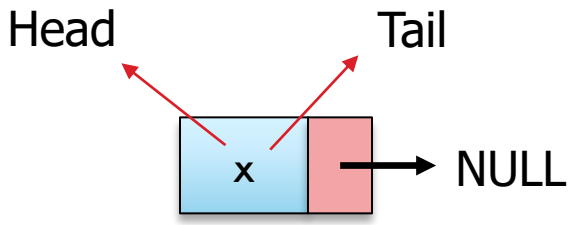
```
bool KiemTraDSRong(Linked_List list)
{
    return (list.Head == NULL && list.Tail == NULL);
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT



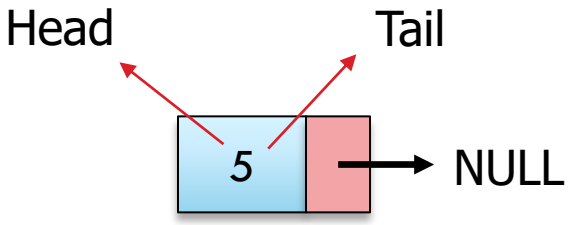

```
Node* TaoNode(int x)
{
    Node* newNode = new Node;
    if (newNode == NULL)
    {
        cout << "Khong du vung nho";
        return NULL;
    }
    newNode->data = x;
    newNode->pNext = NULL;
    return newNode;
}
```

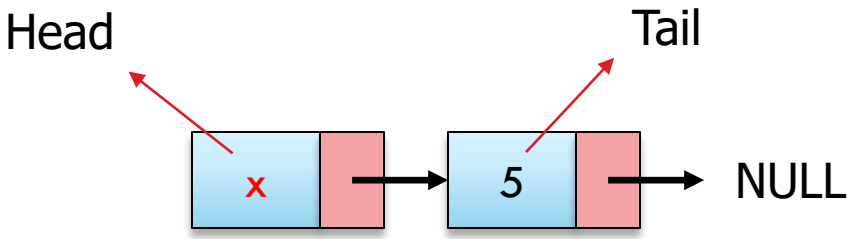
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




```
void ThemVaoDSRong(Linked_List& list, int x)
{
    Node* NewNode = TaoNode(x);
    list.Head = list.Tail = NewNode;
}
```

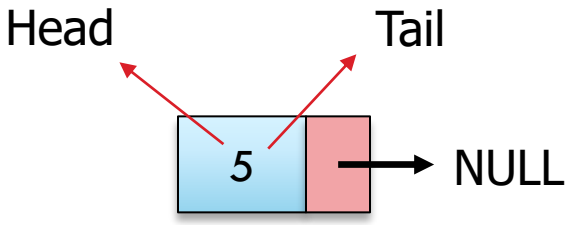
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

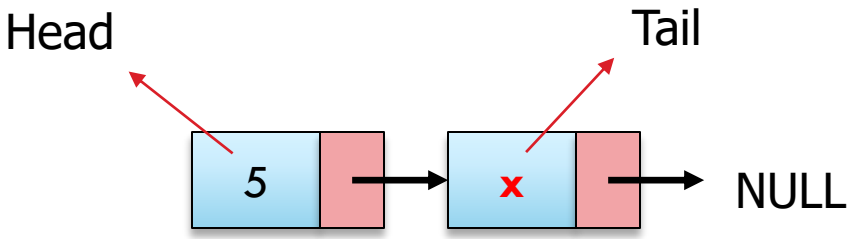





