CẤU TRÚC DỮ LIỆU
MẢNG VS DANH SÁCH LIÊN KẾT

Bùi Tiến Lên

01/01/2017

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

MẢNG

Kiểu dữ liệu mảng

Định nghĩa 1

Mảng (array) là một tập hợp các phần tử $X = \{x_0, ..., x_n\}$ được tổ chức **tuyến tính**

- Các phần tử x_i được lưu trữ liên tiếp nhau
- Các phần tử x_i được truy xuất thông qua các chỉ số

Kiểu dữ liệu mảng (cont.)

Ưu điểm của kiểu dữ liệu mảng

- Đơn giản
- Xử lý nhanh
- Bộ nhớ lưu trữ liên tục
- Số lượng phần tử tương đối cố định

Ứng dụng của mảng

Kiểu dữ liệu mảng rất phù hợp với các đối tượng như vector, hay ma trận. Do đó, nó rất phù hợp với các ứng dụng toán học

Thêm một phần tử vào mảng

1. Di chuyển các phần tử về phía sau một vị trí
2. Sau đó mới chèn phần tử mới vào
3. Vậy chi phí là $O(n)$

Thêm một phần tử vào mảng (cont.)

Chương trình 1: Hàm thêm một phần tử x vào mảng a có n phần tử tại vị trí k

```
1  void Insert(int a[], int &n, int x, int k)
2  {
3      for (int i = n; i > k; i--)
4          a[i] = a[i - 1];
5      a[k] = x;
6      n++;
7  }
```

Minh họa

Ví dụ 1

Một mảng a có 6 phần tử $a = \{1, 2, 4, 3, 8, 5\}$, hãy chèn phần tử 9 vào vị trí có chỉ số 2 của mảng a

1	2	4	3	8	5

- Dời các phần tử từ chỉ số 2 sang phải một đơn vị

1	2	4	4	3	8	5

- Gán giá trị 9 vào phần tử có chỉ số 2

1	2	9	4	3	8	5

Minh họa

Ví dụ 1

Một mảng a có 6 phần tử $a = \{1, 2, 4, 3, 8, 5\}$, hãy chèn phần tử 9 vào vị trí có chỉ số 2 của mảng a

1	2	4	3	8	5

- Dời các phần tử từ chỉ số 2 sang phải một đơn vị

1	2	4	4	3	8	5

- Gán giá trị 9 vào phần tử có chỉ số 2

1	2	9	4	3	8	5

Minh họa

Ví dụ 1

Một mảng a có 6 phần tử $a = \{1, 2, 4, 3, 8, 5\}$, hãy chèn phần tử 9 vào vị trí có chỉ số 2 của mảng a

1	2	4	3	8	5

- Dời các phần tử từ chỉ số 2 sang phải một đơn vị

1	2	4	4	3	8	5

- Gán giá trị 9 vào phần tử có chỉ số 2

1	2	9	4	3	8	5

Xóa một phần tử trong mảng

- Dời các phần tử về phía trước một đơn vị
- Chi phí để xóa một phần tử trong mảng cũng là $O(n)$

Xóa một phần tử trong mảng (cont.)

Chương trình 2: Hàm cài đặt xóa một phần tử tại vị trí k của mảng a có n phần tử

```
1  void Remove(int a[], int &n, int k)
2  {
3      for (int i = k; i < n - 1; i++)
4          a[i] = a[i + 1];
5      n--;
6  }
```

Minh họa

Một mảng a có 6 phần tử $a = \{2, 1, 4, 3, 7, 5\}$, hãy xóa phần tử có chỉ số 1 của mảng a

2	1	4	3	7	5

- Dời các phần tử từ chỉ số 2 sang phải một đơn vị

2	4	3	7	5

Minh họa

Một mảng a có 6 phần tử $a = \{2, 1, 4, 3, 7, 5\}$, hãy xóa phần tử có chỉ số 1 của mảng a

