**I. Kiểu dữ liệu Set trong C++**

**1. Khái niệm kiểu dữ liệu set**

*Set* là một dạng cấu trúc dữ liệu dùng để lưu trữ các phần tử không trùng lặp và được sắp xếp theo thứ tự tăng dần hoặc giảm dần. (Mặc định trong *set* là tăng dần và chúng ta có thể viết lại hàm so sánh theo mục đích của chúng ta)

Trong môn Toán lớp 66, chúng ta đã tiếp xúc với khái niệm tập hợp, và biết đến tính chất không có hai phần tử nào trùng nhau trong một tập hợp. Và kiểu dữ liệu *set* có tính ưu việt hơn so với một tập hợp, ngoài tính chất không có hai phần tử nào giống nhau mà *set* còn có tính tự sắp xếp các phần tử (Có thể rút gọn một số công đoạn sắp xếp của bài toán).

**2. Sử dụng container set**

Để sử dụng container *set* bạn cần khai báo:

#include<set>

Để khai báo một biến kiểu *set*, ta có công thức chung sau:

set<kiểu dữ liệu> tên biến;

stack<int> a;

stack<vector<int>> b;

Ví dụ về viết lại hàm thay đổi thứ tự sắp xếp các phần tử trong *set*:

struct cmp{

bool operator() (int a,int b) {return a>b;}

};

// khai báo thêm biến cmp

set<int, cmp> s;

**3.Các phép toán cơ bản của set**

* Khi duyệt các phần tử của *set*, ta sử dụng con trỏ, khai báo như sau:

set<kiểu dữ liệu>::iterator it; // khai báo con trỏ it

* Thêm một phần tử vào *set*:

s.insert(a); // thêm phần tử a vào set

* Trả về số phần tử của *set*:

s.size();

* Kiểm tra *set* rỗng hoặc không:

s.empty(); // trả về true nếu rỗng, false nếu không rỗng

* Xóa tất cả phần tử của *set*:

s.clear();

* Kiểm tra một giá trị có tồn tại trong *set* hoặc không, nếu có sẻ trả về con trỏ trỏ đến *x*, nếu không trả về s.end():

s.find(x);

* Để xóa phần tử *x* trong *set*:

s.erase(x);

* Xóa phần tử thứ *k* trong *set*:

set<int>::iterator it = s.begin(); // tạo con trỏ trỏ vào phần tử đầu tiên

advance(it,k); // trỏ đến số thứ k trong s

s.erase(it); // xóa phần tử thứ k

* Con trỏ trỏ đến vị trí phần tử nhỏ nhất mà lớn hơn khóa *x*, nếu không tìm thấy trả về vị trí s.end():

s.upper\_bound(x);

* Con trỏ trỏ đến vị trí phần tử nhỏ nhất mà lớn hơn hoặc bằng khóa *x*, nếu không tìm thấy trả về vị trí s.end():

s.lower\_bound(x);