HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN 2**

-----🙞🙜🕮🙜🙞-----

**BÀI THI TIỂU LUẬN**

**MÔN HỌC: NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: TÌM HIỂU VẤN ĐỀ TÌM KIẾM CÓ THÔNG TIN VÀ HIỆN THỰC GIẢI THUẬT A-STAR BẰNG NGÔN NGỮ C++**

Giảng viên : Nguyễn Ngọc Duy

Sinh viên : Nguyễn Ngọc Phương Trinh

Mã sinh viên : N18DCCN231

Lớp : D18CQCN03-N

**TP. HỒ CHÍ MINH, 2021**

**MỤC LỤC**

[I. Giới thiệu bài toán tìm kiếm có thông tin 1](#_Toc79005604)

[1. Tổng quan về giải thuật tìm kiếm có thông tin 1](#_Toc79005605)

[2. Heuristic 2](#_Toc79005606)

[3. Ví dụ về xây dựng hàm Heuristic 3](#_Toc79005607)

[II. Giới thiệu giải thuật tìm kiếm A\* 4](#_Toc79005609)

[1. Tổng quan về lịch sử phát triển 4](#_Toc79005610)

[2. Ý tưởng trực quan 5](#_Toc79005611)

[3. Mô tả thuật toán 6](#_Toc79005612)

[4. Thuật toán A\* 8](#_Toc79005613)

[5. Đánh giá thuật toán A\* 9](#_Toc79005614)

[III. Giới thiệu ngôn ngữ lập trình C++ 9](#_Toc79005615)

[1. Ngôn ngữ C++ là gì? 9](#_Toc79005616)

[2. Ứng dụng của ngôn ngữ C++ 11](#_Toc79005618)

[IV. Hiện thực giải thuật A-Star bằng ngôn ngữ C++ 11](#_Toc79005619)

[1. Sơ lược bài toán 11](#_Toc79005620)

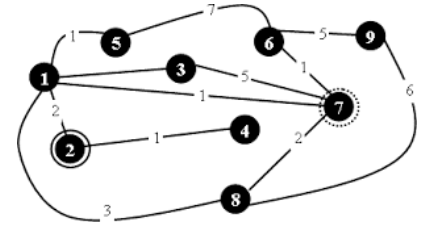
[2. Triển khai thuật toán A-Star bằng ngôn ngữ C++ 13](#_Toc79005621)

# Giới thiệu bài toán tìm kiếm có thông tin

## Tổng quan về giải thuật tìm kiếm có thông tin

Trong ngành [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), một **giải thuật tìm kiếm** là một [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) lấy đầu vào là một bài toán và trả về kết quả là một lời giải cho bài toán đó, thường là sau khi cân nhắc giữa một loạt các lời giải có thể. Hầu hết các thuật toán được nghiên cứu bởi các nhà khoa học máy tính để giải quyết các bài toán đều là các thuật toán tìm kiếm. Tập hợp tất cả các lời giải có thể đối với một bài toán được gọi là [không gian tìm kiếm](https://vi.wikipedia.org/w/index.php?title=Kh%C3%B4ng_gian_t%C3%ACm_ki%E1%BA%BFm&action=edit&redlink=1). [Thuật toán thử sai](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_th%E1%BB%AD_sai&action=edit&redlink=1) (*brute-force search*) hay các thuật toán tìm kiếm "sơ đẳng" không có thông tin sử dụng phương pháp đơn giản nhất và trực quan nhất. Trong khi đó, các **thuật toán tìm kiếm có thông tin** sử dụng [heuristics](https://vi.wikipedia.org/wiki/Heuristics) để áp dụng các tri thức về cấu trúc của [không gian tìm kiếm](https://vi.wikipedia.org/w/index.php?title=Kh%C3%B4ng_gian_t%C3%ACm_ki%E1%BA%BFm&action=edit&redlink=1) nhằm giảm thời gian cần thiết cho việc tìm kiếm.

Trong tìm kiếm có thông tin, người ta sử dụng một đánh giá [heuristic](https://vi.wikipedia.org/wiki/Heuristic) đặc thù cho bài toán cần giải quyết với vai trò hướng dẫn cho quá trình tìm kiếm. Một cách đánh giá heuristic tốt sẽ làm cho quá trình tìm kiếm có thông tin hoạt động hiệu quả hơn hẳn một phương pháp tìm kiếm không có thông tin bất kỳ.



Có một vài thuật toán tìm kiếm có thông tin nổi trội dành cho danh sách. Một trong số đó là một [bảng băm](https://vi.wikipedia.org/wiki/B%E1%BA%A3ng_b%C4%83m) với một hàm băm là một heuristic dựa trên bài toán đang được giải. Đa số các thuật toán tìm kiếm có thông tin đều là tìm kiếm trên cây. Trong đó có [tìm kiếm theo lựa chọn tốt nhất](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t) và [A\*](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm_A*). Cũng như các thuật toán không có thông tin, các thuật toán này có thể được mở rộng để làm việc trên cả các đồ thị.