2	1	4	3	7	5

- Dời các phần tử từ chỉ số 2 sang phải một đơn vị

2	4	3	7	5

DANH SÁCH LIÊN KẾT

Vấn đề của mảng

Vấn đề

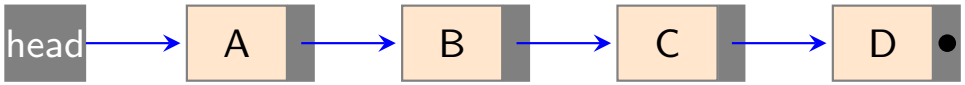
Làm sao có thể thêm (hay xóa) 1 phần tử mà không phải di chuyển các phần tử như trong mảng

- Tách rời các phần tử trong mảng
- Kết dính chúng lại với nhau bằng các "liên kết"

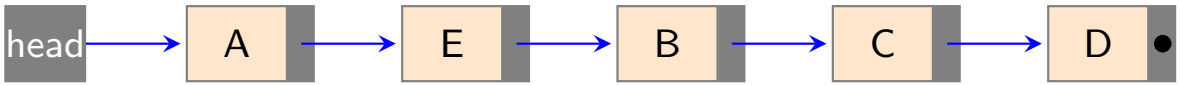
Vấn đề của mảng (cont.)

Ví dụ 2

Một danh sách 4 phần tử {A, B, C, D} có thể biểu diễn bằng một chuỗi các "móc xích" như sau



Chèn một phần tử E vào vị trí của phần tử B thì chuỗi trở thành



Danh sách liên kết

Định nghĩa 2

Danh sách liên kết (linked list) là một tập hợp các phần tử $X = \{x_0, x_1, ..., x_{n-1}\}$ được tổ chức tuyến tính

- Các phần tử x_i được liên kết với các phần tử đứng trước hoặc đứng sau
- Các phần tử x_i không thể truy xuất qua chỉ số
- Các phần tử trong một danh sách liên kết sẽ được gọi là **nút** (node)

Danh sách liên kết (cont.)

Đặc điểm của danh sách liên kết

- Sử dụng con trỏ
- Cấp phát bộ nhớ động
- Dãy tuần tự các nút
- Giữa hai nút có một hay nhiều con trỏ liên kết
- Các nút không cần phải liên tiếp nhau trong bộ nhớ vật lý
- Có thể mở rộng tùy ý
- Thao tác thêm/xóa không cần phải dịch chuyển phần tử

Ứng dụng của danh sách liên kết

Kiểu dữ liệu danh sách liên kết phù hợp với các ứng dụng có dữ liệu không phức tạp và hay bị thay đổi bởi các thao tác như: thêm, xóa

Danh sách liên kết đơn

Định nghĩa 3

Danh sách liên kết đơn (singly linked list) là một tập hợp tuyến tính các phần tử $\{x_0 \rightarrow x_1 \rightarrow ... \rightarrow x_{n-1}\}$

- Mỗi nút liên kết đến nút kế tiếp trong danh sách
- Nút cuối của danh sách không trỏ đến nút nào cả do đó giá trị sẽ là null
- Cần liên kết tới nút đầu danh sách

Danh sách liên kết đơn (cont.)

Kiểu dữ liệu cho một nút trong danh sách liên kết đơn

```
1  template <class T>
2  struct Node
3  {
4      T data;
5    int key;
6      Node *next;
7  };
```

Danh sách liên kết đơn (cont.)

Cài đặt lớp cho một danh sách liên kết

```
1  template <class T>
2  class LinkedList
3  {
4    private:
5      int size;
6      Node<T> *head;
7
8    public:
9      isEmpty(...);
10     insert(...);
11     remove(...);
12     search(...);
13 };
```

Danh sách liên kết đơn (cont.)

