THƯ VIỆN LỚP THÔNG DỤNG

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GVHD: Trương Toàn Thịnh

NỘI DUNG

- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

GIỚI THIỆU

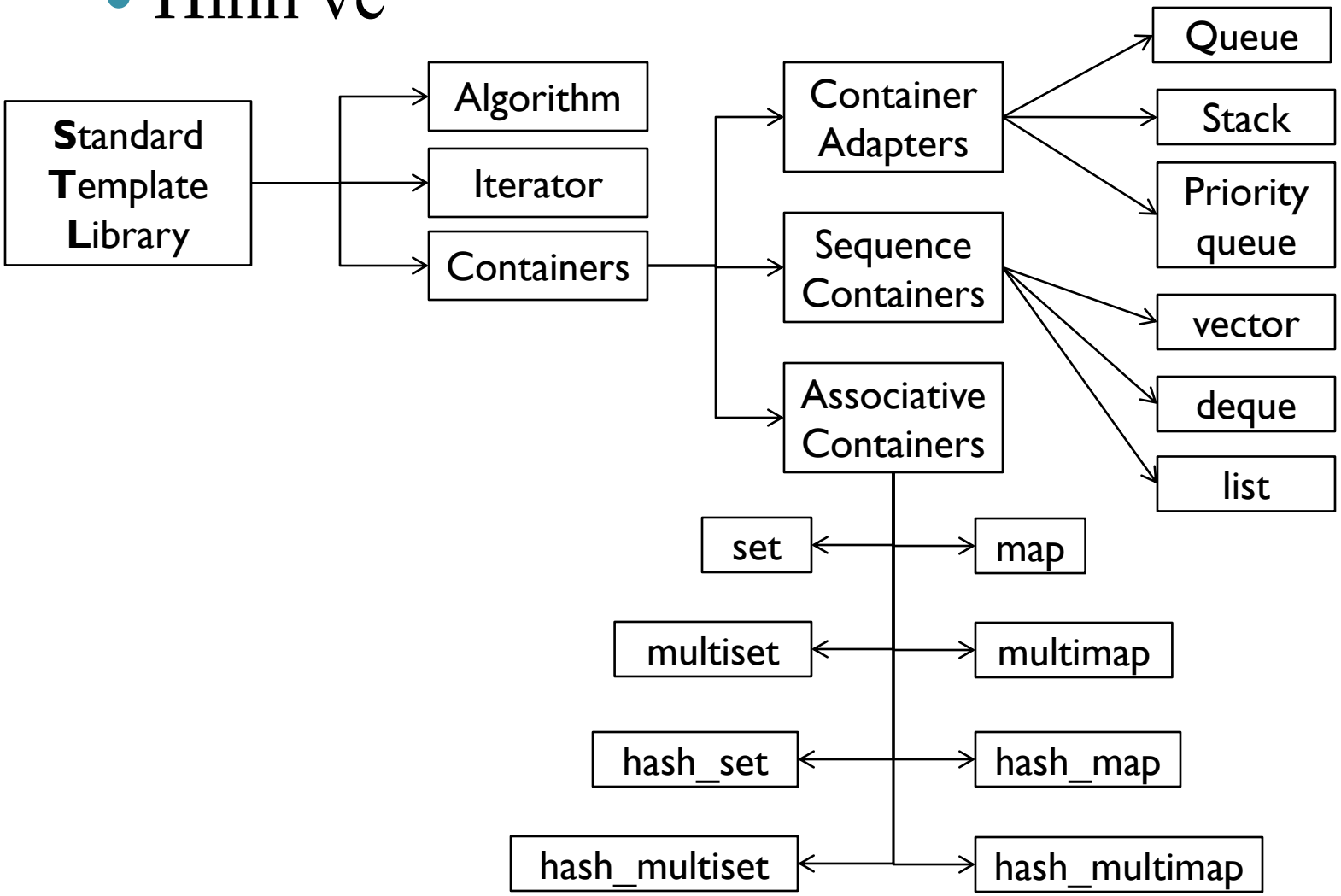
- Thư viện chuẩn C++ là nền tảng của ngôn ngữ C++
- Thư viện các lớp thông dụng của C++ (gọi là Standard Template Library – STL) là một thành phần quan trọng của thư viện chuẩn
- Thư viện STL gồm ba phần chính
  - Container: các lớp dạng tập hợp
  - Iterator: lớp đại diện để duyệt các phần tử trong Container
  - Algorithm: Các phương thức để xử lý các phần tử trong Container