## Heuristic

**Heuristic** ([/hjʊəˈrɪstɪk/](https://en.wikipedia.org/wiki/Help:IPA/English); [tiếng Hy Lạp cổ](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Hy_L%E1%BA%A1p_c%E1%BB%95): εὑρίσκω, "tìm kiếm" hay "khám phá") là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hay khám phá nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu. Với việc nghiên cứu khảo sát không có tính thực tế, các phương pháp heuristic được dùng nhằm tăng nhanh quá trình tìm kiếm với các giải pháp hợp lý thông qua các suy nghĩ rút gọn để giảm bớt việc nhận thức vấn đề khi đưa ra quyết định. Ví dụ của phương pháp này bao gồm sử dụng một [luật ngón tay cái](https://vi.wikipedia.org/w/index.php?title=Lu%E1%BA%ADt_ng%C3%B3n_tay_c%C3%A1i&action=edit&redlink=1), [giả thuyết](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3_thuy%E1%BA%BFt), phán đoán trực giác, khuôn mẫu hay [nhận thức thông thường](https://vi.wikipedia.org/w/index.php?title=Nh%E1%BA%ADn_th%E1%BB%A9c_th%C3%B4ng_th%C6%B0%E1%BB%9Dng&action=edit&redlink=1).

Thuật giải Heuristic là một sự mở rộng khái niệm thuật toán. Nó thể hiện cách giải bài toán với các đặc tính sau :

* Thườngtìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất)
* Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn
* Thuật giải Heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người

    Có nhiều phương pháp để xây dựng một thuật giải Heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ sở như sau:

***Nguyên lý vét cạn thông minh :***

Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

***Nguyên lý tham lam (Greedy):***

Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

***Nguyên lý thứ tự :***

Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

***Hàm Heuristic:***

Trong việc xây dựng các thuật giải Heuristic, người ta thường dùng các *hàm Heuristic*. Ðó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

Một số vấn đề của hàm heuristic h(n):

* Nếu xây dựng hàm heuristic đúng bản chất vấn đề thì tìm kiếm sẽ hiệu quả. Ngược lại, xây dựng không tốt có thể đi chệch hướng và tìm kiếm kém hiệu quả
* Việc xây dựng hàm heuristic sẽ tùy thuộc vào vấn đề cần giải quyết cụ thể
* Hàm heuristic được gọi là chấp nhận được khi: h(n) ≤ h\*(n)

h\*(n): giá thành đường đi thực tế từ n đến node đích

## Ví dụ về xây dựng hàm Heuristic

## 

* Xét bài toán xếp ô số mong muốn.
* Với một trạng thái u:
* h1 (n) là hàm trạng thái các số đặt sai chỗ
  + h1 (u) = 6 (các số 2, 4, 5, 6, 7, 8 đặt sai chỗ)
* h2 (u) là tổng khoảng cách giữa vị trí số hiện tại với vị trí ở trạng thái đích
  + h2 (u) = 0 + 3 + 0 + 2 + 2 + 2 +1 + 2 = 12
* Hàm h2 (n) mang nhiều thông tin hơn nên có thể dẫn đến kết quả nhanh hơn. Chọn h2 (n).

🡪 Tổng quát: Chọn hàm h’(n) = max(h1 (n), h2 (n))

* Cách xây dựng hàm heuristic:
* Mỗi bài toán cụ thể sẽ có cách chọn hàm heuristic riêng để được tối ưu.
* Nguyên tắc chung: Nới lỏng các ràng buộc của bài toán đó.
* Ví dụ bài toán 8 số ở trên: bỏ ràng buộc về việc các ô phải di chuyển từng bước để đến được vị trí đích.

# Giới thiệu giải thuật tìm kiếm A\*

## 1. Tổng quan về lịch sử phát triển

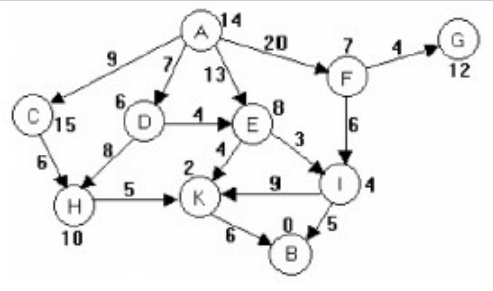
* Trong [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), **A\*** (đọc là *A sao*) là [thuật toán tìm kiếm trong đồ thị](https://vi.wikipedia.org/wiki/Duy%E1%BB%87t_c%C3%A2y). Thuật toán này tìm một đường đi từ một [nút](https://vi.wikipedia.org/w/index.php?title=N%C3%BAt_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1) khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này. Do đó, thuật toán A\* là một ví dụ của [tìm kiếm theo lựa chọn tốt nhất](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t) (*best-first search*).
* Thuật toán A\* được mô tả lần đầu vào năm 1968 bởi [Peter Hart](https://vi.wikipedia.org/w/index.php?title=Peter_Hart&action=edit&redlink=1), [Nils Nilsson](https://vi.wikipedia.org/w/index.php?title=Nils_Nilsson&action=edit&redlink=1), và [Bertram Raphael](https://vi.wikipedia.org/w/index.php?title=Bertram_Raphael&action=edit&redlink=1). Trong bài báo của họ, thuật toán được gọi là thuật toán A; khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu, do đó mà có tên A\*.
* Năm 1964, Nils Nilsson phát minh ra một phương pháp tiếp cận dựa trên khám phá để tăng tốc độ của thuật toán Dijkstra. Thuật toán này được gọi là A1. Năm 1967 Bertram Raphael đã cải thiện đáng kể thuật toán này, nhưng không thể hiển thị tối ưu. Ông gọi thuật toán này là A2. Sau đó, trong năm 1968 Peter E. Hart đã giới thiệu một đối số chứng minh A2 là tối ưu khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu. Chứng minh của ông về thuật toán cũng bao gồm một phần cho thấy rằng các thuật toán A2 mới là thuật toán tốt nhất có thể được đưa ra các điều kiện. Do đó ông đặt tên cho thuật toán mới là A \*(A sao, A-star).

