**Bài 5: Set - Khái niệm - Sử dụng thư viện chuẩn STL cho C/C++**

**Đăng bởi: Admin | Lượt xem: 8473 | Chuyên mục: C/C++**

**Giới thiệu về Set :**

  Set là một loại associative containers để lưu trữ các phần tử không bị trùng lặp (unique elements), và các phần tử này chính là các khóa (keys).  
Ví dụ như là không tồn tại một set có  2 phần tử giống nhau như {1,2,2}, {3,4,4,4,5}.

Khi duyệt set, ta sử dụng con trỏ iterator từ begin đến end

**Các hàm của set :**

* size : trả về kích thước hiện tại của set.
* empty : true nếu set rỗng, và ngược lại.
* insert : Chèn phần tử vào set.
* erase : xóa phần tử, có 2 kiểu xóa: xóa theo iterator, hoặc là xóa theo khóa
* clear : xóa tất cả set.
* swap : đổi 2 set cho nhau.

**Truy cập phần tử :**

* find : trả về itarator trỏ đến phần tử cần tìm kiếm. Nếu không tìm thấy itarator trỏ về “end” của set.
* lower\_bound : trả về iterator đến vị trí phần tử bé nhất mà không bé hơn (lớn hơn hoặc bằng) khóa (dĩ nhiên là theo phép so sánh), nếu không tìm thấy trả về vị trí “end” của set.
* upper\_bound: trả về iterator đến vị trí phần tử bé nhất mà lớn hơn khóa, nếu không tìm thấy trả về vị trí “end” của set.
* count : trả về số lần xuất hiện của khóa trong container. Nhưng trong set, các phần tử chỉ xuất hiện một lần, nên hàm này có ý nghĩa là sẽ return 1 nếu khóa có trong container, và 0 nếu không có.

Set được thực hiện giống như cây tìm kiếm nhị phân (Binary search tree).

*Để sử dùng set ta cần dùng thư viện :*

#**include**<set>

set <Kiểu\_dữ\_liệu> tên\_Set;

//Ví dụ: set <int> s;

Để thêm một giá trị và set s ta sử dụng hàm insert(). (Độ phức tạp O(logN). Khi thêm vào một phần tử thì size() của set sẽ tự tăng thêm một.

set <**int**> s;

s.insert(1); // s={1}

s.insert(4); // s={1,4}

s.insert(2); // s={1,2,4}

s.insert(9); // s={1,2,4,9}

*\*\*Lưu ý*: trong một set sẽ không có hai phần tử cùng giá trị, nên khi bạn gọi hàm insert(x) mà trong set đó đã tồn tại giá trị x rồi thì set đó sẽ không thêm phần tử đó vào nữa.

**Bài tập ví dụ**

Cho một vector chứa các số nguyên. Hãy đưa ra số lượng phần tử khác nhau trong vector đó.

* Với inputVector = [1, 3, 3, 2], thì differentNumbers(inputVector ) = 3.  
  Giải thích: Có 3 phần tử khác nhau trong vector là: 1, 3, 2
* Với inputVector = [3, 3, 3], thì differentNumbers(inputVector ) = 1.

Đầu tiên ta cần input phần tử vào set , dưới đây là input phần tử theo thứ tự tăng dần và in ra s.size():

#**include**<iostream>

#**include**<set>

**using** **namespace** std;

**struct** **cmp**{

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

};

**int** **main**() {

set <**int**,cmp> s;

s.insert(1); // s={1}

s.insert(4); // s={4,1}

s.insert(2); // s={4,2,1}

s.insert(9); // s={9,4,2,1}

**for** (set<**int**>:: iterator it = s.begin(); it != s.end(); it++){

cout<< \*it << " ";

}

cout << s.size();

**return** 0;

}

Ở bài sau chúng ta sẽ làm quen với các bài tập về set. Chúc các bạn học tốt <3

**Bài 6: Các hàm thông dụng và bài tập minh họa về set - Sử dụng thư viện chuẩn STL cho C/C++**

**Đăng bởi: Admin | Lượt xem: 7228 | Chuyên mục: C/C++**

Ta sẽ cùng tìm hiểu một vài hàm thông dụng của set

**Hàm set::begin(std::cend) và std::end(std::cbegin):**

Hàm này sử dụng con trỏ iterator để đưa con trỏ đến vị trí đầu tiên hoặc cuối cùng của set

