Chương trình này sử dụng thuật toán tô màu đồ thị bằng phương pháp tô màu tham lam. Dưới đây là giải thích từng dòng code trong từng hàm của chương trình:

Hàm readGraph:

* vector<vector<int> > readGraph(): khai báo hàm readGraph, trả về một vector chứa thông tin về đồ thị.
* int numVertices, numEdges;: khai báo hai biến numVertices và numEdges, lần lượt lưu số lượng đỉnh và số lượng cạnh của đồ thị.
* cout << "Nhap so dinh cua do thi: ";: in ra thông báo yêu cầu người dùng nhập số lượng đỉnh của đồ thị.
* cin >> numVertices;: đọc giá trị số lượng đỉnh mà người dùng đã nhập vào từ bàn phím.
* cout << "Nhap so canh cua do thi: ";: in ra thông báo yêu cầu người dùng nhập số lượng cạnh của đồ thị.
* cin >> numEdges;: đọc giá trị số lượng cạnh mà người dùng đã nhập vào từ bàn phím.
* vector<vector<int> > graph(numVertices);: khai báo một vector graph chứa numVertices phần tử, mỗi phần tử là một vector chứa thông tin về các đỉnh liên kết với đỉnh tương ứng trong đồ thị.
* cout << "Nhap cac canh cua do thi:\n";: in ra thông báo yêu cầu người dùng nhập thông tin về các cạnh của đồ thị.
* for (int i = 0; i < numEdges; i++): lặp numEdges lần để đọc thông tin về các cạnh của đồ thị.
* int u, v; cin >> u >> v;: đọc hai giá trị u và v là các đỉnh của một cạnh từ bàn phím.
* graph[u - 1].push\_back(v - 1); graph[v - 1].push\_back(u - 1);: thêm đỉnh v-1 vào vector chứa thông tin về các đỉnh liên kết với đỉnh u-1, và thêm đỉnh u-1 vào vector chứa thông tin về các đỉnh liên kết với đỉnh v-1.
* return graph;: trả về vector graph chứa thông tin về các cạnh của đồ thị.

Hàm printColors:

* void printColors(const vector<int>& colors): khai báo hàm printColors, nhận một vector colors là vector chứa thông tin về màu của các đỉnh.
* cout << "\nMau cua cac dinh:\n";: in ra thông báo để người dùng biết đây là thông tin về màu của các đỉnh.
* for (int vertex = 0; vertex < colors.size(); vertex++): lặp qua các phần tử của vector colors.
* cout << "Dinh " << vertex + 1 << ": Mau " << colors[vertex] << "\n";: in ra thông tin về màu của đỉnh vertex+1.

Hàm greedyColoring:

* vector<int> greedyColoring(const vector<vector<int> >& graph): khai báo hàm greedyColoring, nhận một vector graph chứa thông tin về các cạnh của đồ thị.
* int numVertices = graph.size();: lấy số lượng đỉnh của đồ thị từ kích thước của vector graph.
* vector<int> vertexColors(numVertices, -1);: khởi tạo một vector vertexColors có numVertices phần tử, mỗi phần tử ban đầu được gán màu -1.
* set<int> usedColors;: khởi tạo một setusedColors để lưu trữ các màu đã được sử dụng để tô các đỉnh kề của đỉnh đang xét.
* int numUsedColors = 0;: khởi tạo biến numUsedColors để đếm số lượng màu đã sử dụng để tô đồ thị.
* for (int vertex = 0; vertex < numVertices; vertex++): lặp qua từng đỉnh của đồ thị.
* for (int i = 0; i < graph[vertex].size(); i++): lặp qua từng đỉnh kề của đỉnh đang xét.
* int neighbor = graph[vertex][i]; if (vertexColors[neighbor] != -1) { usedColors.insert(vertexColors[neighbor]); }: nếu đỉnh kề đã được tô màu trước đó, thêm màu đã sử dụng để tô đỉnh kề đó vào usedColors.
* for (int color = 0; color < numVertices; color++): lặp qua các màu có thể sử dụng để tô đỉnh đang xét.
* if (usedColors.find(color) == usedColors.end()): nếu màu color chưa được sử dụng để tô các đỉnh kề của đỉnh đang xét.
* vertexColors[vertex] = color;: gán màu color cho đỉnh đang xét.
* numUsedColors = max(numUsedColors, color + 1);: cập nhật số lượng màu đã sử dụng để tô đồ thị.
* break;: thoát khỏi vòng lặp để tô màu cho đỉnh đang xét.
* usedColors.clear();: xóa tất cả các màu đã được sử dụng để tô các đỉnh kề của đỉnh đang xét.
* cout << "\nSo mau can dung: " << numUsedColors << "\n";: in ra thông tin về số lượng màu cần sử dụng để tô đồ thị.
* return vertexColors;: trả về vector vertexColors chứa thông tin về màu của các đỉnh.