GIỚI THIỆU

- Các lớp Container có thể chia ra như sau
  - **Sequence Containers**: gồm lớp vector, deque, list, các phần tử được lưu trữ đúng theo thứ tự khi đưa vào
  - **Associative Containers**: gồm lớp set, map, hash_map, hash_set, multimap, multiset, hash_multimap, hash_multiset, các phần tử được lưu trữ theo thứ tự định nghĩa trước. Associative containers được chia thành hai nhóm con
    - **Nhóm set**: gồm set, hash_set, multiset và hash_multiset, các phần tử không có thành phần khóa
    - **Nhóm map**: gồm map, hash_map, multimap và hash_multimap, các phần tử có thành phần khóa và giá trị
  - **Container Adapters**: gồm các lớp priority_queue, queue, stack, thực chất là lớp 'bao bọc' của các lớp thuộc nhóm 1

GIỚI THIỆU

- Hình vẽ



NỘI DUNG

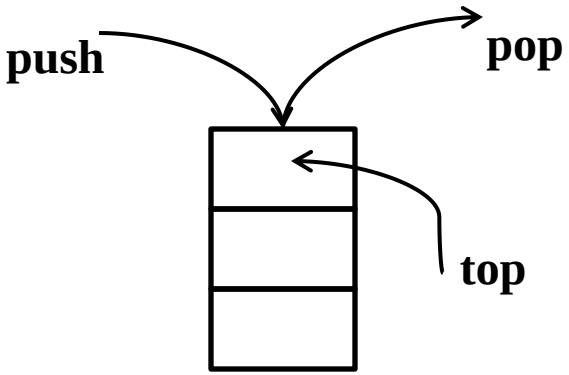
- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Lớp Stack

- Khai báo trong thư viện <stack>

```
template <class Type, class Container = deque<Type>>
class stack{}
```

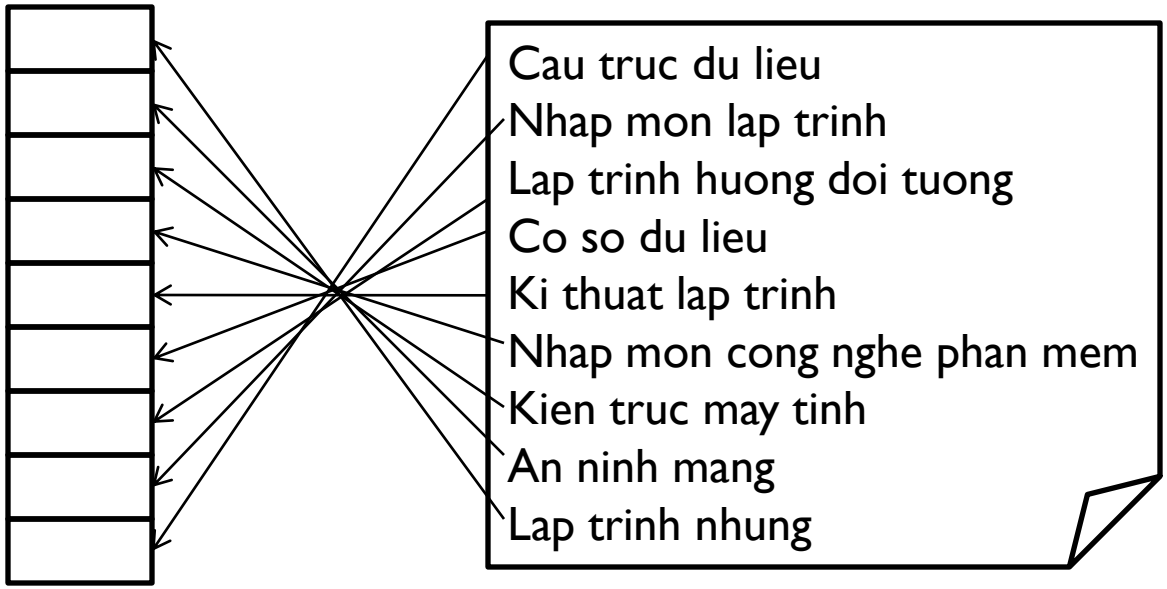
- Trong đó:
  - Type: là kiểu dữ liệu mà stack chứa
  - Container: là cấu trúc dữ liệu chính của stack
- Hình ảnh



