

BT#04: So sánh các ưu khuyết điểm của

a/ Linked List có và không có con trỏ Tail luôn lưu địa chỉ Node cuối.

b/ Linked List có và không có Header Node.

a) So sánh Linked List có và không có con trỏ Tail luôn lưu địa chỉ Node cuối

- Linked List không có con trỏ Tail:

+ Ưu điểm:

- **Đơn giản hơn:** Cấu trúc chỉ cần quản lý con trỏ Head (đầu danh sách).
- **Tiết kiệm bộ nhớ:** Không tốn thêm bộ nhớ cho một con trỏ Tail.

+ Khuyết điểm:

- **Thêm vào cuối chậm:** Thao tác thêm một nút vào cuối danh sách (append) đòi hỏi phải duyệt từ đầu đến cuối danh sách để tìm nút cuối cùng. Độ phức tạp thời gian là $O(n)$, với n là số lượng nút.
- **Truy cập nút cuối chậm:** Lấy thông tin hoặc địa chỉ của nút cuối cùng cũng yêu cầu duyệt toàn bộ danh sách ($O(n)$).

- Linked List có con trỏ Tail:

+ Ưu điểm:

- **Thêm vào cuối nhanh:** Thao tác thêm một nút vào cuối danh sách (append) trở nên rất hiệu quả, chỉ cần cập nhật con trỏ next của nút Tail hiện tại và cập nhật Tail trỏ đến nút mới. Độ phức tạp thời gian là $O(1)$.
- **Truy cập nút cuối nhanh:** Lấy thông tin hoặc địa chỉ nút cuối cùng rất nhanh ($O(1)$) vì đã có con trỏ Tail.

+ Khuyết điểm:

- **Phức tạp hơn:** Cần quản lý thêm con trỏ Tail, phải cập nhật Tail đúng cách trong các thao tác thêm/xóa (đặc biệt là khi xóa nút cuối cùng hoặc khi danh sách trở nên rỗng).
- **Tốn thêm bộ nhớ:** Cần thêm không gian bộ nhớ để lưu trữ con trỏ Tail.

b) So sánh Linked List có và không có Header Node

- Linked List không có Header Node:

+ Ưu điểm:

- **Tiết kiệm bộ nhớ:** Không tốn bộ nhớ cho một nút giả không chứa dữ liệu.
- **Tự nhiên hơn:** Cấu trúc có vẻ trực quan hơn khi con trỏ Head trỏ trực tiếp đến nút dữ liệu đầu tiên (hoặc là NULL nếu danh sách rỗng).

+ Khuyết điểm:

- **Xử lý trường hợp đặc biệt:** Các thao tác chèn hoặc xóa ở đầu danh sách (nút đầu tiên) thường yêu cầu xử lý đặc biệt. Ví dụ, khi chèn vào đầu, phải cập nhật con trỏ Head của danh sách. Khi xóa nút đầu tiên, cũng phải cập nhật Head. Điều này làm cho mã nguồn có thể có nhiều câu lệnh điều kiện if hơn để xử lý các trường hợp biên này.

- Linked List có Header Node:

+ Ưu điểm:

- **Đơn giản hóa thuật toán:** Các thao tác chèn và xóa (đặc biệt là ở đầu danh sách hoặc khi danh sách rỗng) trở nên nhất quán hơn. Mọi thao tác chèn/xóa đều được coi là chèn/xóa *sau* một nút nào đó (kể cả chèn vào "đầu" danh sách thực tế thì cũng là chèn sau Header Node). Điều này loại bỏ các trường hợp đặc biệt cần xử lý cho nút đầu tiên, giúp mã nguồn gọn gàng và ít lỗi hơn.
- **Danh sách không bao giờ rỗng:** Luôn có ít nhất Header Node, giúp tránh lỗi con trỏ NULL trong một số trường hợp.

