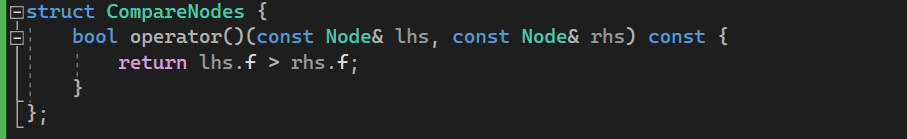
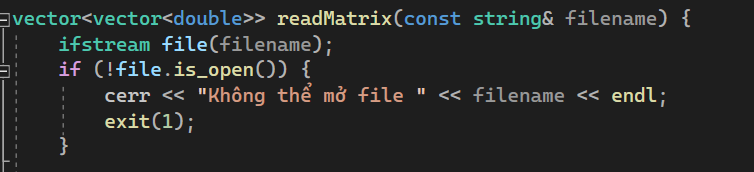


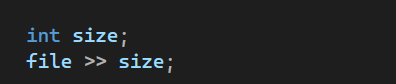
Mỗi nút có một id (định danh), giá trị g (chi phí thực tế từ nút gốc đến nút hiện tại), và giá trị h (ước lượng khoảng cách từ nút hiện tại đến nút đích). f là tổng của g và h, được sử dụng để xác định xem nút nào nên được xem xét tiếp theo trong hàng đợi ưu tiên.



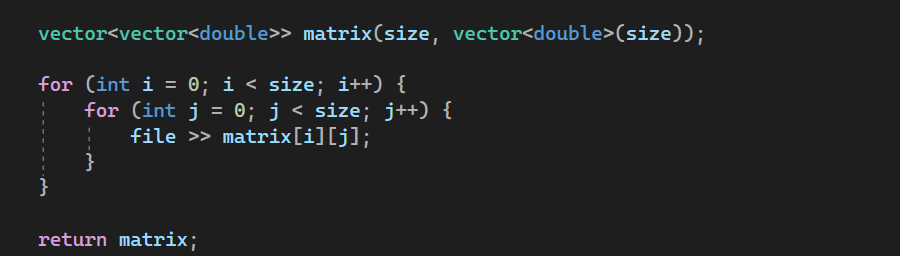
CompareNodes để so sánh hai nút dựa trên giá trị f. Điều này cần thiết để sắp xếp các nút trong hàng đợi ưu tiên theo thứ tự tăng dần của f



Hàm nhận một tham số là filename, là tên của tệp văn bản chứa ma trận. Nó trả về một vector hai chiều chứa ma trận và được đánh dấu const để bảo vệ tham số đầu vào khỏi sửa đổi. Chúng ta bắt đầu bằng việc mở tệp với tên được truyền vào bằng cách sử dụng ifstream. Nếu không thể mở tệp, chúng ta in một thông báo lỗi và thoát khỏi chương trình với lỗi mã 1.

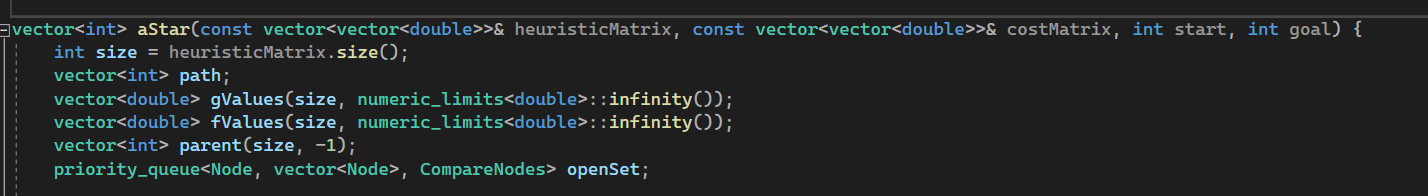


đọc giá trị size từ tệp văn bản. Giá trị này biểu thị kích thước của ma trận (số hàng và số cột).



Sử dụng hai vòng lặp lồng nhau, chúng ta đọc từng giá trị trong tệp và lưu chúng vào ma trận matrix. Vòng lặp ngoài duyệt qua từng hàng của ma trận, và vòng lặp trong duyệt qua từng cột của hàng đó. Mỗi lần duyệt, chúng ta đọc một giá trị từ tệp bằng file >> matrix[i][j].

Sau đó trả về ma trận đã đọc từ tệp văn bản, hoàn thành việc đọc dữ liệu từ tệp và lưu nó trong ma trận.



Hàm aStar nhận ba đối số:

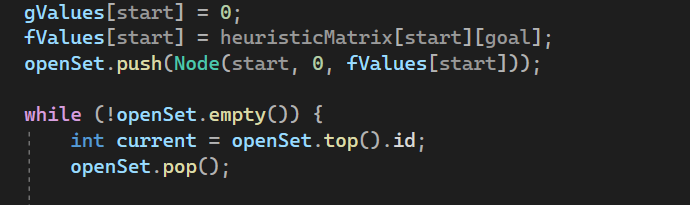
heuristicMatrix: Ma trận ước lượng khoảng cách h(x) từ mỗi đỉnh đến đỉnh đích. Đây là một ma trận hai chiều size x size, trong đó size là số đỉnh trong đồ thị.

costMatrix: Ma trận chi phí costMatrix[i][j] cho biết chi phí đi từ đỉnh i đến đỉnh j. Nếu không có kết nối trực tiếp từ i đến j, thì giá trị này là 0.

start: Đỉnh xuất phát.

goal: Đỉnh đích.

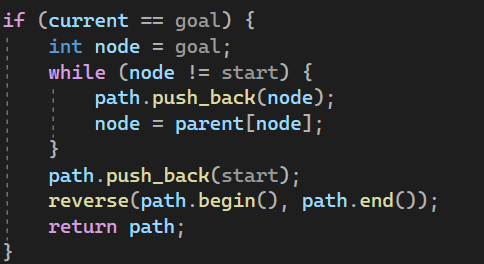
khởi tạo các biến và containers cần thiết cho thuật toán A\*. size là số đỉnh trong đồ thị. path sẽ chứa đường đi cuối cùng từ start đến goal. gValues là vector chứa giá trị g(x) (chi phí thực tế từ start đến mỗi đỉnh). fValues là vector chứa giá trị f(x) (tổng của g(x) và h(x)). parent là vector chứa đỉnh cha của mỗi đỉnh để truy vết đường đi. openSet là hàng đợi ưu tiên lưu trữ các đỉnh chưa được xem xét theo thứ tự tăng dần của f(x).



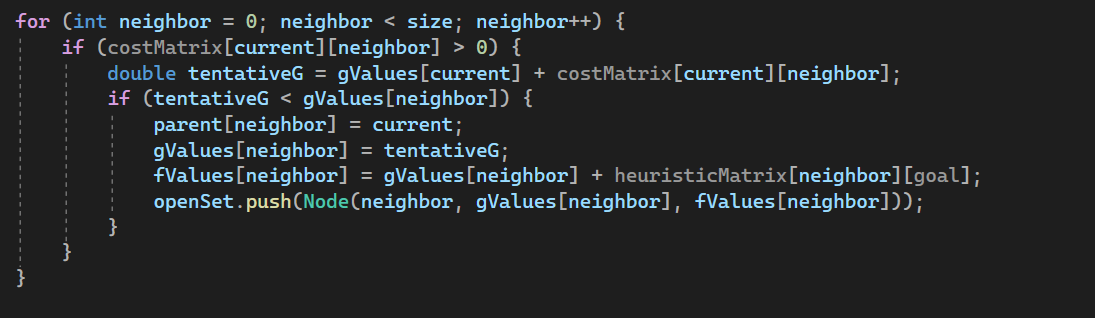
khởi tạo giá trị g và f ban đầu cho đỉnh xuất phát (start). gValues[start] được thiết lập thành 0 vì chi phí từ start đến chính nó là 0. fValues[start] được thiết lập bằng giá trị h(start, goal) từ ma trận ước lượng heuristicMatrix.

Sử dụng một vòng lặp while để duyệt qua các đỉnh trong openSet cho đến khi openSet trống.

Ở mỗi vòng lặp, lấy ra đỉnh có giá trị f(x) nhỏ nhất từ openSet và lưu nó vào biến current.



Nếu current là đỉnh đích (goal), chúng ta đã tìm thấy đường đi từ start đến goal. Thực hiện truy vết từ đỉnh goal trở về start thông qua parent để xây dựng đường đi. Đường đi được lưu trong vector path và được đảo ngược để có thứ tự đúng từ start đến goal. Sau đó, trả về path.



Trong vòng lặp, duyệt qua các đỉnh hàng xóm của current bằng cách sử dụng vòng lặp for. Nếu có một kết nối (chi phí lớn hơn 0) từ current đến neighbor, chúng ta kiểm tra xem có một đường đi ngắn hơn từ start đến neighbor thông qua current không. Nếu có, chúng ta cập nhật giá trị g(x), f(x), và đỉnh cha cho neighbor, sau đó đưa neighbor vào openSet để xem xét sau. Thuật toán sẽ tiếp tục duyệt qua các đỉnh hàng xóm và cập nhật thông tin cho đến khi tìm thấy đường đi ngắn nhất hoặc không còn đỉnh nào trong openSet. Nếu openSet trống mà vẫn chưa tìm thấy đường đi, trả về một vector rỗng, cho biết rằng không có đường đi từ start đến goal.