Lớp Stack

- Ví dụ

```
#include <iostream>
#include <fstream>
#include <stack>
#include <string>
using namespace std;
void main()
{
  ifstream f("Data.txt");
  stack<string> stk;
  string line;
  while(getline(f, line))
    stk.push(line + "\n");
  while(!stk.empty()){
    cout << stk.top();
    stk.pop();
  }
}
```



NỘI DUNG

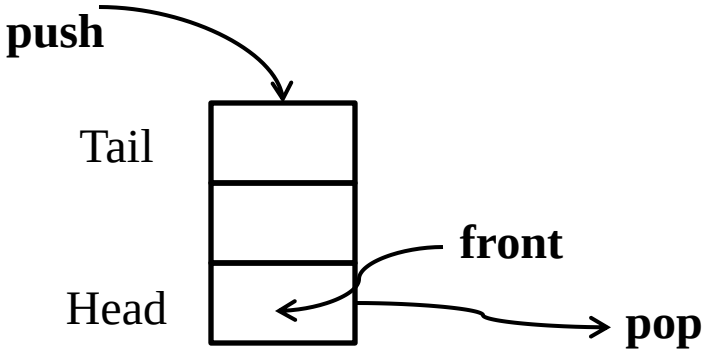
- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Lớp Queue

- Khai báo trong thư viện <queue>

```
template <class Type, class Container = deque<Type>>
class queue{}
```

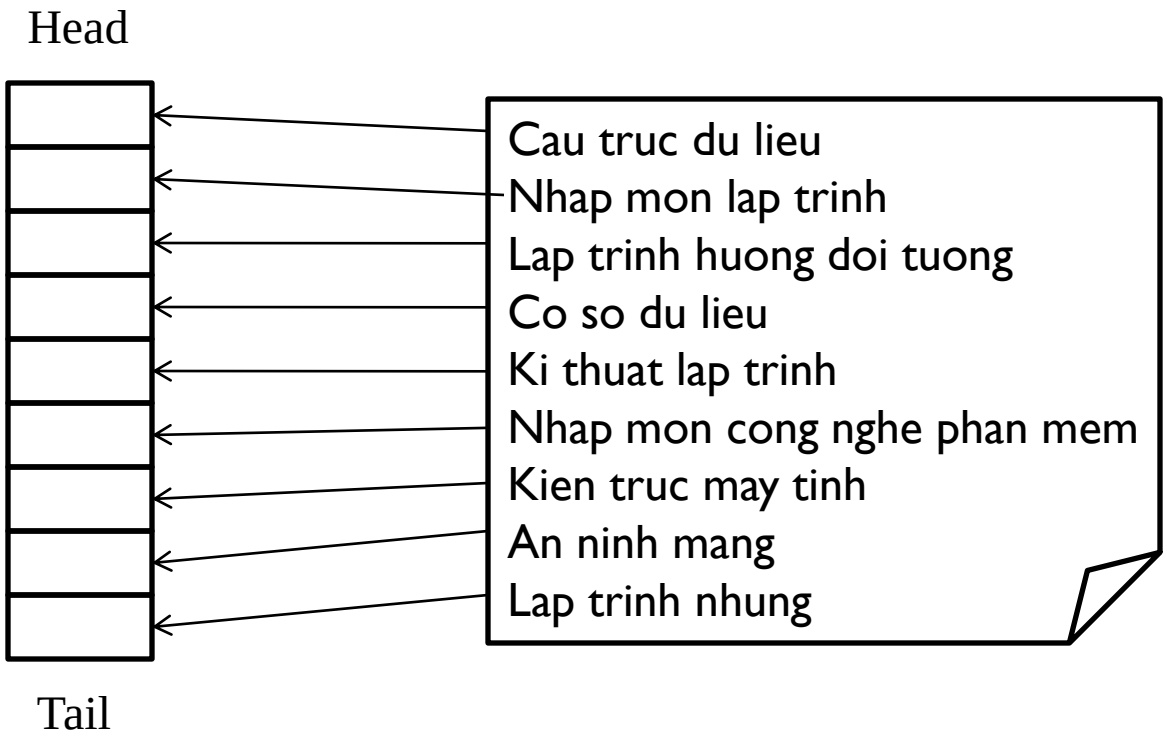
- Trong đó:
  - Type: là kiểu dữ liệu mà queue chứa
  - Container: là cấu trúc dữ liệu chính của stack
- Hình ảnh