```
void ThemVaoDau(Linked_List& list, int x)
{
    Node* NewNode = TaoNode(x);
    NewNode->pNext = list.Head;
    list.Head = NewNode;
}
```

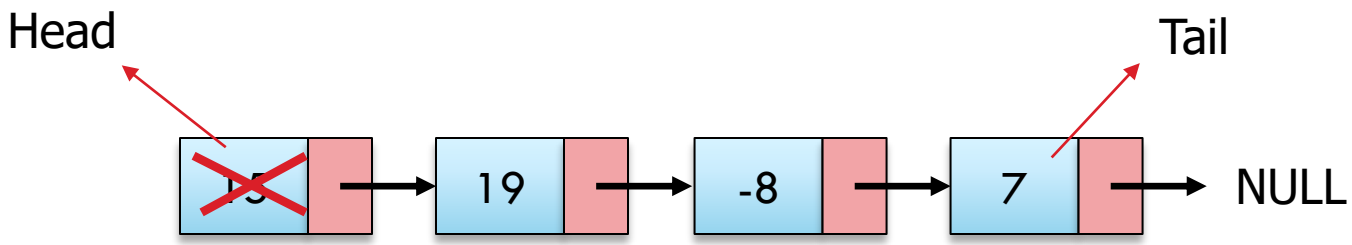
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT





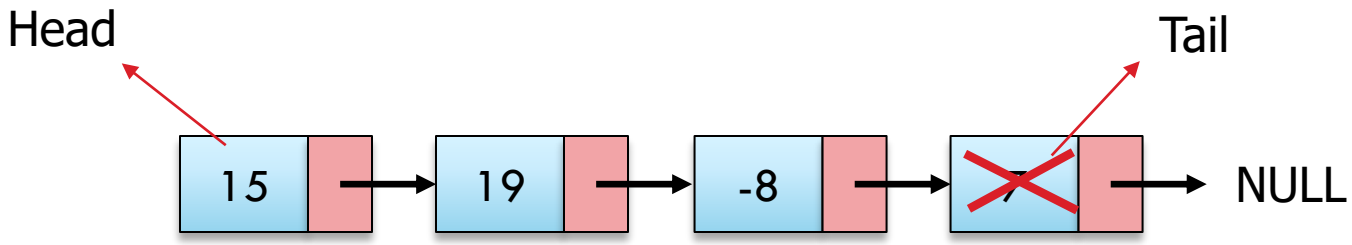

```
void ThemVaoCuoi(Linked_List& list, int x)
{
    Node* NewNode = TaoNode(x);
    list.Tail->pNext = NewNode;
    list.Tail = NewNode;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




```
void XoaDau(Linked_List& list)
{
    if (list.Head!=NULL)
    {
        Node* pDel = list.Head;
        list.Head = list.Head->pNext;
        delete pDel;
        if (list.Head == NULL)
            list.Head = list.Tail = NULL;
    }
}
```

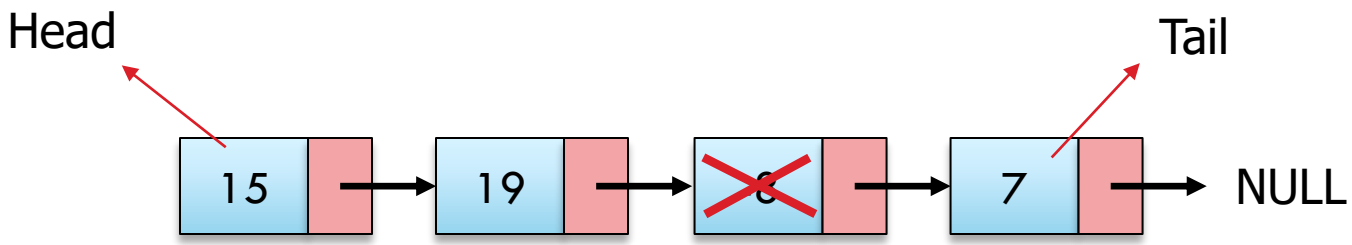
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




```
void XoaCuoi(Linked_List& list)
{
    if (list.Tail)
    {
        if (list.Tail != list.Head)
        {
            Node* pDel = list.Tail;
            Node* pCur = list.Head;
            while (pCur->pNext != list.Tail)
                pCur = pCur->pNext;
            pCur->pNext = NULL;
            list.Tail = pCur;
            delete pDel;
        }
        else
            XoaDau(list);

    }
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT




```
void XoaGiua(Linked_List& list, Node* pDel)
{
    Node* pCur = list.Head;
    while (pCur->pNext != pDel)
        pCur = pCur->pNext;
    pCur->pNext = pDel->pNext;
    delete pDel;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

```
void XoaNode(Linked_List& list, int x)
{
    Node* pDel = list.Head;
    while (pDel->data != x && pDel->pNext != NULL)
    {
        pDel = pDel->pNext;
        if (pDel->pNext == NULL)
            return;
    }
    if (pDel == list.Head)
        XoaDau(list);
    else if (pDel == list.Tail)
        XoaCuoi(list);
    else if (pDel != NULL)
        XoaGiua(list, pDel);
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Dùng vòng lặp **while** xuất ra toàn bộ danh sách liên kết.

```
void XuatDanhSach(Linked_List list)
{
    Node* pCur = list.Head;
    while (pCur != NULL)
    {
        cout << pCur->data << " ";
        pCur = pCur->pNext;
    }
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Dùng vòng lặp **for** xuất ra toàn bộ danh sách liên kết.

```
void XuatDanhSach2(Linked_List list)
{
    for (Node* pCur = list.Head; pCur!=NULL; pCur = pCur->pNext)
    {
        cout << pCur->data << " ";
    }
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Tìm phần tử lớn nhất trong danh sách liên kết.

```
Node* TimLonNhat(Linked_List list)
{
    Node* pmax = list.Head;
    for (Node* p = list.Head->pNext; p ;p = p->pNext)
    {
        if (p->data > pmax->data)
        pmax = p;
    }
    return pmax;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Tìm phần tử chẵn đầu tiên.

```
Node* TimChanDau(Linked_List list)
{
    Node* p = list.Head;
    while (p != NULL)
    {
        if (p->data % 2 == 0)
            break;
        p = p->pNext;
    }
    return p;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Thêm một phần tử vào sau Node p.

```
void ThemVaoSauP(Linked_List& list, int x, Node* p)
{
    Node* NewNode = TaoNode(x);
    NewNode->pNext = p->pNext;
    p->pNext = NewNode;
}
```

MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Theo bạn hàm này sẽ cho ra kết quả là gì.

```
void XuLy(Linked_List &list)
{
     Node *prev = NULL;
     struct Node *current = list.Head;
     struct Node *next;
     while (current != NULL)
    {
         next = current->pNext;
         current->pNext = prev;
         prev = current;
         current = next;
    }
    list.Head = prev;
}
```

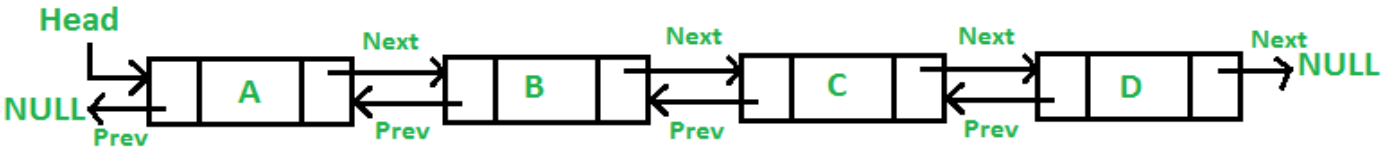
MỘT SỐ THAO TÁC VỚI DANH SÁCH LIÊN KẾT

Theo bạn hàm này sẽ cho ra kết quả là gì. Nếu cho vào list.Head (2, 4, 6, 8, 10, 12)

```
void XuLy2(Node *list)
{
    if (list == NULL)
        return;
    cout << list->data << " ";
    if (list->pNext != NULL)
        XuLy2(list->pNext->pNext);
     cout << list->data << " ";
}
```

MỘT SỐ LOẠI DANH SÁCH LIÊN KẾT KHÁC

- Danh sách liên kết đơn. (singly linked list)
- Danh sách liên kết đôi. (doubly linked list)



- Danh sách liên kết vòng. (circular linked list)