+ Khuyết điểm:

- **Tốn thêm bộ nhớ:** Cần thêm không gian cho một Header Node không chứa dữ liệu hữu ích.
- **Kém trực quan ban đầu:** Khái niệm về một nút giả ở đầu có thể hơi khó hiểu lúc ban đầu.
- **Truy cập nút đầu tiên:** Để truy cập nút chứa dữ liệu thực tế đầu tiên, ta cần dùng `head->next`.

BT#05: Viết hàm chuyển danh sách sinh viên đang lưu trong một mảng động vào

- danh sách liên kết đơn không có con trỏ cuối và ngược lại.**
- danh sách liên kết đơn luôn có con trỏ cuối và ngược lại.**
- danh sách liên kết kép và ngược lại.**
- danh sách liên kết vòng và ngược lại.**
- danh sách liên kết kép vòng và ngược lại.**

a) danh sách liên kết đơn không có con trỏ cuối và ngược lại.

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

struct SinhVienNode {
    char *data;
    SinhVienNode *next;
};

void ChuyenArraySangLinkedList(char **SinhVienArray,
SinhVienNode *&SinhVienLinkedList, int n) {
    SinhVienLinkedList = NULL;

    for (int i = n - 1; i >= 0; i--) {
        SinhVienNode *newNode = (SinhVienNode
```

```

*)malloc(sizeof(SinhVienNode));
    newNode->data = SinhVienArray[i];
    newNode->next = SinhVienLinkedList;
    SinhVienLinkedList = newNode;
}
}

```

```

void ChuyenLinkedListSangArray(SinhVienNode
*SinhVienLinkedList, char **&SinhVienArray, int n) {
    SinhVienNode *current = SinhVienLinkedList;
    for (int i = 0; i < n; i++) {
        SinhVienArray[i] = current->data;
        current = current->next;
    }
}

```

```

void HienThiThongTinLinkedList(SinhVienNode
*SinhVienLinkedList) {
    SinhVienNode *current = SinhVienLinkedList;
    while (current) {
        cout << current->data << " ";
        current = current->next;
    }
}

```

```

int main() {
    int n;
    cin >> n;
    char **SinhVienArray = (char**)malloc(n * sizeof(char*));
    for (int i = 0; i < n; i++) {
        SinhVienArray[i] = (char *)malloc(1000);
        cin >> SinhVienArray[i];
    }
}

```

```

cout << "Array: ";
for (int i = 0; i < n; i++) cout << SinhVienArray[i] << " ";
cout << "\n";

SinhVienNode *SinhVienLinkedList;
ChuyenArraySangLinkedList(SinhVienArray,
SinhVienLinkedList, n);

cout << "Linked List: ";
HienThiThongTinLinkedList(SinhVienLinkedList);
cout << "\n";

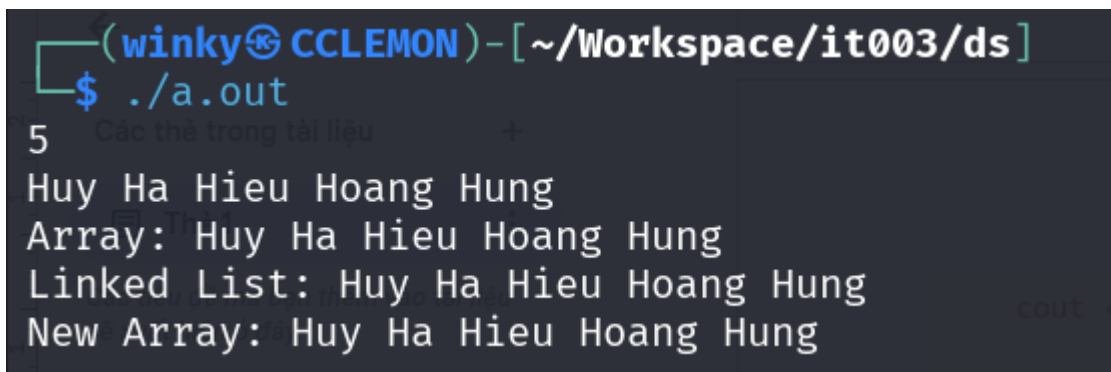
char **NewSinhVienArray = (char**)malloc(n * sizeof(char*));
ChuyenLinkedListSangArray(SinhVienLinkedList,
NewSinhVienArray, n);

cout << "New Array: ";
for (int i = 0; i < n; i++) cout << NewSinhVienArray[i] << " ";