## 2. Ý tưởng trực quan

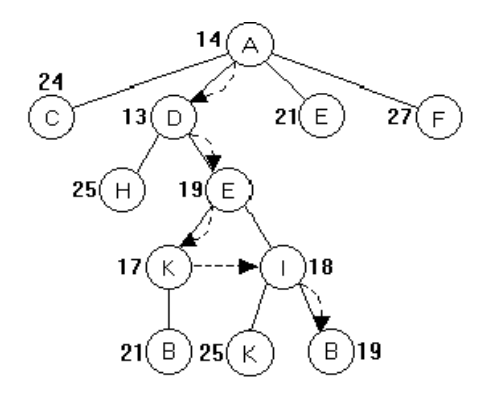
* Xét bài toán tìm đường - bài toán mà A\* thường được dùng để giải. A\* xây dựng tăng dần tất cả các tuyến đường từ điểm xuất phát cho tới khi nó tìm thấy một đường đi chạm tới đích. Tuy nhiên, cũng như tất cả các [thuật toán tìm kiếm có thông tin](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_t%C3%ACm_ki%E1%BA%BFm_c%C3%B3_th%C3%B4ng_tin&action=edit&redlink=1), nó chỉ xây dựng các tuyến đường "có vẻ" dẫn về phía đích.
* Để biết những tuyến đường nào có khả năng sẽ dẫn tới đích, A\* sử dụng một "đánh giá heuristic" về khoảng cách từ điểm bất kỳ cho trước tới đích. Trong trường hợp tìm đường đi, đánh giá này có thể là khoảng cách **đường chim bay** - một đánh giá xấp xỉ thường dùng cho khoảng cách của đường giao thông.
* Điểm khác biệt của A\* đối với [tìm kiếm theo lựa chọn tốt nhất](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t) là nó còn tính đến khoảng cách đã đi qua. Điều đó làm cho A\* "đầy đủ" và "tối ưu", nghĩa là, A\* sẽ luôn luôn tìm thấy đường đi ngắn nhất nếu tồn tại một đường đi như vậy. A\* không đảm bảo sẽ chạy nhanh hơn các thuật toán tìm kiếm đơn giản hơn. Trong một môi trường dạng mê cung, cách duy nhất để đến đích có thể là trước hết phải đi về phía xa đích và cuối cùng mới quay lại. Trong trường hợp đó, việc thử các nút theo thứ tự "gần đích hơn thì được thử trước" có thể gây tốn thời gian.
* Ý tưởng: Sử dụng hàm heuristic chấp nhận được, tìm kiếm theo chiều rộng để loại bỏ những đường đi có chi phí cao.
* Hàm lượng giá: **f(u) = g(u) + h(u)**
* g(u): Chi phí để đến u
* h(u): Lượng giá từ u đến đích
* f(u): Ước lượng tổng giá đến đích qua u

## 3. Mô tả thuật toán

* A\* lưu giữ một tập các lời giải chưa hoàn chỉnh, nghĩa là các đường đi qua đồ thị, bắt đầu từ nút xuất phát. Tập lời giải này được lưu trong một [hàng đợi ưu tiên](https://vi.wikipedia.org/w/index.php?title=H%C3%A0ng_%C4%91%E1%BB%A3i_%C6%B0u_ti%C3%AAn&action=edit&redlink=1) (*priority queue*). Thứ tự ưu tiên gán cho một đường đi **{\displaystyle x}x** được quyết định bởi hàm **{\displaystyle f(x)=g(x)+h(x)}f(x) = g(x) + h(x)**.
* Trong đó, **{\displaystyle g(x)}g(x)** là chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua. **{\displaystyle h(x)}h(x)** là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích từ **{\displaystyle x}x**. Ví dụ, nếu "chi phí" được tính là khoảng cách đã đi qua, [khoảng cách đường chim bay](https://vi.wikipedia.org/wiki/Kho%E1%BA%A3ng_c%C3%A1ch_%C4%91%C6%B0%E1%BB%9Dng_chim_bay) giữa hai điểm trên một bản đồ là một đánh giá heuristic cho khoảng cách còn phải đi tiếp.
* Hàm **{\displaystyle f(x)}f(x)** có giá trị càng thấp thì độ ưu tiên của **{\displaystyle x}x** càng cao *(do đó có thể sử dụng một cấu trúc*[*heap tối thiểu*](https://vi.wikipedia.org/w/index.php?title=Heap_(c%E1%BA%A5u_tr%C3%BAc_d%E1%BB%AF_li%E1%BB%87u)&action=edit&redlink=1)*để cài đặt hàng đợi ưu tiên này)*
* Ví dụ: Cho không gian trạng thái như hình. Trạng thái đầu: A, trạng thái đích: B. Trọng số của cạnh là độ dài đường đi. Các số cạnh các đỉnh là giá trị hàm h. Tìm đường đi ngắn nhất từ A đến B bằng thuật toán A\*.



* Cây tìm kiếm thuật toán A\*



Đầu tiên, phát triển A sinh ra C, D, E, F. Ta có:

* g(C)=9, h(C)=15, f(C)=g(C) + h(C)=9+15=24;
* Tính tương tự f(D)=13, f(E)=21, f(F)=27.
* Như vậy đỉnh tốt nhất là D vì f(D) nhỏ nhất.
* Phát triển D, ta được H và E mới. Ta có:

g(H)=g(D)+Độ dài cung (D, H) = 7 + 8 = 15, f(H) = 15 + 10 = 25.