Lớp Queue

- Ví dụ

```
#include <iostream>
#include <fstream>
#include <queue>
#include <string>
using namespace std;
void main()
{
  ifstream f("Data.txt");
  queue<string> q;
  string line;
  while(getline(f, line))
    q.push(line + "\n");
  while(!q.empty()){
    cout << q.front();
    q.pop();
  }
}
```

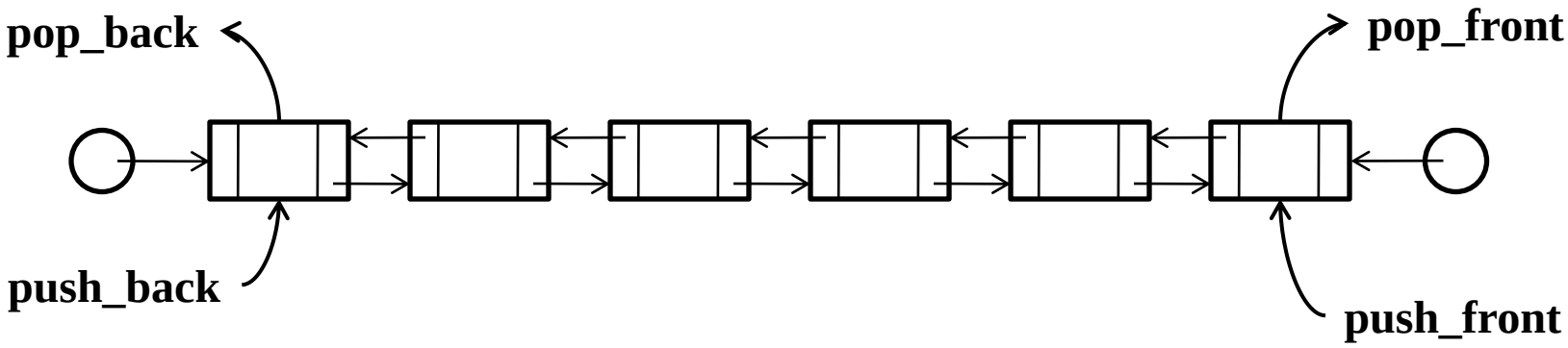


NỘI DUNG

- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Lớp List

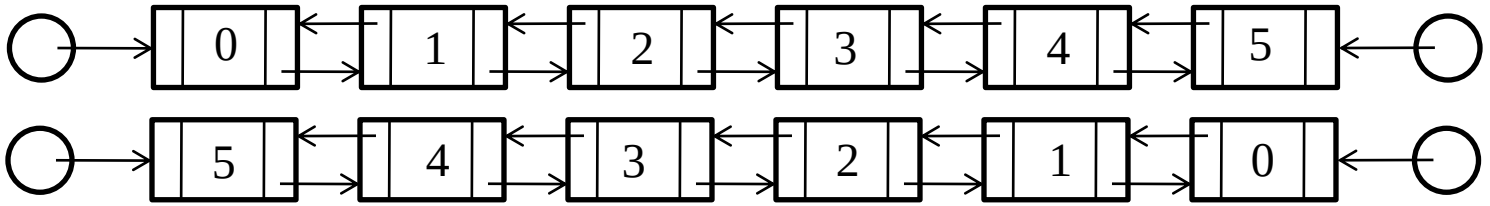
- Lưu trữ các phần tử dưới dạng danh sách liên kết đôi
- Thao tác thêm hay bớt phần tử nhanh
- Thao tác truy xuất ngẫu nhiên chậm
- Hình ảnh



Lớp List

- Ví dụ

```
#include <iostream>
#include <list>
using namespace std;
void main()
{
  list<int> l1, l2;
  for(int i = 0; i < 6; i++){l1.push_back(i); l2.push_front(i);}
  list<int>::iterator p1 = l1.begin();
  list<int>::iterator p2 = l2.begin();
  for(; p1!= l1.end(); p1++){cout<<(*p1)<<" "; }
  cout<<endl;
  for(; p2!= l2.end(); p2++){cout<<(*p2)<<" "; }
  cout<<endl;
}
```

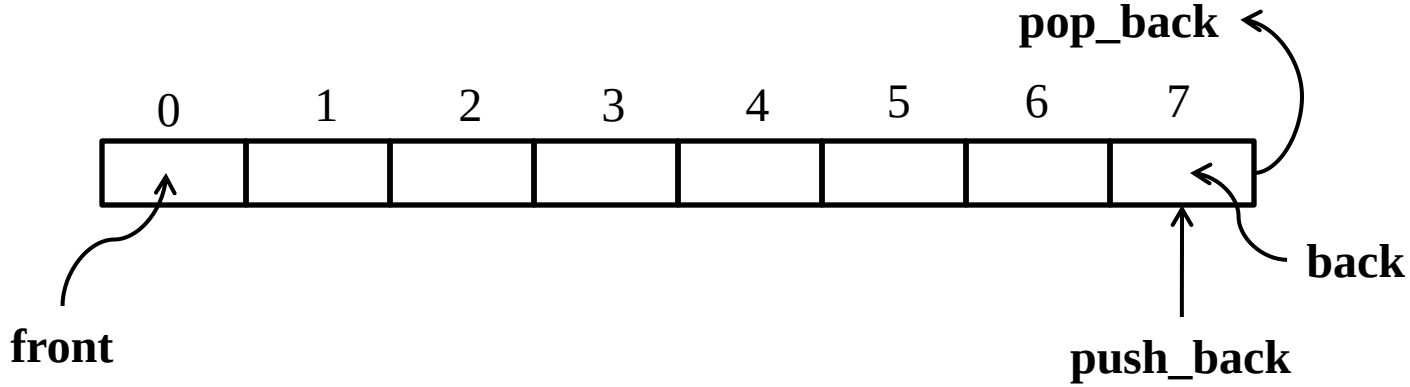


NỘI DUNG

- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Lớp Vector

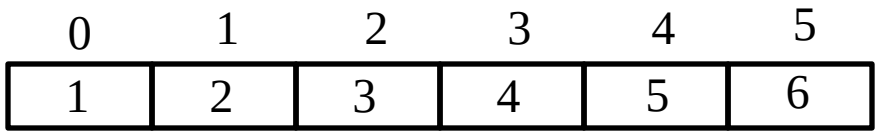
- Lưu trữ các phần tử dưới dạng liền kề nhau như mảng
- Thao tác thêm hay bớt phần tử chi phí cao
- Thao tác truy xuất ngẫu nhiên nhanh
- Dễ mở rộng các phần tử hơn mảng động
- Hình ảnh



Lớp Vector

- Ví dụ

```
#include <iostream>
#include <vector>
using namespace std;
void main()
{
  vector<int> c;
  for(int i = 1; i <= 6; i++){
    c.push_back(i);
  }
  for(int i = 0; i < c.size(); i++){
    cout<<c[i]<<“ ”;
  }
  cout<<endl;
}
```



NỘI DUNG

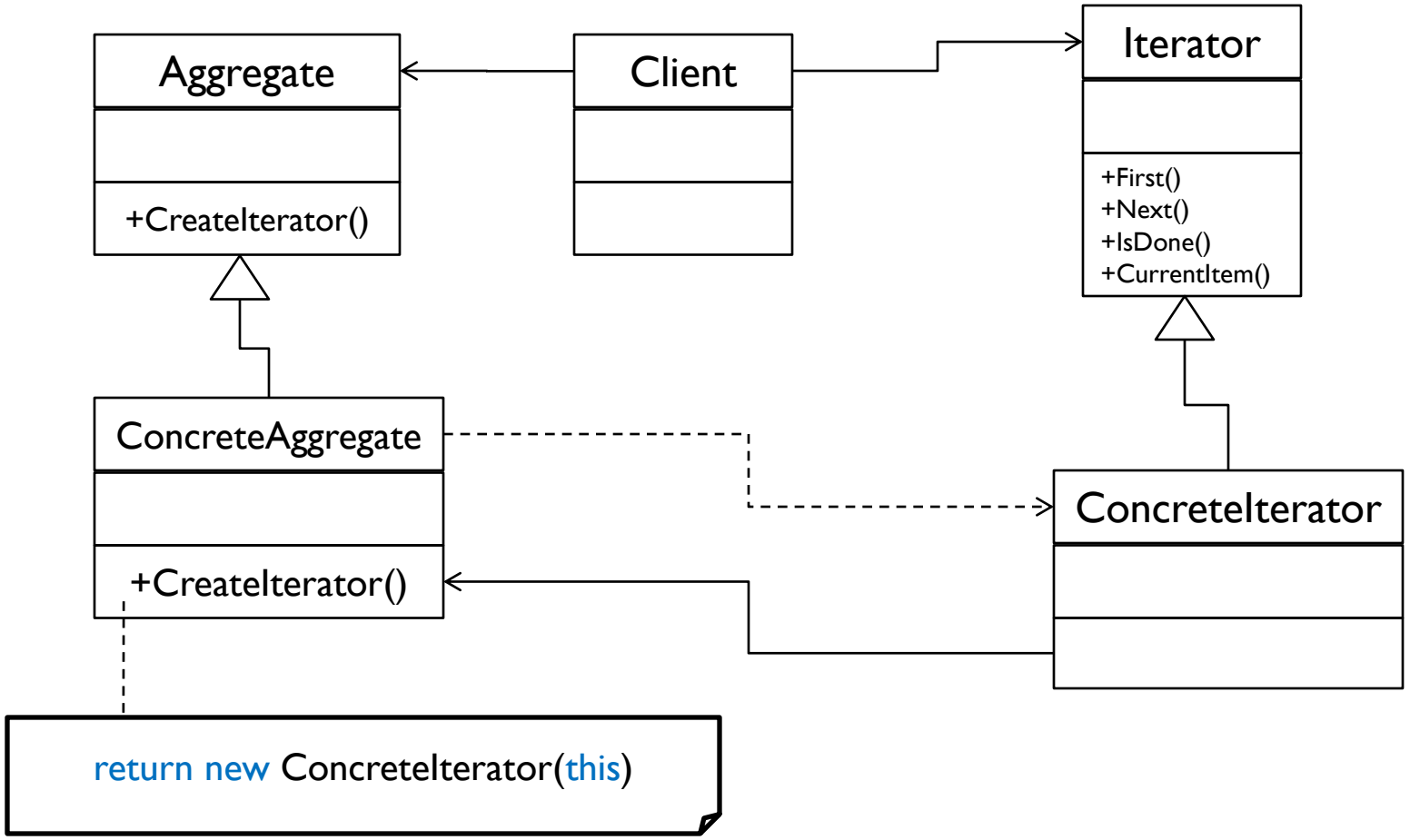
- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Mẫu thiết kế Iterator

- Tách bạch việc duyệt tập hợp ra khỏi tập hợp.
- Có nhiều cách duyệt tập hợp
- Ưu điểm:
  - Duyệt theo nhiều cách trên cùng một tập hợp
  - Đơn giản hóa lớp tập hợp (vì đã tách việc duyệt ra một lớp khác)
  - Một tập hợp có thể chọn nhiều cách duyệt

Mẫu thiết kế Iterator

- Hình vẽ



Mẫu thiết kế Iterator

- Các iterator để duyệt các container được khai báo như sau
  - Tên_container<tham số>::iterator tên
  - Tên_container<tham số>::const_iterator tên
- Các container cung cấp hai phương thức cơ bản
  - begin(): trả ra đối tượng iterator trỏ tới phần tử đầu tiên trong tập hợp
  - end(): trả ra đối tượng iterator trỏ tới phần tử sau phần tử cuối trong tập hợp

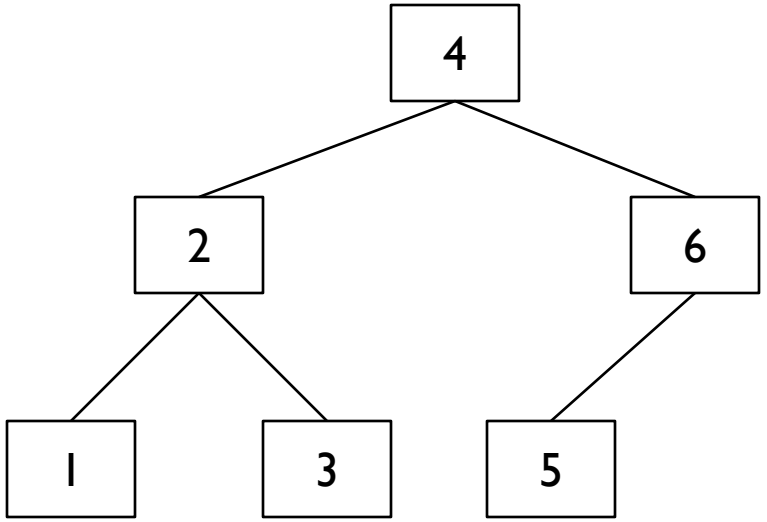
Mẫu thiết kế Iterator

- Lớp Iterator cung cấp một vài toán tử sau:
  - Toán tử *: lấy giá trị tại phần tử mà đối tượng iterator trỏ tới
  - Toán tử ++: đối tượng iterator trỏ tới phần tử kế tiếp
  - Toán tử == và !=: so sánh xem hai đối tượng iterator có trỏ tới cùng một phần tử hay không
  - Toán tử gán =

Mẫu thiết kế Iterator

- Ví dụ

```
#include <iostream>
#include <set>
using namespace std;
void main()
{
  set<int> c;
  int data[] = {3, 1, 5, 4, 1, 6, 2};
  int n = sizeof(data)/sizeof(data[0]);
  for(int i = 0; i < n; i++){
    c.insert(data[i]);
  }
  set<int>::const_iterator p = c.begin();
  while(p!=c.end()){
    cout<<*p<<" ";
    ++pos
  }
}
```



Duyệt theo thứ tự mặc định của set (LNR):
1 2 3 4 5 6

NỘI DUNG

- Giới thiệu
- Lớp Stack
- Lớp Queue
- Lớp Vector
- Lớp List
- Mẫu thiết kế Iterator
- Các thuật toán tổng quát

Các thuật toán tổng quát

- Thư viện STL cài đặt sẵn một số thuật toán xử lý các phần tử thuộc kiểu Container
  - Sắp xếp
  - Sao chép
  - Tính tổng…
- Các thuật toán được khai báo trong <algorithm> và <numeric>
  - <numeric>: cung cấp các phương thức xử lý tính toán trên số như tính tổng, tính tích
  - <algorithm>: cung cấp các hàm xử lý trên tập hợp như sắp xếp, tìm kiếm…

Các thuật toán tổng quát

- Ví dụ

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void main()
{
  vector<int> c;
  vector<int>::iterator p;
  int data[] = {2, 1, 5, 4, 3, 6};
  int n = sizeof(data)/sizeof(data[0]);
  for(int i = 0; i < n; i++) c.push_back(data[i]);
  p = min_element(c.begin(), c.end());
  cout<<“min: ” << *p<<“ ”;
  p = find(c.begin(), c.end(), 3);
}
```

Các thuật toán tổng quát

- Các thuật toán tương tác nhiều dãy con cùng lúc
  - copy(): sao chép giá trị của một dãy các phần tử tới một dãy mới
  - equal(): Kiểm tra xem hai dãy có bằng nhau hay không
- Ví dụ

```
void main(){
  list<int> c1; vector<int> c2;
  for(int i = 0; i < 10; i++) c1.push_back(i);
  c2.resize(c1.size());
  copy(c1.begin(), c1.end(), c2.begin());
}
```