}

```

Kết quả khi chạy :



```

(winky@CCLEMON) - [~/Workspace/it003/ds]
$ ./a.out
5
Huy Ha Hieu Hoang Hung
Array: Huy Ha Hieu Hoang Hung
Linked List: Huy Ha Hieu Hoang Hung
New Array: Huy Ha Hieu Hoang Hung

```

b) danh sách liên kết đơn luôn có con trỏ cuối và ngược lại.

```
struct SinhVienNode {  
    char *data;  
    SinhVienNode *next;  
};
```

```
void ChuyenArraySangLinkedList(char **SinhVienArray,  
SinhVienNode *&SinhVienLinkedList, int n) {
```

```
    SinhVienLinkedList = NULL;  
    SinhVienNode *tail = NULL;
```

```
    for (int i = 0; i < n; i++) {  
        SinhVienNode *newNode = (SinhVienNode  
*)malloc(sizeof(SinhVienNode));  
        newNode->data = SinhVienArray[i];  
        newNode->next = NULL;
```

```
        if (SinhVienLinkedList == NULL) {  
            SinhVienLinkedList = newNode;  
            tail = newNode;  
        } else {  
            tail->next = newNode;  
            tail = newNode;  
        }  
    }
```

```
}
```

```
void ChuyenLinkedListSangArray(SinhVienNode  
*SinhVienLinkedList, char **&SinhVienArray, int n) {
```

```
    SinhVienNode *current = SinhVienLinkedList;  
    for (int i = 0; i < n; i++) {  
        SinhVienArray[i] = current->data;  
        current = current->next;
```

```
}
```

```

}

void HienThiThongTinLinkedList(SinhVienNode
*SinhVienLinkedList) {
    SinhVienNode *current = SinhVienLinkedList;
    while (current) {
        cout << current->data << " ";
        current = current->next;
    }
}

```

c) danh sách liên kết kép và ngược lại.

```

struct SinhVienNode {
    char *data;
    SinhVienNode *prev, *next;
};

void ChuyenArraySangLinkedList(char
**SinhVienArray, SinhVienNode *&head,
SinhVienNode *&tail, int n) {
    head = tail = NULL;
    for (int i = 0; i < n; i++) {
        SinhVienNode *newNode = (SinhVienNode
*)malloc(sizeof(SinhVienNode));
        newNode->data = SinhVienArray[i];
        newNode->next = NULL;
        newNode->prev = tail;
        if (!head) head = newNode;
        else tail->next = newNode;
    }
}

```

```

        tail = newNode;
    }
}

void ChuyenLinkedListSangArray(SinhVienNode
*head, char **&SinhVienArray, int n) {
    SinhVienNode *current = head;
    for (int i = 0; i < n; i++, current =
current->next)
        SinhVienArray[i] = current->data;
}

void HienThiThongTinLinkedList(SinhVienNode
*head) {
    for (SinhVienNode *current = head; current;
current = current->next)
        cout << current->data << " ";
}

```

d) danh sách liên kết vòng và ngược lại.