* g(E) = g(D) + Độ dài cung (D, E) = 7 + 4 = 11. Vậy đỉnh E mới có f(E) = g(E) + h(E) = 11 + 8 = 19< f(E cũ) => bỏ đỉnh E cũ thay bằng E mới
* Nếu E mới có giá trị >= giá trị E cũ thì giữ E cũ, bỏ E mới.
* Như vậy đỉnh tốt nhất là E.
* Phát triển E được K và I. Tiếp tục cho tới khi đỉnh được chọn để phát triển là đỉnh kết thúc B thì ngừng
* Độ dài đường đi ngắn nhất tới B là g(B) = 19
* Chuỗi biến đổi : A → D → E → I → B

|  |  |  |
| --- | --- | --- |
| Bước lặp | Open | Close |
| Khởi tạo | (A,14,null) // A có giá trị f=14, và có cha là null | Rỗng |
| 1 | (D,13,A), (E,21,A), (C,24,A), (F,27,A) | A |
| 2 | (E,19,D), (C,24,A), (H,25,D), (F,27,A) | A, D |
| 3 | (K,17,E), (I,18,E), (C,24,A), (H,25,D), (F,27,A) | A, D, E |
| 4 | (I,18,E), (B,21,K), (C,24,A), (H,25,D), (F,27,A) | A, D, E, K |
| 5 | (B,19,I), (C,24,A), (K,25,I), (H,25,D), (F,27,A) | A, D, E, K, I |
| 6 | Lấy được đỉnh B trong open => ngừng | A, D, E, K, I, B |

## 4. Thuật toán A\*

* Open: tập các trạng thái đã được sinh ra nhưng chưa được xét đến.
* Close: tập các trạng thái đã được xét đến.
* cost(p, q): là khoảng cách giữa p, q.
* g(p): khoảng cách từ trạng thái đầu đến trạng thái hiện tại p.
* h(p): giá trị được lượng giá từ trạng thái hiện tại đến trạng thái đích.
* f(p) = g(p) + h(p)
  + Bước 1:
    - Open: = {s}
    - Close: = {}
  + Bước 2: while (Open !={})
    - Chọn trạng thái (đỉnh) tốt nhất p trong Open (Open được sắp xếp theo thứ tự giá trị f tăng dần).
    - Nếu p là trạng thái kết thúc thì thoát.
    - Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p.
      * Nếu q đã có trong Open
        + Nếu g(p) + cost(p, q) < g(q)

g(q) = g(p) + cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p (đỉnh cha của q là p)

* + - * Nếu q chưa có trong Open
        + g(q) = g(p) + cost(p, q)
        + f(q) = g(q) + h(q)
        + prev(q) = p
        + Thêm q vào Open
      * Nếu q có trong Close (điều này để ngăn thuật toán quay ngược trở lại)
        + Nếu g(p) + cost(p, q) < g(q)

Bỏ q khỏi Close

Thêm q vào Open

* + Bước 3: Không tìm được.

## 5. Đánh giá thuật toán A\*

* Nếu hàm h(n) đánh giá thấp nhất thì thuật toán A\* là tối ưu
* Nếu các cung không nhỏ hơn một số α nào đó thì A\* đầy đủ
* Nếu h(n) = 0 ∀n, thuật toán A\* chính là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(n).
* Có tính đủ, trừ trường hợp có vô số node với hàm f có giá trị rất nhỏ nằm giữa node xuất phát và node đích.
* Có tính tối ưu và hiệu quả
* Độ phức tạp theo thời gian là hàm mũ

# Giới thiệu ngôn ngữ lập trình C++

## 1. Ngôn ngữ C++ là gì?

Ngôn ngữ lập trình C++ là một ngôn ngữ lập trình hướng đối tượng(OOP – Object-oriented programming) được phát triển bởi Bjarne Stroustrup. C++ là ngôn ngữ lập trình được phát triển trên nên tảng của ngôn ngữ lập trình C. Do đó, C++ có song song cả 2 phong cách(style) lập trình hướng cấu trúc giống C và có thêm phong cách hướng đối tượng. Trong nhiều trường hợp, C++ sử dụng kết hợp cả 2 style trên. Do đó, nó được xem là một ngôn ngữ “lai tạo”.

Ngôn ngữ C++ là một ngôn ngữ lập trình cấp trung. Bởi vì nó có các tính chất của cả ngôn ngữ lập trình bậc thấp(Pascal, C…) và ngôn ngữ lập trình bậc cao(C#, Java, Python…).

Ngôn ngữ lập trình C++(C plus plus) có đuôi mở rộng là ***.cpp***

C++ được thiết kế thiên về lập trình hệ thống và phần mềm máy tính hiệu suất và hiệu quả làm việc linh hoạt chính là điểm nhấn của ngôn ngữ này. C++ cũng thể hiện rất nhiều những tính năng hữu ích của mình trong nhiều bối cảnh công việc khác nhau, ngôn ngư lập trình này vô cùng thích hợp dành cho các ứng dụng máy tính, máy chủ hay chuyển mạch điện thoại, thăm dò không gian.