Ví dụ sau đây dùng để duyệt các phần tử trong set và in ra giá trị các phần tử trong set, sử dụng con trỏ iterator để duyệt:

// set::begin/end

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

**int** myints[] = {75,23,65,42,13};

std::set<**int**> **myset** (myints,myints+5);

std::cout << "myset contains:";

**for** (std::set<**int**>::iterator it=myset.begin(); it!=myset.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

**return** 0;

}

Kết quả như sau :

myset contains: 13 23 42 65 75

**Hàm set::clear :**

Xóa tất cả các phần tử trong set (bị phá hủy), kích thước của set sau khi dùng hàm bằng 0

// set::clear

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

std::set<**int**> myset;

myset.insert (100);

myset.insert (200);

myset.insert (300);

std::cout << "myset contains:";

**for** (std::set<**int**>::iterator it=myset.begin(); it!=myset.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

myset.clear();

myset.insert (1101);

myset.insert (2202);

std::cout << "myset contains:";

**for** (std::set<**int**>::iterator it=myset.begin(); it!=myset.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

**return** 0;

}

Kết quả trả về :

myset contains: 100 200 300

myset contains: 1101 2202

**Hàm set::count :**

Hàm được sử dụng để kiểm tra phần tử i có nằm trong set hay không, nếu có trả về 1 , không trả về 0;

Ta cùng tìm hiểu ví dụ sau :

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

std::set<**int**> myset;

// set some initial values:

**for** (**int** i=1; i<5; ++i) myset.insert(i\*3); // set: 3 6 9 12

**for** (**int** i=0; i<10; ++i)

{

std::cout << i;

**if** (myset.count(i)!=0)

std::cout << " la phan tu trong myset.\n";

**else**

std::cout << " khong phai phan tu trong myset.\n";

}

**return** 0;

}

Kết quả trả về :

0 khong phai phan tu trong myset.

1 khong phai phan tu trong myset.

2 khong phai phan tu trong myset.

3 la phan tu trong myset.

4 khong phai phan tu trong myset.

5 khong phai phan tu trong myset.

6 la phan tu trong myset.

7 khong phai phan tu trong myset.

8 khong phai phan tu trong myset.

9 la phan tu trong myset.

myset.count(i) != 0 : hàm kiểm tra giá trị ở i có bằng với bằng với giá trị nào trong set không, nếu có trả về vị trí 0, nếu không trả về vị trí 1

**Hàm set::empty :**

Hàm dùng để kiểm tra liệu set có rỗng hay không

// set::empty

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

std::set<**int**> myset;

myset.insert(20);

myset.insert(30);

myset.insert(10);

std::cout << "myset contains:";

**while** (!myset.empty())

{

std::cout << ' ' << \*myset.begin();

myset.erase(myset.begin());

}

std::cout << '\n';

**return** 0;

}

Kết quả trả về :

myset contains: 10 20 30

**Hàm set::erase :**

Xóa một hay nhiều phần tử trong set, các phần tử bị xóa sẽ bị phá hủy

3 tham số chính :

* position : Con trỏ đến vị trí được chỉ định trong set và xóa phần tử đó
* val : xóa giá trị của phần tử trong set
* first, last : đây là 2 giá trị sẽ xóa từ vị trí first đến vị trí last trong set

// erasing from set

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

std::set<**int**> myset;

std::set<**int**>::iterator it;

// insert some values:

**for** (**int** i=1; i<10; i++) myset.insert(i\*10); // 10 20 30 40 50 60 70 80 90

it = myset.begin();

++it; // "it" points now to 20

myset.erase (it);

myset.erase (40);

it = myset.find (60);

myset.erase (it, myset.end());

std::cout << "myset contains:";

**for** (it=myset.begin(); it!=myset.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

**return** 0;

}

Kết quả trả về :

myset contains: 10 30 50

**Hàm set::find :**

Tìm kiếm phần tử tại ví trí có giá trị bằng val, nếu giá trị đó tồn tại trong mảng thì con trỏ it sẽ truy cập đến vị trí phần tử đó

Ví dụ sau dùng để tìm kiếm xem có tồn tại vị trí có giá trị 20 trong set, nếu tồn tại con trỏ it sẽ trỏ đến vị trí đó và xóa phần tử đó:

// set::find

#**include** <iostream>

#**include** <set>

**int** **main** ()

{

std::set<**int**> myset;

std::set<**int**>::iterator it;

// set some initial values:

**for** (**int** i=1; i<=5; i++) myset.insert(i\*10); // set: 10 20 30 40 50

it=myset.find(20);

myset.erase (it);

myset.erase (myset.find(40));

std::cout << "myset contains:";

**for** (it=myset.begin(); it!=myset.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

**return** 0;

}

Kết quả trả về :

myset contains: 10 30 50

**Bài tập ví dụ :**

**Lý thuyết.**

Khi duyệt các phần tử trong list chúng ta phải làm quen với 1 kiểu dữ liệu là iterator, hiểu đơn giản thì đây là một con trỏ.

Ta cũng cần chú ý là 2 phương thức là begin() và end(), hai phương thức này sẽ trả về con trỏ của phần tử thứ nhất và con trỏ ở sau phần tử cuối cùng.

Để duyệt list theo chiều thuận (từ trái qua phải) ta làm như sau:

**for** (list<int>::iterator it = a.**begin**(); it !=a.**end**(); it++)

**Bài 1:**Cho một vector chứa các số nguyên. Hãy đưa ra số lượng phần tử khác nhau trong vector đó.

* Với*inputVector = [1, 3, 3, 2]*, thì *differentNumbers(inputVector ) = 3*. Giải thích: Có 3 phần tử khác nhau trong vector là: 1, 3, 2
* Với *inputVector = [3, 3, 3*], thì *differentNumbers(inputVector ) = 1*.

Hướng dẫn : Ta sẽ thêm các phần tử của vector và một set, sau đó đưa ra số phần tử của set đó

**int** **differentNumbers**(vector<**int**> inputVector )

{

set<**int**> s;

**for** (**int** i = 0; i < inputVector.size(); i++){

s.insert(inputVector[i]);

}

**return** s.size();

}

**Bài 2 :**Cho một ma trận gồm các dãy nhị phân khác nhau. Hãy đưa ra các dãy nhị phân khác nhau trong ma trận đó.

  Với matrix =   
[[1,1,0,1],  
 [1,0,0,1],  
 [1,1,0,1]].  
thì đầu ra sẽ là: uniqueRow(matrix) =  
[[1,1,0,1],  
 [1,0,0,1]].

std::vector<std::vector<**int**>> **uniqueRows**(std::vector<std::vector<**int**>> matrix)

{

set<vector<**int**>> s;

vector<vector<**int**>> result;

**for**(std::vector<std::vector<**int**>>::iterator it = matrix.begin();it!=matrix.end();it++){

**if**(!s.count(\*it)){

s.insert(\*it);

result.push\_back(\*it);

}

}

**return** result;

}

**Bài 3 :**Anh Việt đang thông kê số liệu cho công ty, anh muốn giá trị nhỏ thứ hai ở trong một dãy số nguyên. Hãy giúp anh Việt tìm ra giá trị đó, nếu không có kết quả như yêu cầu thì trả về "NO".

* Với*arr = [1, 2, 3, 1, 1]* thì kết quả sẽ là*secondOrder(arr) = "2"*.
* Với *arr = [-4, 1, 2, 2]* thì kết quả sẽ là *secondOrder(arr) = "1"*.

Hướn dẫn :

Lưu dãy và một set, sau đó kiểm tra, nếu số lượng phần tử trong set đó nhỏ hơn 2 thì in ra "NO", ngược lại in số thứ 2 ra.

std::string **secondOrder**(std::vector<**int**> arr)

{

std::set<**int**> **s**(arr.begin(), arr.end());

std::set<**int**>::iterator it = s.begin();

std::advance(it, 1);

**if**(s.size() > 1)

**return** std::to\_string(\*it);

**else**

**return** "NO";

}