Tóm lại, các thao tác cơ bản trên một danh sách liên kết đơn cần có

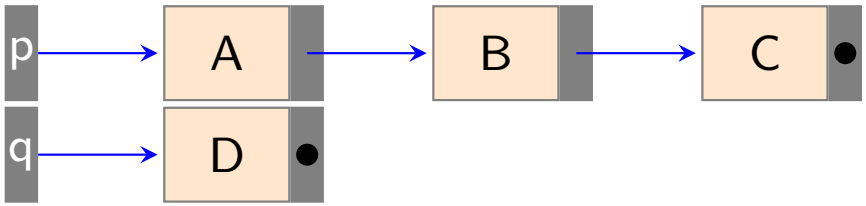
- Khởi tạo danh sách rỗng
- Sao chép danh sách
- Thêm một nút mới
- Xóa một nút
- Tìm một nút

Minh họa các thao tác

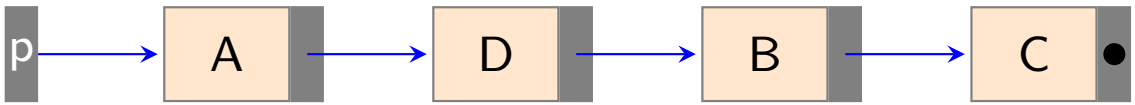
- Khởi tạo danh sách

```
1  head = NULL;
```

- Thêm một nút $q \rightarrow$ mới sau nút $p \rightarrow$ trong danh sách

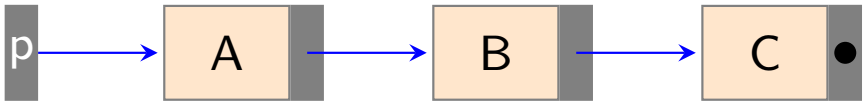



```
1  q->next = p->next;
2  p->next = q;
```

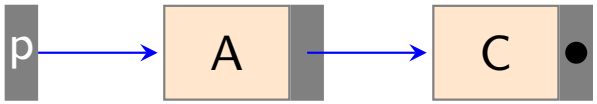


Minh họa các thao tác (cont.)

- Xóa một nút sau nút $p \rightarrow$ trong danh sách




```
1  q = p->next;
2  p->next = q->next;
3  delete q;
```



Danh sách liên kết vòng

Định nghĩa 4

Danh sách liên kết vòng (singly linked list) là một tập hợp tuyến tính các phần tử $\{x_0 \to x_1 \to ... \to x_{n-1} \to x_0\}$

- Mỗi nút chỉ có một con trỏ liên kết để trỏ đến nút kế tiếp trong danh sách
- Trong danh sách không có nút đầu tiên và nút cuối cùng
- Cần lưu trữ con trỏ liên kết tới một phần tử trong danh sách

Danh sách liên kết vòng vẫn sử dụng nút của danh sách liên kết đơn

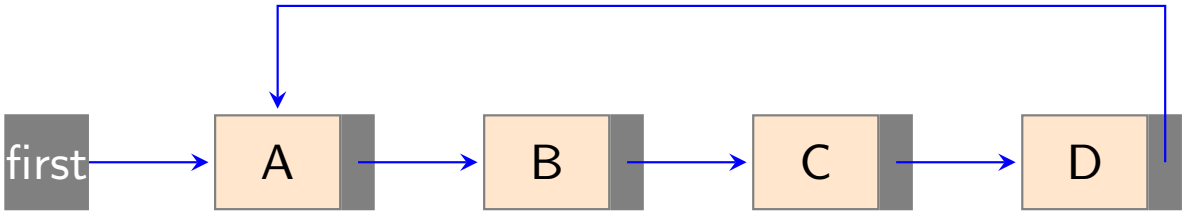
Danh sách liên kết vòng (cont.)

Cài đặt lớp cho danh sách liên kết vòng

```
1  template <class T>
2  class LinkedList
3  {
4    private:
5      int size;
6      Node<T> *head;
7
8    public:
9      isEmpty(...);
10     insert(...);
11     remove(...);
12     search(...);
13 };
```

Danh sách liên kết vòng (cont.)