### **Những phiên bản C++**

* Ngôn ngữ lập trình này được ISO công nhận chuẩn hóa đầu tiền vào năm 1998 với tên gọi là dạng ISO/ IEC 14882: 1998.
* Sau đó là C++ 03, C++ 11 và C++ 14.
* Tháng 12 năm 2017 tổ chức tiêu chuẩn quốc tế (ISO) đã công nhận và chuẩn hóa phiên bản mới nhất của C++ là ISO/ IEC 14882: 2017 (gọi tắt là C++ 17). Và kế hoạch tiếp theo sẽ là C++ 20.

C++ là ngôn ngữ lập trình dạng tự do, nó được biên soạn, phân biệt chữ hoa chữ thường, gõ tĩnh, hỗ trợ trình thủ tục, hướng đến những đối tượng chung. Ngôn ngữ có một thư viên tiêu chuẩn phong phú với một tập hợp các hàm phong phú thao tác các tệp và phương thức thao tác các cấu trúc dữ liệu,… C++ được sử dụng rộng rãi trong công việc lập trình của các lập trình viên hoặc nhà phát triển chủ yếu trong một miền của ứng dụng. Vì lẽ, ngôn ngữ C++ chưa đựng những phần quan trọng bao gồm những ngôn ngữ cốt lõi, các khối xây dựng cần thiết,…

## 2. Ứng dụng của ngôn ngữ C++

Vai trò đặc biệt quan trọng của bộ đôi ngôn ngữ lập trình C và C++ là phục vụ cho học lập trình cơ bản. Bởi vì đây là một ngôn ngữ lập trình bậc trung. Hầu hết các trường đào tạo công nghệ thông tin ở Việt Nam đều dùng 2 ngôn ngữ này làm môn cơ sở ngành.

Môn lập trình C giúp chúng ta có nền tảng với kỹ thuật lập trình, các kiến thức cơ bản và tư duy lập trình. Môn lập trình C++ cung cấp kiến thức về lập trình hướng đối tượng. Cả 2 ngôn ngữ này được sử dụng để học các môn về cấu trúc dữ liệu và giải thuật.

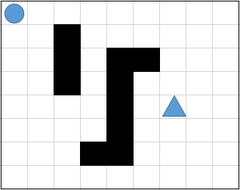
Đối với các ứng dụng, bài toán yêu cầu hiệu năng cao, tiết kiệm bộ nhớ thì C++ là một lựa chọn tuyệt vời. Đồng thời, ngôn ngữ C++ vẫn có tính chất hướng đối tượng giúp dễ dàng bảo trì và nâng cấp.

# Hiện thực giải thuật A-Star bằng ngôn ngữ C++

## 1. Sơ lược bài toán

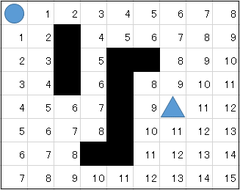
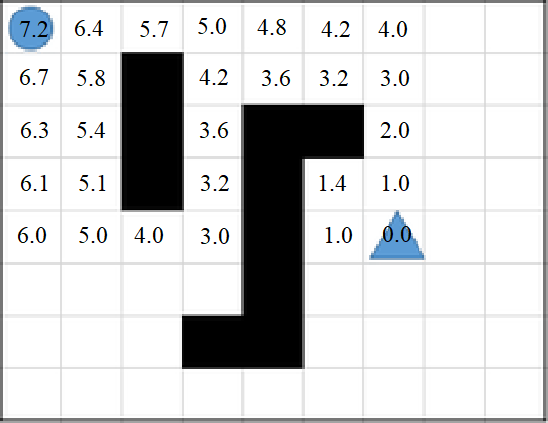
Cho một bàn cờ kích thước n x n. Hãy tìm đường đi ngắn nhất từ điểm 〇 đến điểm △ biết rằng:

* Các ô trắng là đường đi được
* Các ô đen là vật cản, không đi được
* Tại một ô bất kỳ, chỉ có thể đi lên, xuông, trái, phải nếu không bị cản. Không được đi chéo.



Hình 1. Bàn cờ với kích thước 9\*8

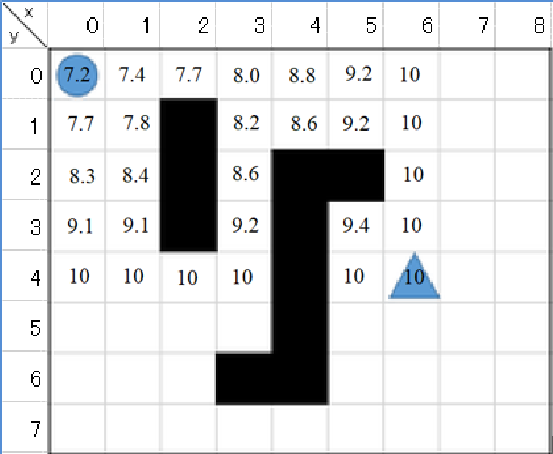
* Trong bài toán trên ta thấy rằng đường đi ngắn nhất là đường chéo nối từ 〇 đến △, chiều dài đường chéo được tính bằng công thức{\displaystyle h(x)={\sqrt {a^{2}+b^{2}}}={\sqrt {7^{2}+5^{2}}}=8.6}
* Khi di chuyển qua mỗi ô thì giá trị của ô sẽ tăng lên 1 đơn vị

Hình 2. g(x)

