* Đơn đồ thị
* Đa đồ thị
* Giả đồ thị vô hướng
* Đơn đồ thị có hướng
* Đa đồ thị có hướng
* Đỉnh kề , cạnh liên thuộc (Cạnh nối hai đỉnh kề)
* Bặc của đỉnh : Đồ thị vô hướng
* Bán bậc của đỉnh : Đồ thị có hướng

-----------------------------------------------------------

**CÁCH KẾT HỢP GIỮA MẢNG VÀ VECTOR TRONG C++**

* Trong C++, mảng (array) và vector là hai cấu trúc dữ liệu phổ biến được sử dụng để lưu trữ một tập hợp các phần tử. Dưới đây là cách kết hợp giữa mảng và vector trong C++
* Mảng trong C++ là một cấu trúc dữ liệu cố định, có kích thước xác định khi khai báo và không thể thay đổi sau đó.
* Khai báo một mảng có thể được thực hiện như sau:

int arr[5];

int x = arr[1];

vector<int> vt;

vt.push\_back(10);

* Ta cũng có thể chuyển đổi qua lại giữa mảng và vector :

+ CHUYỂN ĐỔI TỪ MẢNG SANG VECTOR

int arr[5] = {1, 2, 3, 4, 5};

vector<int> vec(arr,arr+5);

vector <int> vt;

vt.assign(arr,arr+5);

+ CHUYỂN ĐỔI VECTOR SANG MẢNG

int a[5];

copy(vt.begin(),vt.end(), a);

* Để tạo một mảng các vector trong C++ , bạn có thể sử dụng một vector của vector, tức là vector lớn chứa các vector con.

vector<vector<int> > vt;

a.push\_back(vector<int>{1,2,3});

for (auto i =0;i<vt.size();vt++){

cout << “Vector : “ << i <<” :”;

for (auto j =0;j< vt[i].size() ;j++){

cout<< vt[i][j] << “ “;

}

cout<< endl;

}

* Ta duyệt qua từng thằng như thường , nhưng mỗi thằng lúc này lại là một vector nên ta lại phải tiếp tục duyệt tiếp một mảng mới.

int main() {

- Tạo một vector các vector

vector<vector<int>> vector\_of\_vectors;

- Nhập dữ liệu từ bàn phím và thêm vào vector chính

vector<int> vec1, vec2, vec3;

cout << "Nhap vao cac phan tu cua vector 1: ";

int num;

while (cin >> num) {

vec1.push\_back(num);

}

cout << "Nhap vao cac phan tu cua vector 2: ";

while (cin >> num) {

vec2.push\_back(num);

}

cout << "Nhap vao cac phan tu cua vector 3: ";

while (cin >> num) {

vec3.push\_back(num);

}

vector\_of\_vectors.push\_back(vec1);

vector\_of\_vectors.push\_back(vec2);

vector\_of\_vectors.push\_back(vec3);

// Truy cập và sử dụng các phần tử của vector các vector

for (size\_t i = 0; i < vector\_of\_vectors.size(); ++i) {

cout << "Vector " << i << ": ";

for (size\_t j = 0; j < vector\_of\_vectors[i].size(); ++j){

cout << vector\_of\_vectors[i][j] << " ";

}

cout << endl;

}

return 0;

}

* Để tạo và sử dụng một vector trong đó mỗi phần tử là một mảng, bạn có thể sử dụng một mảng sau đó chuyển đổi sang vector để đưa vào vector:

#include <iostream>

#include <vector>

int main() {

- Tạo một vector chứa các mảng

vector<vector<int>> vt;

- Tạo và thêm các mảng vào vector

int arr1[] = {1, 2, 3};

int arr2[] = {4, 5, 6};

int arr3[] = {7, 8, 9};

vt.push\_back(vector<int>(arr1, arr1 + sizeof(arr1) / sizeof(arr1[0])));

vt.push\_back(vector<int>(arr2, arr2 + sizeof(arr2) / sizeof(arr2[0])));

vt.push\_back(vector<int>(arr3, arr3 + sizeof(arr3) / sizeof(arr3[0])));

- Truy cập và sử dụng các phần tử của vector

for (size\_t i = 0; i < vt.size(); ++i) {

cout << "Array " << i << ": ";

for (size\_t j = 0; j < vector\_of\_arrays[i].size(); ++j) {

cout << vector\_of\_arrays[i][j] << " ";

}

cout << endl;

}

return 0;

}

-------------------------------------------------------

* Cách khai báo một mảng bình thường :

datatype tenmang[size];

vector<int> a[1000];

* Ví dụ :

+ Khai báo một mảng với mỗi phần tử của mảng là 1 số nguyên :

int a[1000];

+ Khai báo một mảng với mỗi phần tử của mảng là 1 chuỗi:

string a[1000];

+ Khai báo một mảng với mỗi phần tử của mảng là một vector:

vector<int> a[1000] ;

* Các phần tử trong mảng giờ có các
* Giờ ta coi 1 phần tử trong mảng là 1 vector nên để thêm một phần tử thì ta phải tại cái giá trị cuả chỉ số đó dùng thêm hàm push\_back mới thêm đc phần tử.

 adj[x].push\_back(y);

       adj[y].push\_back(x);

for (int i=0;i< m ;i++){

       int x,y;

       cin >> x >> y;

       a[x][y] = a[y][x] = 1;

    }

    for (int i=1;i<=n ;i++){

        for (int j=1;j<=n;j++){

            cout<<a[i][j] <<"   ";

        }

        cout <<endl;

    }

* Các cái đỉnh mà ta nhập vào luôn > 0 nên các phần tử ở 0 luôn không luôn bằng 0 nên ta không duyệt thành

0 1 1 1 0

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

0 1 1 1 0

TEST :

0 0 0 0 0

0 0 1 1 1

0 1 0 1 1

0 1 1 0 1

0 1 1 1 0

  string s,num;

    getline(cin,s);

    stringstream ss(s);

    while(ss >> num){

        a[i][stoi(num)] = 1;

        }

   }

* Dòng code bạn đã cung cấp thực hiện việc đọc một dòng từ đầu vào (được nhập bởi người dùng) vào biến s dưới dạng một chuỗi. Sau đó, nó sử dụng stringstream để phân tích chuỗi s thành các phần tử riêng biệt, được tách ra bởi khoảng trắng hoặc dấu phân tách khác (ví dụ: dấu cách).
* getline(cin, s);: Dòng này đọc một dòng từ đầu vào tiêu chuẩn (cin) và lưu trữ nó vào biến s dưới dạng một chuỗi.
* stringstream ss(s);: Dòng này khởi tạo một đối tượng stringstream với chuỗi s. stringstream là một lớp trong thư viện chuẩn C++ (<sstream>) cho phép bạn thực hiện các thao tác đọc và ghi trên chuỗi dưới dạng dữ liệu.
* while (ss >> num) { ... }: Đây là một vòng lặp while. Trong mỗi lần lặp, nó cố gắng đọc một phần tử từ stringstream ss vào biến num. Nếu việc đọc thành công (tức là vẫn còn dữ liệu để đọc), vòng lặp sẽ thực hiện các câu lệnh bên trong. Nếu không còn dữ liệu để đọc, điều kiện của vòng lặp sẽ trở thành sai và vòng lặp sẽ kết thúc.
* a[i][stoi(num)] = 1;: Đây là phần trong vòng lặp while. Nó chuyển đổi chuỗi num thành một số nguyên (stoi là hàm trong thư viện <string> để chuyển đổi chuỗi sang số nguyên), sau đó đặt giá trị 1 vào phần tử tương ứng trong mảng hai chiều a.
* Điều này giúp bạn chuyển đổi dữ liệu được nhập từ đầu vào (có thể là danh sách kề của một đồ thị) thành một ma trận kề, nơi mỗi hàng của ma trận biểu diễn một đỉnh và mỗi cột biểu diễn một đỉnh khác trong đồ thị. Nếu có một cạnh giữa đỉnh i và j, thì a[i][j] sẽ được đặt thành 1, ngược lại sẽ là 0.
* Khởi tạo stringstream từ chuỗi s: Dòng code stringstream ss(s); tạo ra một đối tượng stringstream mới được khởi tạo từ chuỗi s. stringstream hoạt động như một bộ đệm tạm thời, nó có thể nhận dữ liệu từ nhiều nguồn khác nhau, nhưng trong trường hợp này, nó nhận dữ liệu từ chuỗi s.
* Phân tích chuỗi thành các phần tử: Sau khi khởi tạo, stringstream ss có thể được sử dụng giống như cin hoặc bất kỳ luồng đầu vào nào khác. Khi bạn sử dụng ss >> num, nó cố gắng "đọc" dữ liệu từ stringstream, cụ thể là các phần tử được tách ra từ chuỗi s. Khi gặp khoảng trắng hoặc dấu phân tách khác, nó sẽ ngừng và lưu trữ phần tử đã đọc vào biến num.
* Lặp lại quá trình: Dòng while (ss >> num) sẽ tiếp tục lặp lại việc đọc từ stringstream cho đến khi không còn dữ liệu để đọc nữa. Trong mỗi lần lặp, nếu việc đọc thành công, dữ liệu được lưu trữ vào biến num, ngược lại, nếu không còn dữ liệu nào để đọc, điều kiện của vòng lặp sẽ trở thành sai và vòng lặp sẽ kết thúc.
* Như vậy, stringstream giúp bạn chuyển đổi một chuỗi thành các phần tử riêng biệt để dễ dàng xử lý chúng một cách độc lập. Trong trường hợp của bạn, bạn sử dụng stringstream để phân tách chuỗi đó thành các số nguyên và lưu vào ma trận tương ứng.

Top of Form