Minh họa danh sách liên kết vòng



- Phần tử kế tiếp của A là B
- Phần tử kế tiếp của D là A
- Con trỏ *first* trỏ đến A

Danh sách liên kết đôi

Định nghĩa 5

Danh sách liên kết đôi (doubly linked list) là một tập hợp tuyến tính các phần tử $\{x_0 \leftrightarrow x_1 \leftrightarrow ... \leftrightarrow x_{n-1}\}$

- Mỗi nút sẽ có hai con trỏ liên kết để trỏ đến nút phía sau và nút phía trước của nút
- Nút đầu tiên con trỏ nút trước là null
- Nút cuối cùng con trỏ nút sau là null
- Cần lưu trữ con trỏ liên kết tới phần tử đầu trong danh sách

Danh sách liên kết đôi (cont.)

Khai báo kiểu dữ liệu nút cho danh sách liên kết đôi

```
1  template <class T>
2  struct Node
3  {
4      T data;
5      int key;
6      Node *prev;
7    Note *next;
8  };
```

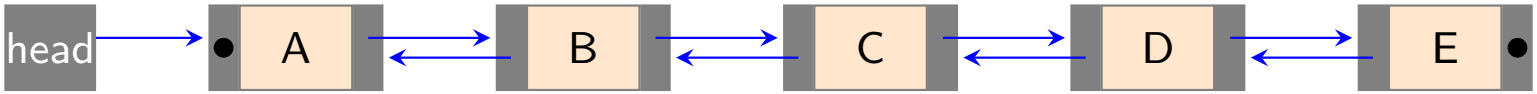
Danh sách liên kết đôi (cont.)

Cài đặt lớp cho danh sách liên kết vòng

```
1  template <class T>
2  class LinkedList
3  {
4    private:
5      int size;
6      Node<T> *head;
7
8    public:
9      isEmpty(...);
10     insert(...);
11     remove(...);
12     search(...);
13 };
```

Danh sách liên kết đôi (cont.)

Minh họa danh sách liên kết đôi



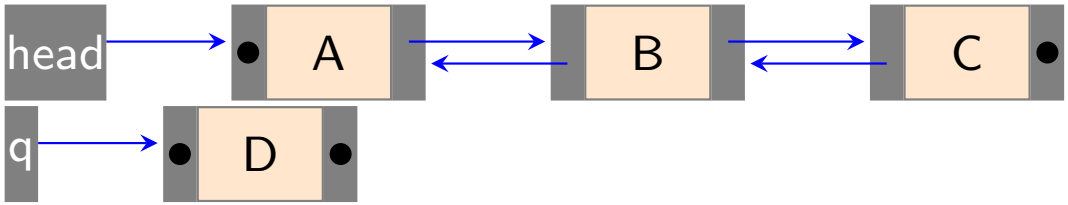
- Phần tử đầu tiên là A
- Phần tử cuối cùng là E
- Phần tử sau của B là C
- Phần tử trước của B là A

Minh họa một số thao tác

- Khởi tạo danh sách rỗng

```
1  head = NULL;
```

- Thêm một phần tử $q \rightarrow$ vào sau phần tử $p \rightarrow$ của danh sách (giả sử p trỏ tới B)




```
1  q->prev = p;
2  q->next = p->next;
3  q->next->prev = q;
4  q->prev->next = q;
```

Đánh giá Mảng vs Danh sách liên kết

Bảng 1: Bảng đánh giá

Mảng	**Danh sách liên kết**
Kích thước tương đối cố định	Kích thước thay đổi liên tục
Các phần tử lưu trữ tuần tự trong bộ nhớ	Các phần tử lưu trữ rời rạc, liên kết với nhau bằng con trỏ
Sử dụng ít bộ nhớ hơn	Sử dụng nhiều bộ nhớ hơn
Truy xuất ngẫu nhiên (nhanh)	Truy xuất tuần tự (chậm)

Tài liệu tham khảo