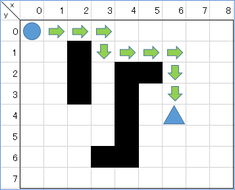
Hình 3. h(x)=

Vì là {\displaystyle f(x)=g(x)+h(x)}f(x) = g(x) + h(x) nên ta có bảng như sau:



Hình 4. f(x) = g(x) + h(x)

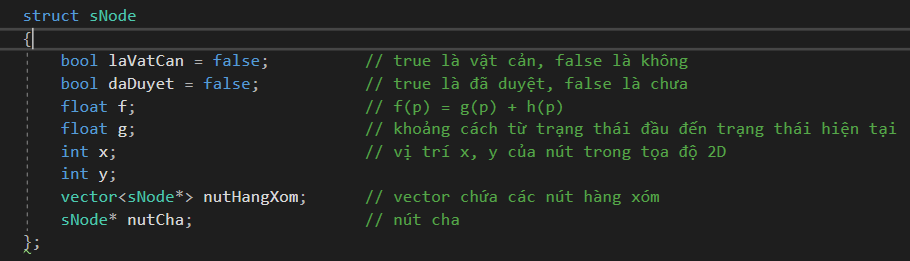
Tiếp theo từ vị trí 0 sẽ duyệt các ô lân cận nó để tìm ra đường ngắn nhất



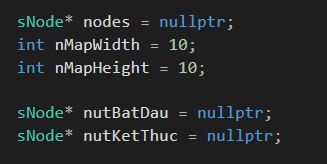
Hình 5. Đường đi tìm được

## 2. Triển khai thuật toán A-Star bằng ngôn ngữ C++

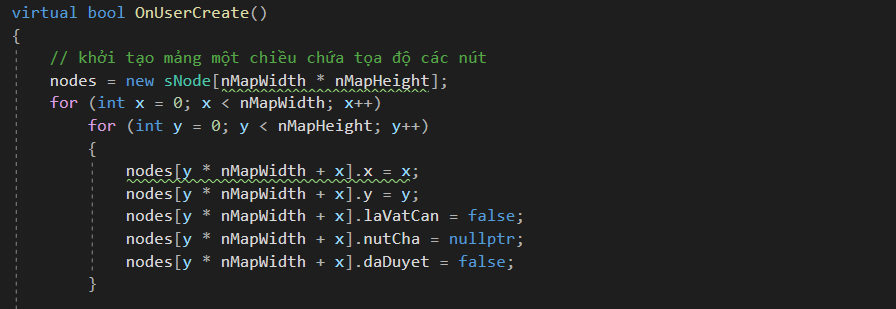
* Khai báo struct sNode gồm các thuộc tính cần thiết của một nút trong bàn cờ

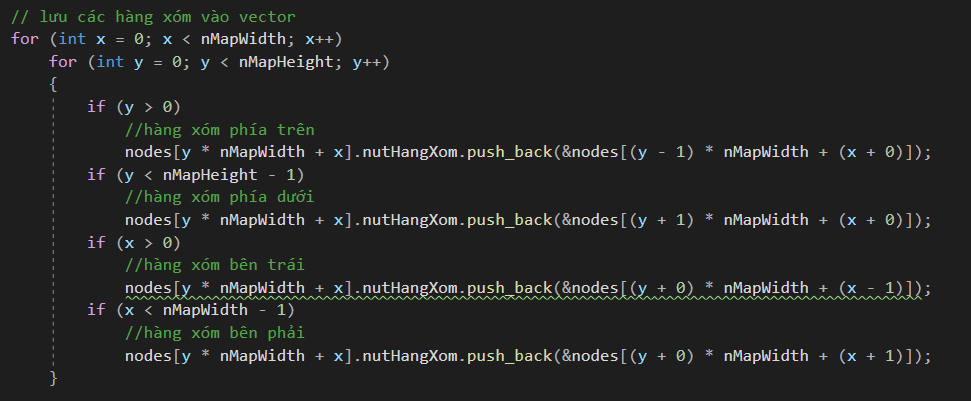


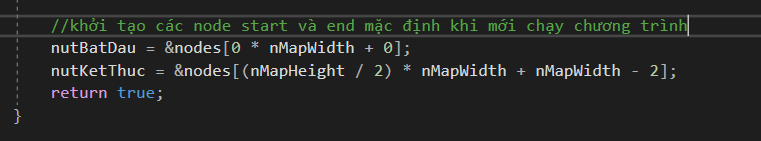
* Khai báo biến lưu các nút trong bàn cờ, nút bắt đầu và kết thúc, kích cỡ bàn cờ là 10x10



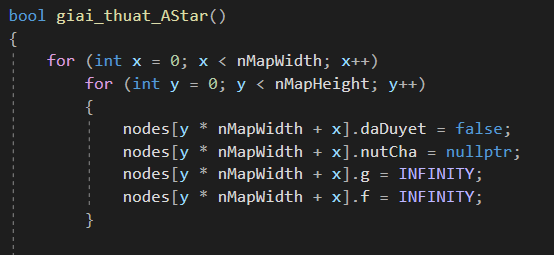
* Hàm OnUserCreate() dùng để khởi tạo các giá trị ban đầu của bàn cờ