```

struct SinhVienNode {
    char *data;
    SinhVienNode *next;
};

void ChuyenArraySangLinkedList(char **SinhVienArray,
SinhVienNode *&head, int n) {
    head = NULL;
    SinhVienNode *tail = NULL;
}

```



```

    for (int i = 0; i < n; i++) {
        SinhVienNode *newNode = (SinhVienNode
*)malloc(sizeof(SinhVienNode));
        newNode->data = SinhVienArray[i];
        if (!head) {
            head = newNode;
            newNode->next = head;
        } else {
            newNode->next = head;
            tail->next = newNode;
        }
        tail = newNode;
    }
}

void ChuyenLinkedListSangArray(SinhVienNode *head, char
**&SinhVienArray, int n) {
    SinhVienNode *current = head;
    for (int i = 0; i < n; i++, current = current->next)
        SinhVienArray[i] = current->data;
}

void HienThiThongTinLinkedList(SinhVienNode *head, int n) {
    SinhVienNode *current = head;
    for (int i = 0; i < n; i++, current = current->next)
        cout << current->data << " ";
}

```

e) danh sách liên kết kép vòng và ngược lại.

```

struct SinhVienNode {
    char *data;

```

```

    SinhVienNode *prev, *next;
};

void ChuyenArraySangLinkedList(char **SinhVienArray,
SinhVienNode *&head, int n) {
    head = NULL;
    SinhVienNode *tail = NULL;
    for (int i = 0; i < n; i++) {
        SinhVienNode *newNode = (SinhVienNode
*)malloc(sizeof(SinhVienNode));
        newNode->data = SinhVienArray[i];
        if (!head) {
            head = newNode;
            newNode->next = newNode;
            newNode->prev = newNode;
        } else {
            newNode->next = head;
            newNode->prev = tail;
            tail->next = newNode;
            head->prev = newNode;
        }
        tail = newNode;
    }
}

void ChuyenLinkedListSangArray(SinhVienNode *head, char
**&SinhVienArray, int n) {
    SinhVienNode *current = head;
    for (int i = 0; i < n; i++, current = current->next)
        SinhVienArray[i] = current->data;
}

```

BT#06: Viết hàm thêm 1 phần tử vào

a) mảng động đang có thứ tự.

b) chuỗi đơn có thứ tự + có và không có con trỏ cuối (2 hàm khác nhau).

c) chuỗi đơn có thứ tự + có Header node + có và không có con trỏ cuối (2 hàm khác nhau).

d) chuỗi đơn có thứ tự + có Header node và luôn có địa chỉ node cuối nằm trong các byte đầu của trường dữ liệu trong Header node.

a) mảng động đang có thứ tự.

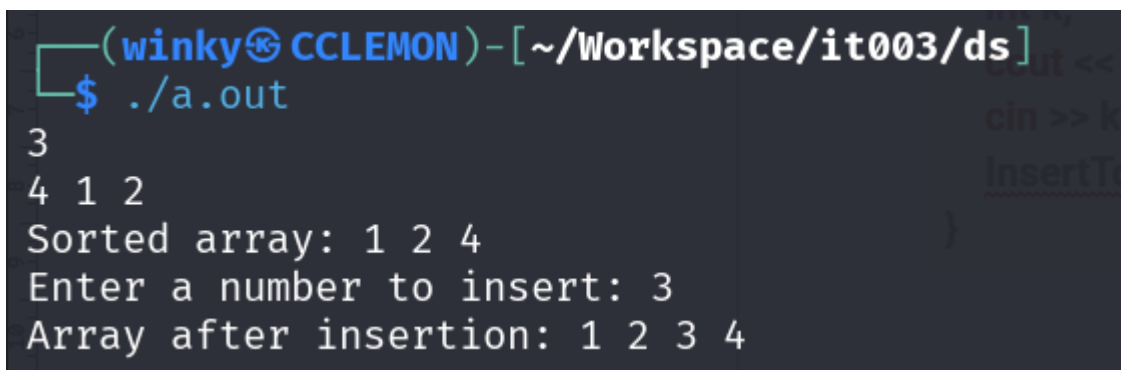
```
#include<iostream>
#include<algorithm>
using namespace std;
void InsertToArray(int *array, int n, int k){
    array = (int*)realloc(array, (n + 1) * sizeof(int));
    int i;
    for (i = n - 1; (i >= 0 && array[i] > k); i--){
        array[i + 1] = array[i];
    }
    array[i + 1] = k;
    cout << "New array: ";
    for (int j = 0; j <= n; j++){
        cout << array[j] << " ";
    }
}
int main(){
    int n; cin >> n;
    int *array = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++){
        cin >> array[i];
    }
}
```

```

    }
    sort(array, array + n);
    cout << "Sorted array: ";
    for (int i = 0; i < n; i++){
        cout << array[i] << " ";
    }
    cout << "\n";
    int k;
    cout << "Enter a number to insert: ";
    cin >> k;
    InsertToArray(array, n, k);
}

```

Kết quả sau khi chạy:



```

(winky CCLEMON) - [~/Workspace/it003/ds]
$ ./a.out
3
4 1 2
Sorted array: 1 2 4
Enter a number to insert: 3
Array after insertion: 1 2 3 4

```

b) xâu đơn có thứ tự + có và không có con trỏ cuối (2 hàm khác nhau)

Không có con trỏ cuối :

```

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

struct Node{
    int data;
    Node *next = nullptr;
};

```

```

void ThemVaoLinkedList(Node *&head, int x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node *tmp = head;
        while (tmp->next != nullptr) {
            tmp = tmp->next;
        }
        tmp->next = newNode;
    }
}

```

```

void HienThiThongTinLinkedList(Node *head) {
    Node *tmp = head;
    cout << "Linked list: ";
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
}

```

```

void ThemVaoLinkedListCoThuTu(Node *&head, int x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr || head->data >= x) {
        newNode->next = head;
        head = newNode;
    }
}

```

```

    }
    else {
        Node *tmp = head;
        while (tmp->next != nullptr && tmp->next->data < x) {
            tmp = tmp->next;
        }
        newNode->next = tmp->next;
        tmp->next = newNode;
    }
}

int main(){
    Node *head = nullptr;
    int n; cin >> n;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        ThemVaoLinkedList(head, x);
    }
    HienThiThongTinLinkedList(head);
    cout << "\n";
    int k;
    cout << "Enter a number to insert: ";
    cin >> k;
    ThemVaoLinkedListCoThuTu(head, k);
    HienThiThongTinLinkedList(head);
}

```

Kết quả sau khi chạy:

```
(winky@CCLEMON)-[~/Workspace/it003/ds]  
$ ./a.out  
3  
1 2 4  
Linked list: 1 2 4  
Enter a number to insert: 3  
Linked list: 1 2 3 4
```

Có con trở cuối:

```
#include<iostream>  
using namespace std;  
  
struct Node {  
    int data;  
    Node *next = nullptr;  
};  
  
void ThemVaoLinkedList(Node *&head, Node *&tail, int x) {  
    Node *newNode = new Node;  
    newNode->data = x;  
    newNode->next = nullptr;  
  
    if (head == nullptr) {  
        head = newNode;  
        tail = newNode;  
    } else {  
        tail->next = newNode;  
        tail = newNode;  
    }  
}  
  
void HienThiThongTinLinkedList(Node *head) {  
    Node *tmp = head;
```

```

    cout << "Linked List with tail pointer: ";
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
}

void ThemVaoLinkedListCoThuTu(Node *&head, Node *&tail, int
x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr || head->data >= x) {
        newNode->next = head;
        head = newNode;
        if (head->next == nullptr) {
            tail = head;
        }
    } else {
        Node *tmp = head;
        while (tmp->next != nullptr && tmp->next->data < x) {
            tmp = tmp->next;
        }
        newNode->next = tmp->next;
        tmp->next = newNode;

        if (newNode->next == nullptr) {
            tail = newNode;
        }
    }
}

```



```

int main() {
    Node *head = nullptr, *tail = nullptr;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        ThemVaoLinkedList(head, tail, x);
    }

    HienThiThongTinLinkedList(head);
    cout << "\n";

    int k;
    cout << "Enter a number to insert: ";
    cin >> k;
    ThemVaoLinkedListCoThuTu(head, tail, k);
    HienThiThongTinLinkedList(head);
}

```

Kết quả sau khi chạy:

```

(winky@CCLEMON)-[~/Workspace/it003/ds]
$ ./a.out
3
1 2 4
Linked List with tail pointer: 1 2 4
Enter a number to insert: 3
Linked List with tail pointer: 1 2 3 4

```

c) xâu đơn có thứ tự + có Header node + có và không có con trỏ cuối (2 hàm khác nhau).

Không có con trỏ cuối:

```

struct Node {
    int data;

```

```

    Node *next = nullptr;
};

void ThemVaoLinkedList(Node *&head, int x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node *tmp = head;
        while (tmp->next != nullptr) {
            tmp = tmp->next;
        }
        tmp->next = newNode;
    }
}

void HienThiThongTinLinkedList(Node *head) {
    Node *tmp = head;
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
}

void ThemVaoLinkedListCoThuTu(Node *&head, int x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr || head->data >= x) {

```

```

    newNode->next = head;
    head = newNode;
} else {
    Node *tmp = head;
    while (tmp->next != nullptr && tmp->next->data < x) {
        tmp = tmp->next;
    }
    newNode->next = tmp->next;
    tmp->next = newNode;
}
}

```

Có con trỏ cuối:

```

struct Node {
    int data;
    Node *next = nullptr;
};

void ThemVaoLinkedList(Node *&head, Node *&tail, int x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}

void HienThiThongTinLinkedList(Node *head) {

```

```
Node *tmp = head;
while (tmp != nullptr) {
    cout << tmp->data << " ";
    tmp = tmp->next;
}
}
```

```
void ThemVaoLinkedListCoThuTu(Node *&head, Node *&tail, int
x) {
    Node *newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    Node *tmp = head;

    if (head == nullptr || head->data >= x) {
        newNode->next = head;
        head = newNode;
        if (head->next == nullptr) {
            tail = head;
        }
    } else {
        while (tmp->next != nullptr && tmp->next->data < x) {
            tmp = tmp->next;
        }
        newNode->next = tmp->next;
        tmp->next = newNode;
        if (newNode->next == nullptr) {
            tail = newNode;
        }
    }
}
```

d) **xâu đơn có thứ tự + có Header node và luôn có địa chỉ node cuối nằm trong các byte đầu của trường dữ liệu trong Header node.**

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

struct Node {
    int data;
    Node* next = nullptr;
};

struct HeaderNode {
    Node* lastNode;
    Node* next;
};

void HienThiThongTinLinkedList(HeaderNode* header) {
    Node* tmp = header->next;
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
    cout << "\n";
}

void ThemVaoLinkedListCoThuTu(HeaderNode* header, int x) {
    Node* newNode = new Node;
    newNode->data = x;
    newNode->next = nullptr;

    Node* tmp = header->next;
```

```

if (header->next == nullptr || header->next->data >= x) {
    newNode->next = header->next;
    header->next = newNode;
    if (header->next->next == nullptr) {
        header->lastNode = newNode;
    }
} else {
    while (tmp->next != nullptr && tmp->next->data < x) {
        tmp = tmp->next;
    }
    newNode->next = tmp->next;
    tmp->next = newNode;

    if (newNode->next == nullptr) {
        header->lastNode = newNode;
    }
}
}

```

```

int main() {
    HeaderNode* header = new HeaderNode;
    header->lastNode = nullptr;
    header->next = nullptr;

    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        ThemVaoLinkedListCoThuTu(header, x);
    }

```

```

HienThiThongTinLinkedList(header);

```

```
int k;  
cout << "Enter a number to insert: ";  
cin >> k;  
ThemVaoLinkedListCoThuTu(header, k);  
HienThiThongTinLinkedList(header);  
}
```

Source của các file có thể xem tại:

<https://github.com/threalwinky/IT003.P21.CTTN/tree/main/Assignment%2003.01/source>