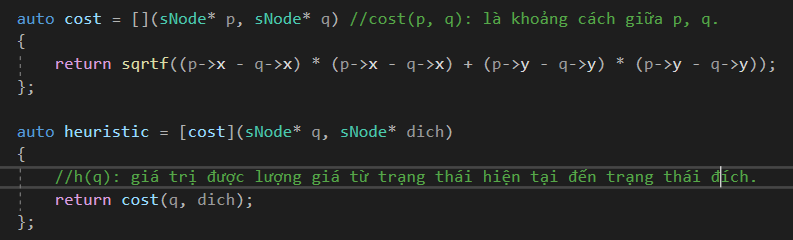




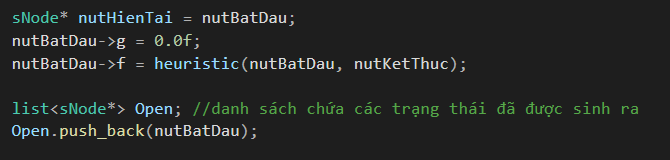
* Hàm giai\_thuat\_AStar() sẽ triển khai thuật toán A-Star

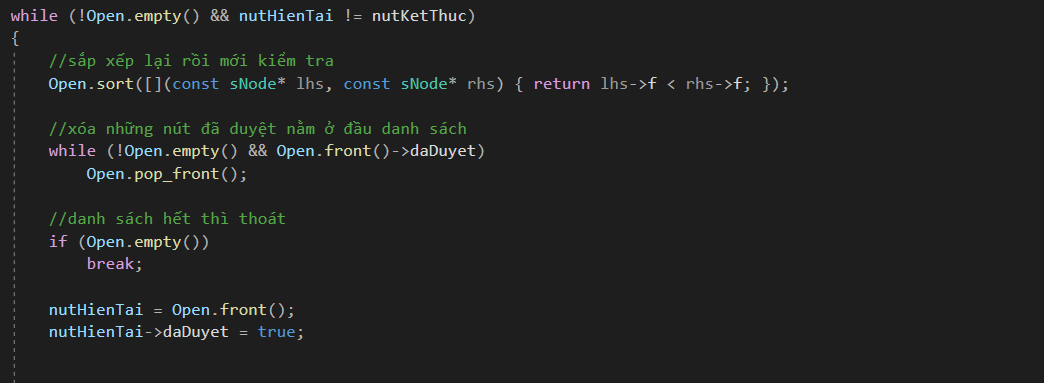


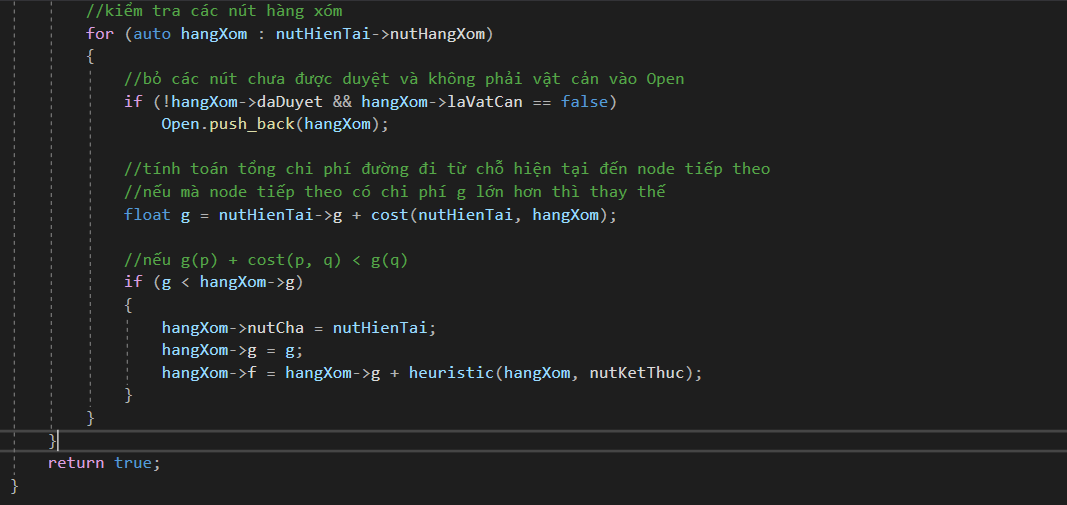
🡪 Reset lại các thuộc tính của các nút để tìm đường đi mới khi người dùng thay đổi trạng thái trên bàn cờ



🡪 Hàm dùng để tính giá trị cost và heuristic (áp dụng công thức Py-ta-go để tính)

🡪 Khởi tạo biến nutHienTai để lưu nút đang được duyệt, đưa nutBatDau vào danh sách Open

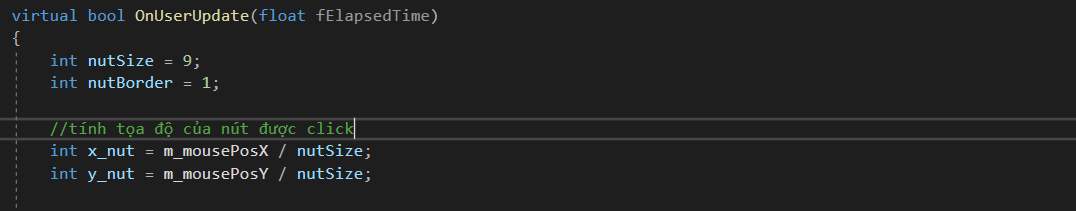


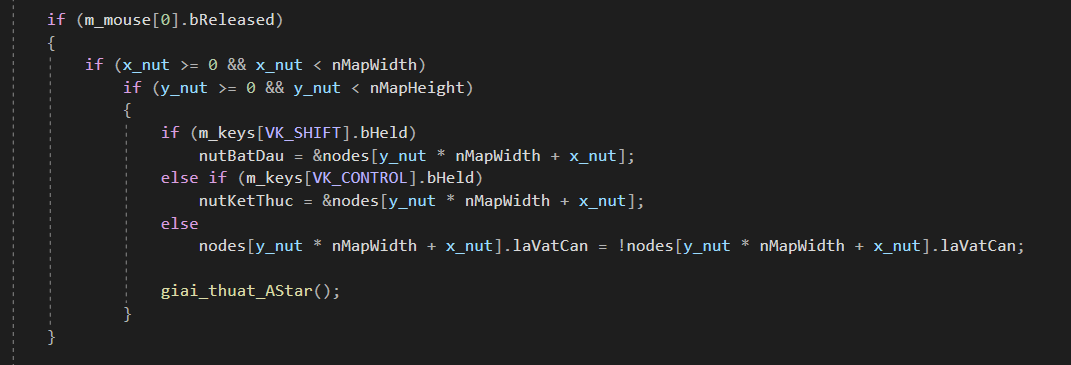


🡪 Duyệt danh sách Open cho đến khi Open rỗng hoặc tìm được nutKetThuc

🡪 Danh sách Open được sắp xếp theo thứ tự giá trị f tăng dần, sau đó lấy nút đầu danh sách để duyệt

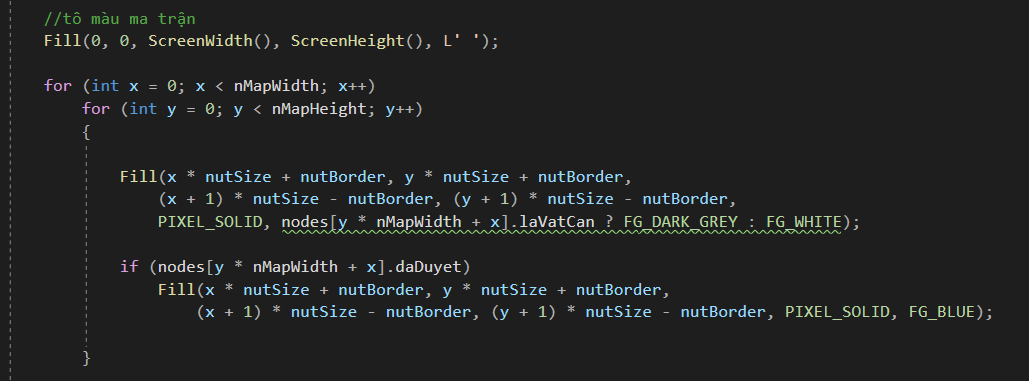
* Hàm OnUserUpdate() dùng để thể hiện hình ảnh trực quan của bàn cờ



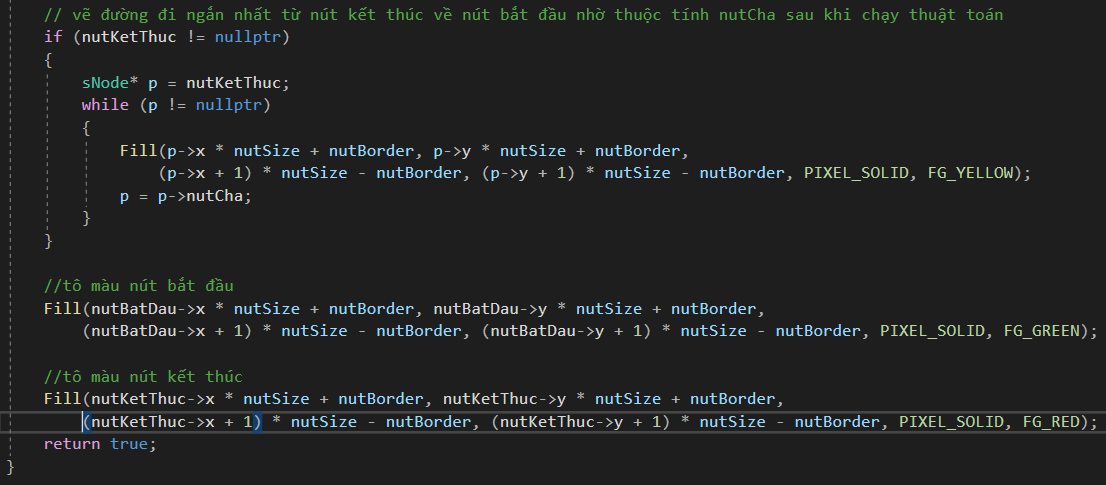


🡪 Tính toán tọa độ ảo của nút được click và đặt lại trạng thái của nó (ấn Shift để chọn nút bắt đầu, Ctrl để chọn nút kết thúc, click trái chuột để chọn nút làm vật cản)

🡪 Sau khi các nút đã được cập nhật trạng thái thì gọi lại hàm giai\_thuat\_AStar() để tìm đường đi

🡪 Dùng hàm Fill() để tô nút:

* Nút vật cản: màu xám
* Nút bình thường: màu trắng
* Nút đã được duyệt qua: màu xanh dương



🡪 Tô màu đường đi từ nút kết thúc trở ngược về nút bắt đầu nhờ thuộc tính nutCha, đường đi tìm được tô màu vàng

🡪 Sau đó tô màu nutBatDau và nutKetThuc:

* Nút bắt đầu: màu xanh lá
* Nút kết thúc: màu đỏ
* Kết quả:

