

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

**Thực Tập Cơ Sở**

**Đề Tài**

* **Chủ đề: Tìm hiểu và cài đặt thuật toán tìm kiếm Greedy Best-First-Search**

 Giáo viên hướng dẫn: Mai Cường Thọ

* Thực hiện đề tài:
* Cao Thái Toàn Phong

MSSV : 60136530

Lớp : 60.CNTT

**Năm học: 2020 - 2021**

1. **Tổng Quan**

A) Người thực hiện

B) Nguồn tài liệu

 C) Giới thiệu

**A) Người thực hiện :**

**1.** Cao Thái Toàn Phong **:** Tìm hiểu thuật toán và biểu diễn thuật toán bằng C# và Windows.Forms

Thực hiện và viết báo cáo.

**B) Nguồn tài liệu tham khảo :**

1. [**https://labs.septeni-technology.jp/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/**](https://labs.septeni-technology.jp/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/)
2. [**https://viblo.asia/p/cac-thuat-toan-co-ban-trong-ai-phan-biet-best-first-search-va-uniform-cost-search-ucs-Eb85omLWZ2G**](https://viblo.asia/p/cac-thuat-toan-co-ban-trong-ai-phan-biet-best-first-search-va-uniform-cost-search-ucs-Eb85omLWZ2G)
3. [**https://huuvinhfit.files.wordpress.com/2015/01/chuong-4-tim-kiem-heuristic.pdf**](https://huuvinhfit.files.wordpress.com/2015/01/chuong-4-tim-kiem-heuristic.pdf)
4. [**https://www.youtube.com/watch?v=yH3YzGH3TrQ**](https://www.youtube.com/watch?v=yH3YzGH3TrQ)

**C) Giới Thiệu :**

* Mục đích đề tài : Vận dụng kiến thức lập trình hướng đối tượng, toán rời rạc để giải quyết đề bài.
* Nội dung
* Tìm hiểu tổng quan bài toán tìm kiếm dựa trên kinh nghiệm (Heuristic Search), và cụ thể cho thuật toán tìm kiếm Greedy Best-First-Search và phân tích cài đặt.
* Tổ chức dữ liệu và cài đặt được thuật toán ứng dụng cho một bài toán cụ thể.
* Tìm Hiểu Chung

Trong lĩnh vực trí tuệ nhân tạo người ta sử dụng những khái niệm, những thuật toán liên quan đến Tìm kiếm (search, query, sort,…) rất nhiều. Về vấn đề tìm kiếm,ta có các điển hình là :**Tìm kiếm theo cấu trúc cây** với chiến lược tìm kiếm là xác định thứ tự xét các nút của cây và **Tìm kiếm với tri thức bổ sung** (uninformed search strategies) , (informed search strategies). Để tìm hiểu kĩ hơn ta cùng phân tích các nội dung ở các mục sau.

**II. Thuật Toán Tìm Kiếm**

**A.Phân loại**

Có nhiều cách phân loại các thuật toán tìm kiếm, trong đó phổ biến hơn cả đó là chia làm 2 loại: **Uninformed Search (**Blind Search**)** và **Informed Search**( Heuristic Search). *Uninformed Search (*tìm kiếm mù *)là dạng bài toán biết trước đích đến. Ta sẽ đi mãi cho đến khi nào tìm được tới đích mà thôi.* Phương pháp tìm kiếm mù không hiệu quả trong hầu hết bài toán, chậm và may rủi.

Các dạng phổ biến:

* DFS (Depth First Search): Tìm kiếm theo chiều sâu
* BFS (Breath First Search): Tìm kiếm theo chiều rộng
* UCS (Uniform Cost Search): Áp dụng thuật toán Dijkstra

Do những phức tạp , tính hiệu quả thấp và có nhiều sai lệch trong kết quả tìm kiếm, người ta sử dụng 1 thuật toán hiệu quả hơn, đó là Heuristic Search

## B) . Tổng quan bài toán Informed Search (Heuristic Search / Tìm kiếm kinh nghiệm)

**Heuristic:**là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu. Các phương pháp heuristic được dùng nhằm tăng nhanh quá trình tìm kiếm với các giải pháp hợp lý để giảm bớt việc nhận thức vấn đề khi đưa ra quyết định. Hai tính chất quan trọng của một heuristic là admissible và consistent với :

[**admissible**](https://en.wikipedia.org/wiki/Admissible_heuristic) ( tính khả nhất -chấp nhận được ) : trong các  thuật toán liên quan đến [tìm đường](https://en.wikipedia.org/wiki/Pathfinding) , một [hàm heuristic](https://en.wikipedia.org/wiki/Heuristic_function) được cho là có thể chấp nhận được nếu nó không bao giờ đánh giá quá cao chi phí đạt được mục tiêu, tức là chi phí mà nó ước tính để đạt được mục tiêu không cao hơn chi phí thấp nhất có thể từ hiện tại chỉ trong đường dẫn.

[**consistent**](https://en.wikipedia.org/wiki/Consistent_heuristic) ( tính đơn nhất - nhất quán , hoặc đơn điệu) : trong các [thuật toán](https://en.wikipedia.org/wiki/Algorithm) liên quan đến [tìm đường](https://en.wikipedia.org/wiki/Pathfinding) , một [hàm heuristic](https://en.wikipedia.org/wiki/Heuristic_function) được cho là  nhất quán , hoặc đơn điệu , nếu ước tính của nó luôn nhỏ hơn hoặc bằng khoảng cách ước tính từ bất kỳ đỉnh lân cận nào đến mục tiêu, cộng với chi phí đạt được hàng xóm đó.

**Heuristic function**: Hàm đánh giá dựa trên kinh nghiệm, dựa vào đó để xếp hạng thứ tự tìm kiếm, cách chọn hàm đánh giá quyết định nhiều đến kết quả tìm kiếm.Hai thuật toán sử dụng hàm đánh giá phổ biến là: Greedy Best-First-Search và A\*

Tuy nhiên,với đề bài yêu cầu, ta chủ yêu nghiên cứu về thuật toán **Greedy Best-First-Search**  với **Greedy** mang ý nghĩa là ( ***Greedy algorithm*** *\_* chiến lược tham lam : thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục. )

**III**) **Nghiên cứu về Greedy Best-First-Search**

**A ) Định nghĩa :**  thuật toán **Greedy best first search (G-BFS)** là một chiến lược tìm kiếm với tri thức bổ sung từ việc sử dụng các tri thức cụ thể của bài toán, là sự kết hợp giữa hàm đánh giá với tìm kiếm theo chiều rộng. Thuật toán sẽ sử dụng 1 hàm đánh giá là heuristic, kí hiệu là h(n),hàm này sẽ đánh giá chi phí ( giá/ cost ) để đi từ nút ( Node ) hiện tại n đến nút đích (mục tiêu) sao cho tốt nhất với yêu cầu của người sử dụng.

Trong tìm kiếm sử dụng kinh nghiệm,hàm đánh giá có vai trò rất quan trọng. Xây dựng được hàm đánh giá tốt sẽ giúp việc tìm kiếm được thuận lợi và thực hiện một cách nhanh chóng và ngược lại. Không có hàm đánh giá chung cho mọi bài toán vì nó phụ thuộc vào vấn đề cần giải quyết của từng loại bài toán. Trong thuật toán tìm đường đi trên bản đồ, ta có thể xây dựng hàm đánh giá như : Đường chim bay từ địa chỉ này đến địa chỉ khác hoặc khoảng cách thực trên đường đi thực tế giữa các địa chỉ.

Tư tưởng của thuật toán **G-BFS** là việc tìm kiếm bắt đầu tại nút gốc và tiếp tục bằng cách duyệt các nút tiếp theo. Nút được lựa chọn ở các bước kế tiếp là nút có giá đến trạng thái đích tốt nhất theo yêu cầu bài toán. Tức là chọn nút gần với trạng thái đích nhất so với các nút còn lại nằm trong hàng đợi để phát triển thuật toán.

**B ) Mã giả:**

procedure Best\_First\_Search;

begin

Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

loop do

if L rỗng then

{thông báo thất bại; stop};

Loại trạng thái u ở đầu danh sách L;

if u là trạng thái kết thúc then

{thông báo thành công; stop};

for mỗi trạng thái v kề u do

Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần

của hàm đánh giá;

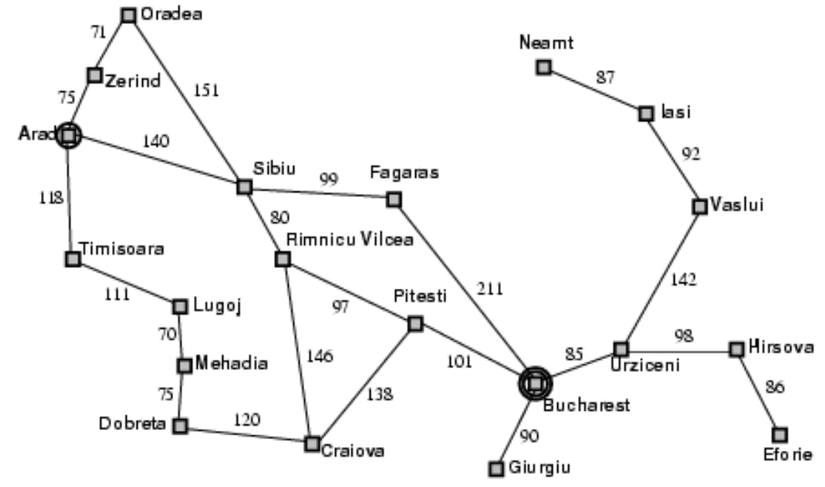
end;

| **C) Đánh giá thuật toán** | Với d, m là độ sâu của cây cần duyệt và mỗi trạng thái khi được phát triển sẽ sinh ra b trạng thái kề (b được gọi là nhân tố nhánh). |
| --- | --- |
| **Chiến lược**  **Tính hoàn chỉnh** | Tìm kiếm kinh nghiệm, sử dụng hàm đánh giá  Không, Vì có thể vướng (chết tắc) trong các vòng lặp kiểu như :Iasi -> Neamt->Iasi ->Neamt… |
| **Chi phí** | Tính từ node trước đến node hiện tại |
| **Độ phức tạp theo thời gian** | O(b^m) |
| **Độ phức tạp theo không gian**  **( bộ nhớ )** | O(b^m) |
| **Danh sách các node chờ duyệt** | Hàng đợi ưu tiên |
| **Khả năng tìm nghiệm** | Quá trình tìm kiếm có thể đi xa khỏi lời giải. Kỹ thuật này chỉ xét một phần của không gian và coi đó là phần hứa hẹn hơn cả. Có thể bị kẹt trong vòng lặp. |
| **Tính tối ưu** | Không |
| **Ghi chú** | Một hàm đánh giá tốt có thể giảm thời gian và không gian nhớ một cách đáng kể. |

**D) Ví dụ minh họa :**

**1.Các ví dụ cơ bản**

**VÍ DỤ 1 : Bài toán tìm đường đi từ Arad đến Bucharest**



**Bài toán sử dụng: hSLD**

(n)= Ước lượng khoảng cách đường thẳng (“chim bay”)từ thành phố hiện tại (Arad ) n đến Bucharest . Phương pháp tìm kiếm Greedy bestfirst search sẽ xét (phát triển) nút “có vẻ” gần với nút đích (mục tiêu) nhất.

**Bước 1 :** Xét Arad = 366

**Bước 2 :** từ Arad xét có 3 đường đi với khoảng cách lần lượt là : Zerind( 374 ) ,Sibiu(253),Timisoara(329) với Sibiu là nút gần nhất nên được xét tiếp.

**Bước 3 :** từ Sibiu xét có 4 đường đi với khoảng cách lần lượt là :

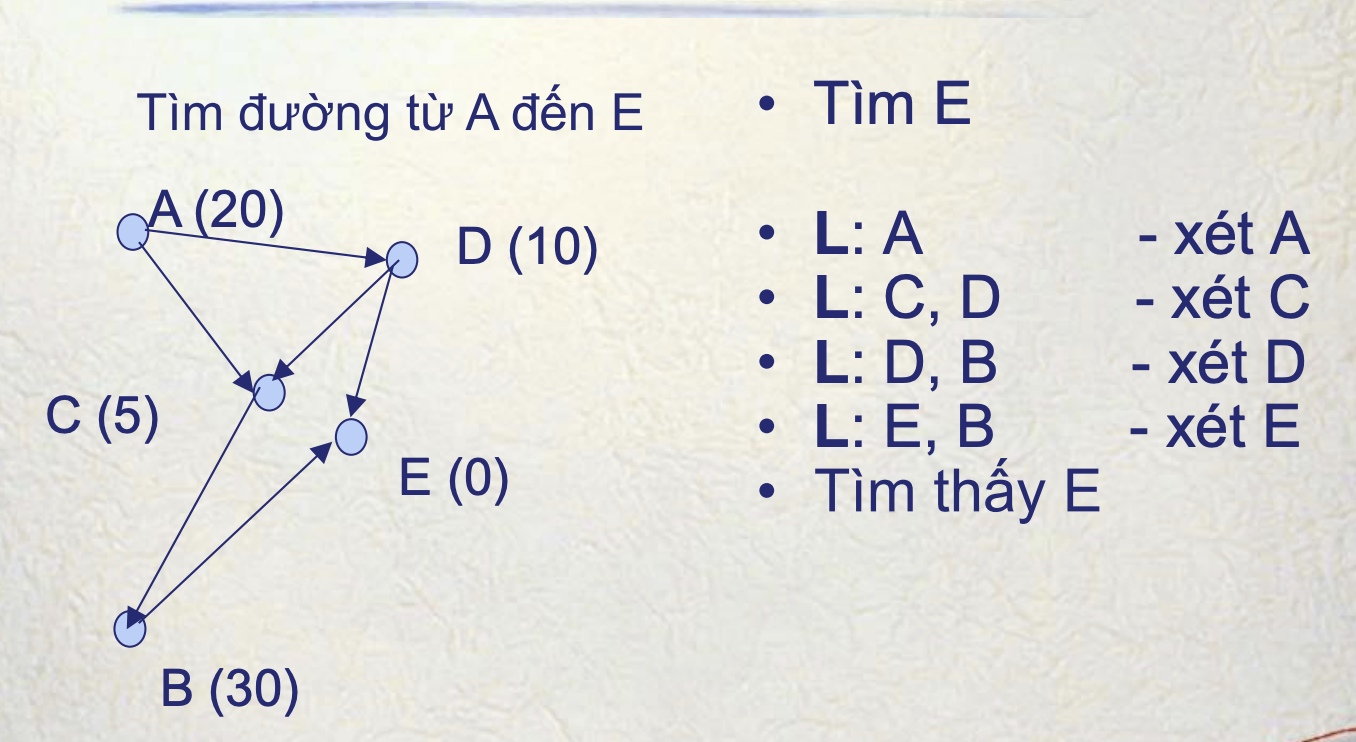
Arad(366), Faragas(176),Oradea(380),Rimnicu Vilcea(193) với Faragas là nút gần nhất nên được xét tiếp.

**Bước 4:** từ Faragas xét có 2 đường đi với khoảng cách lần lượt là :

Sibiu(253),Bucharest(0) với Bucharest là thành phố gần nhất và cũng là điểm nút cần tìm nên thuật toán sẽ cho kết quả đường đi ngắn nhất từ thành phố (Arad ) đến Bucharest là :

**G-BFS**:  **Arad --> Sibiu --> Faragas --> Bucharest**

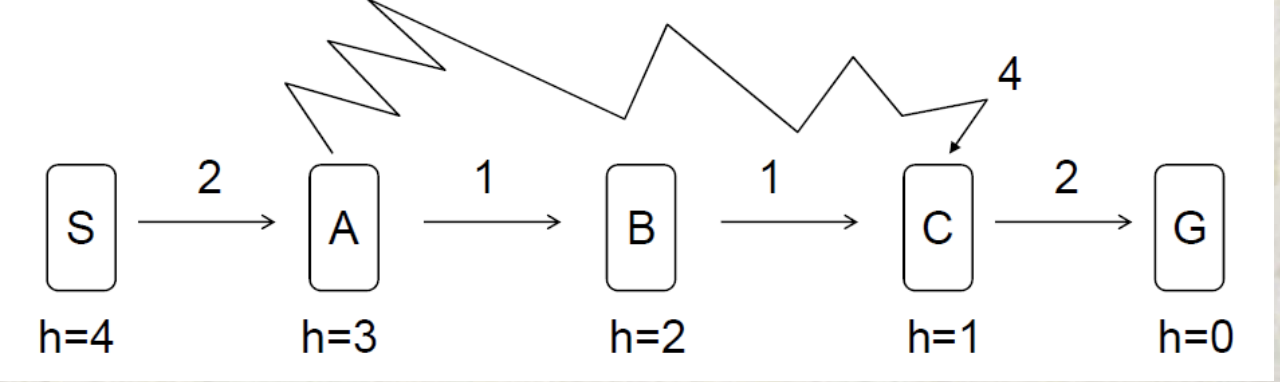
**VÍ DỤ 2 : Áp dụng kiến thức môn toán rời rạc và duyệt đường đi từ A 🡪 E . Hình :**



* Chọn node kế tiếp có được đánh giá là tốt nhất

Giá trị của hàm đánh giá tại 1 điểm được ghi bên cạnh: A(20), C(5)  
Nghĩa là nó đánh giá dựa vào 1 tính chất nào đó mà AE = 20, CE = 5, EE = 0 (tất nhiên)  
**G-BFS**: **A** --> **C** --> **D** --> **E**

VÍ DỤ 3 : Tầm quan trọng của [**admissible**](https://en.wikipedia.org/wiki/Admissible_heuristic) và [**consistent**](https://en.wikipedia.org/wiki/Consistent_heuristic)

**Yêu cầu : Tìm đường từ S 🡪 G**

Hình bên, g(n) là chi phí thực tế, h(n) là ước lượng theo hàm đánh giá.  
Nếu chỉ dựa theo đánh giá h(n): G-BFS: **S**->**A**->**C**->**G** (cost 8)  
Trong khi đi thẳng không nghĩ ngợi: **S**->**A**->**B**->**C**->**G** (cost 6)

**G-BFS** hiệu quả nhưng thật sự chưa chắc đã tối ưu.

Để tránh điều đó, hàm đánh giá cần thoả mãn 2 tính chất quan trọng của một heuristic : [**admissible**](https://en.wikipedia.org/wiki/Admissible_heuristic) và [**consistent**](https://en.wikipedia.org/wiki/Consistent_heuristic)  như đã nêu ở trên mục II-B.

5) Biểu diễn thuật toán bằng C# và WindowForms

File đính kèm với bản word

5.1) Cấu trúc chương trình

Các lớp của chương trình

* Lớp Node :

Trước hết ta xây dựng một list Node để lưu trữ ,sắp xếp và so sánh các nút ( Node), tức là dùng tới phương thức Sort. Phương thức Sort() của List<> sẽ chỉ có thể thực hiện được trên những class có cài đặt interface IComparable. Và tất nhiên lớp Node sẽ được gọi ở các giai đoạn sau của bài toán nên ta khai báo Public.

//Mã :

public class Node:IComparable

{

public List<Node> Next = new List<Node>();

public Node came\_from = null;//đỉnh đích

public string name; //tên đỉnh.

public double h = 0;

//chi phí h ước lượng đến đích

public Node()

{

Next.Clear();

this.came\_from = null; //gán giá trị của đỉnh đích là rỗng

}

public int CompareTo(object obj)

/\* CompareTo .Tham số nhận vào là một kiểu object , ta thực hiện việc so sánh và sắp xếp 2 đối tượng bằng if else đơn giản \*/

{

Node node = obj as Node;

if (this.h > node.h)

return 1;

else

{

if (this.h == node.h)

return 0;

else

return -1;

}

}

/\* Vì chưa biết trước được số lượng phần tử người dùng sẽ truyền vào nên ta dùng param trong Tạo liên kết 2 chiều.Vì biến bên ngoài thay đổi sau khi ra khỏi hàm, chúng ra sẽ truyền đối số vào hàm theo cách **truyền tham chiếu : ref \*/**

/// <summary>

///Tao lk

/// </summary>

/// <param name="node"></param>

public void MakeLink(ref Node node)

{

this.Next.Add(node);

node.Next.Add(this);

}

* Lớp BestFirstSearch

// khởi tạo danh sách

public class BestFirstSearch

{

public static List<Node> Find(Node start, Node Target)

{

//khởi tạo danh sách L chỉ chứa trạng thái ban đầu.

List<Node> L = new List<Node>();

L.Add(start);

List<Node> Visited = new List<Node>();

//vòng lặp

while (true)

{

// nếu L rỗng thì thông báo tìm kiếm thất bại và dừng

if(L.Count == 0)

{

MessageBox.Show("Error!");

//MessageBox sẽ hiển thị nếu lỗi

return Visited;

}

//Loại trạng thái u ở đầu danh sách L

Node node = new Node();

//Gán tên nút bằng phần tử L[0]

node.name = L[0].name;

node.h = L[0].h;

//Dùng vòng lặp để duyệt tất cả các trường hợp của bài toán

for (int i = 0; i < L[0].Next.Count; i++)

{

node.Next.Add(L[0].Next[i]);

}

L.RemoveAt(0);

//nếu u là trạng thái kết thúc thì thông báo thành công

if(node.name == Target.name)

{

Visited.Add(node);

MessageBox.Show("Success!");

//MessageBox sẽ hiển thị nếu thành công

return Visited;

}

/\* for mỗi trạng thái v kề u do ,xen v vào danh sách L sao cho L được sắp xếp theo thứ tự tăng dần của hàm đánh giá và kết thúc \*/

if(node.Next.Count != 0)

{

Visited.Add(node);

for (int i = 0; i < node.Next.Count; i++)

{

L.Add(node.Next[i]);

}

L.Sort();

}

}

}

}

IV ) BIỂU DIỄN THUẬT TOÁN BẰNG WINDOWS FORM

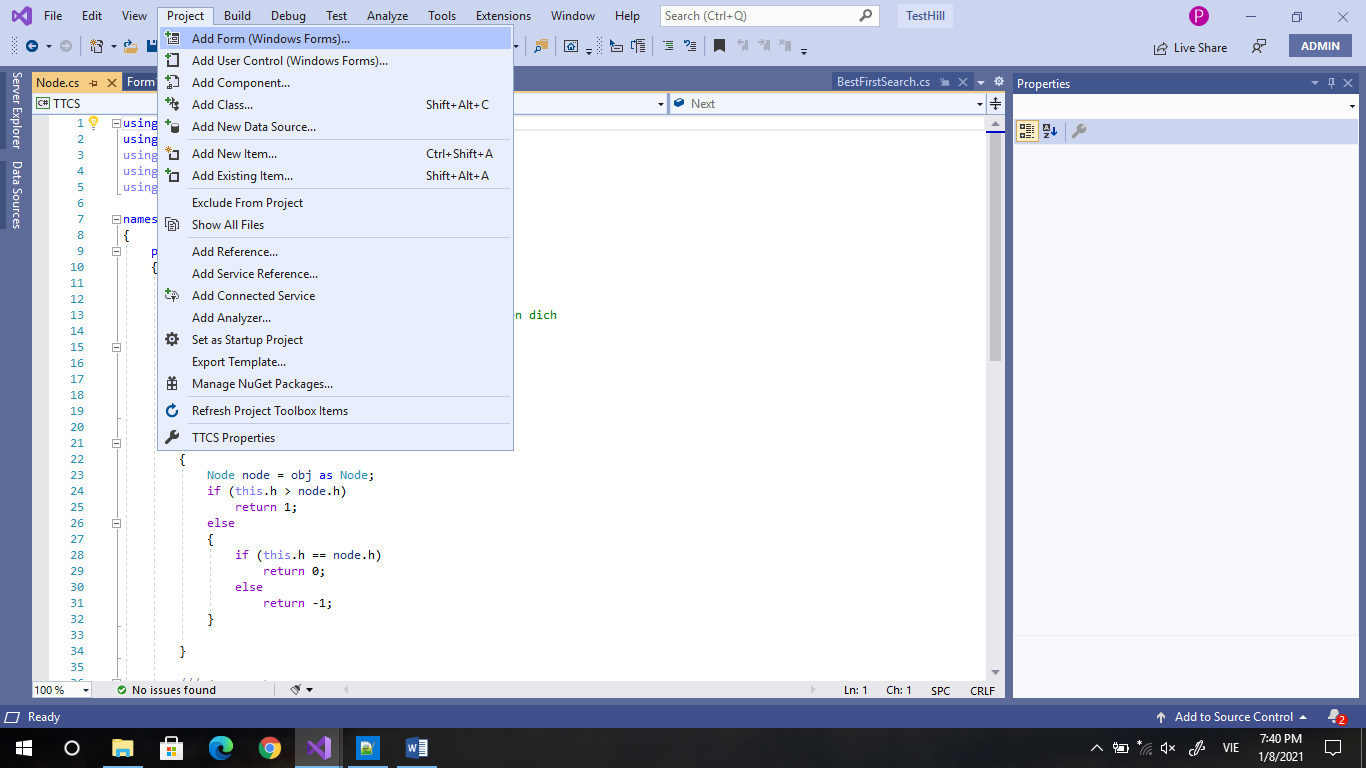
A.Khai báo

using System.Drawing;

using System.Drawing.Drawing2D;

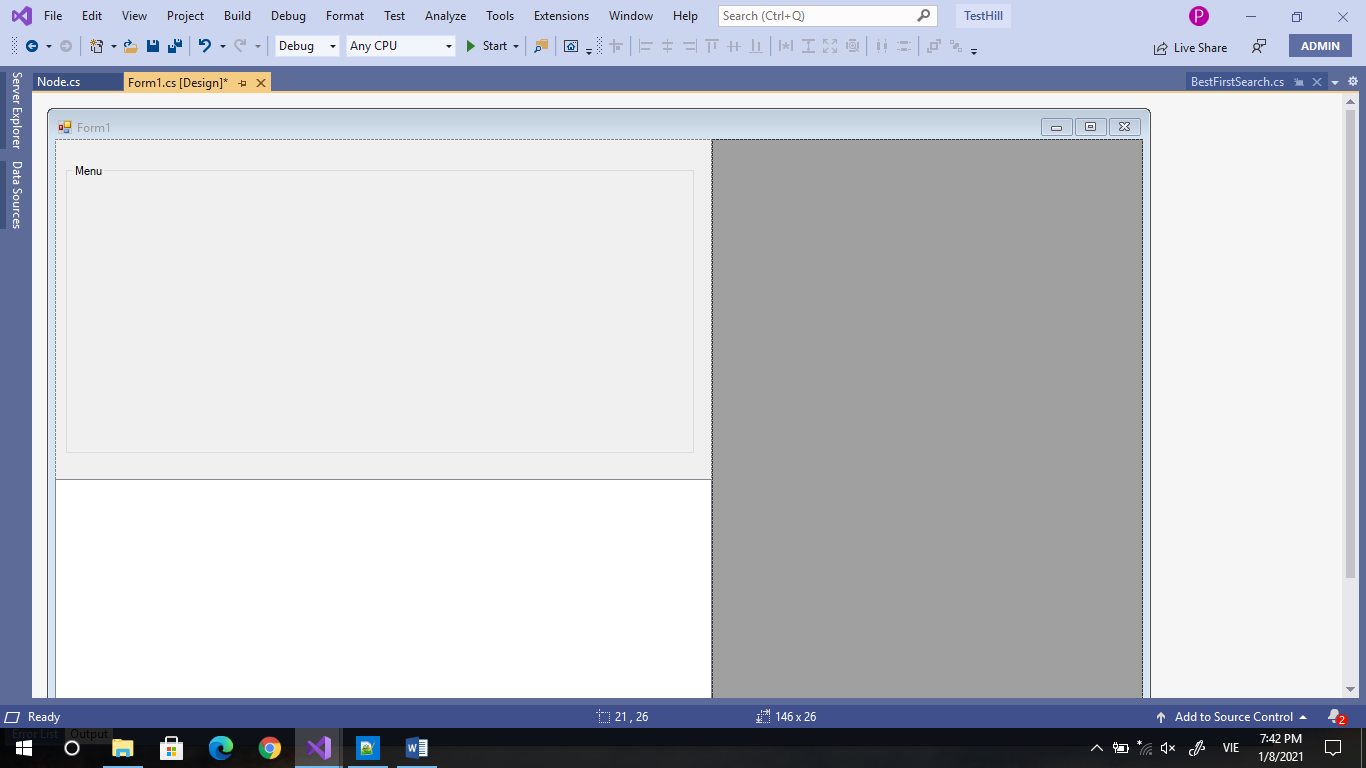
using System.Linq;

using System.Windows.Forms;

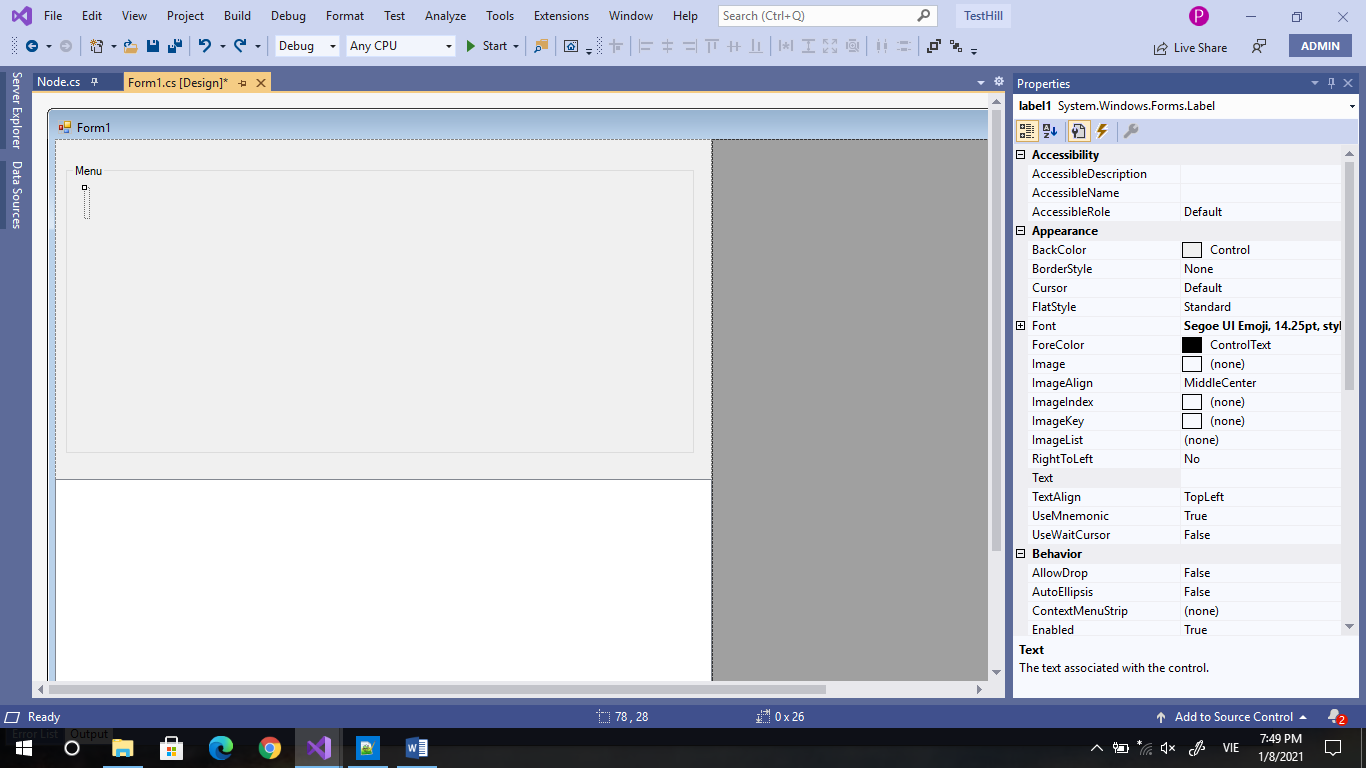


1.Tạo Menu

Từ màn hình làm việc ta thêm 1 Form

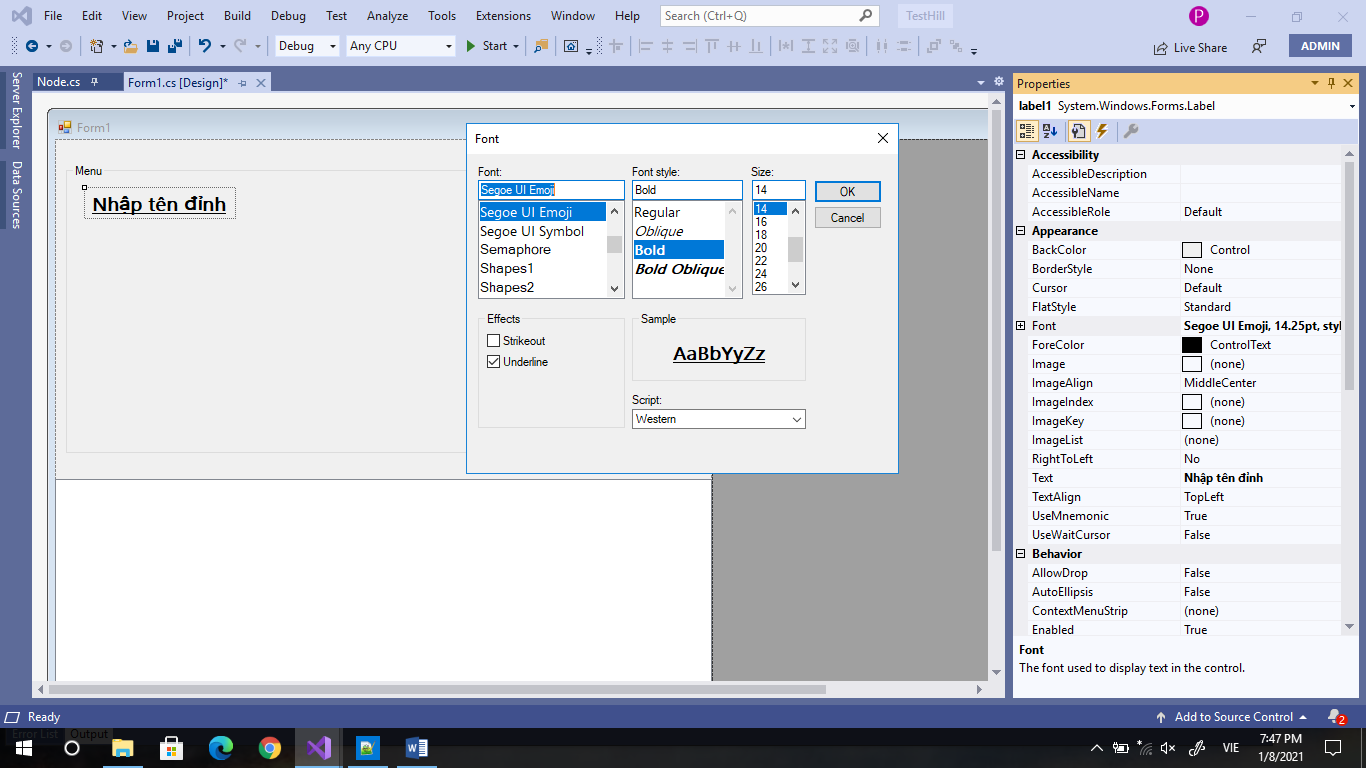


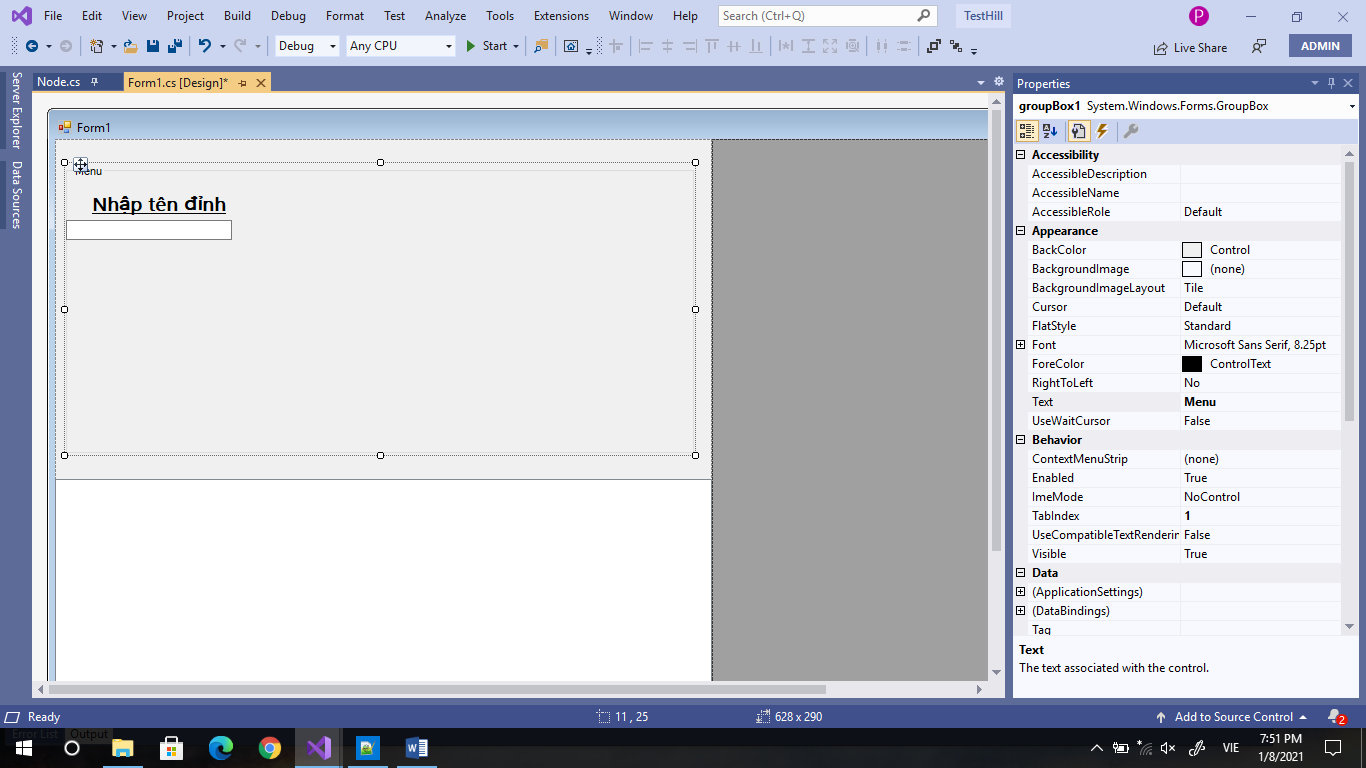
2.Tạo Label 1



Tại text đổi tên Label thành : Nhập Tên Đỉnh

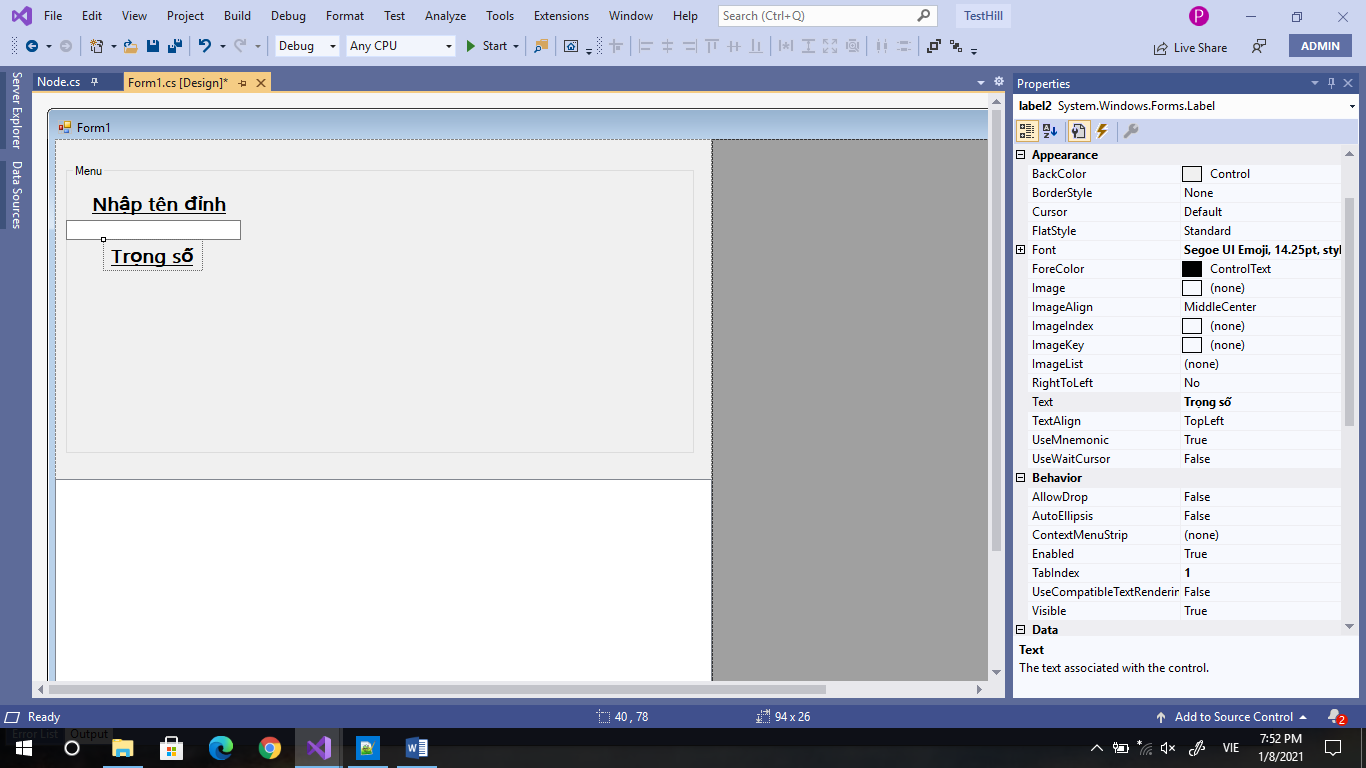
Set: phont chữ : Segoe UI Emoji Font Style : Bont và size :14, lấy font này là mặc định cho các Label tiếp theo

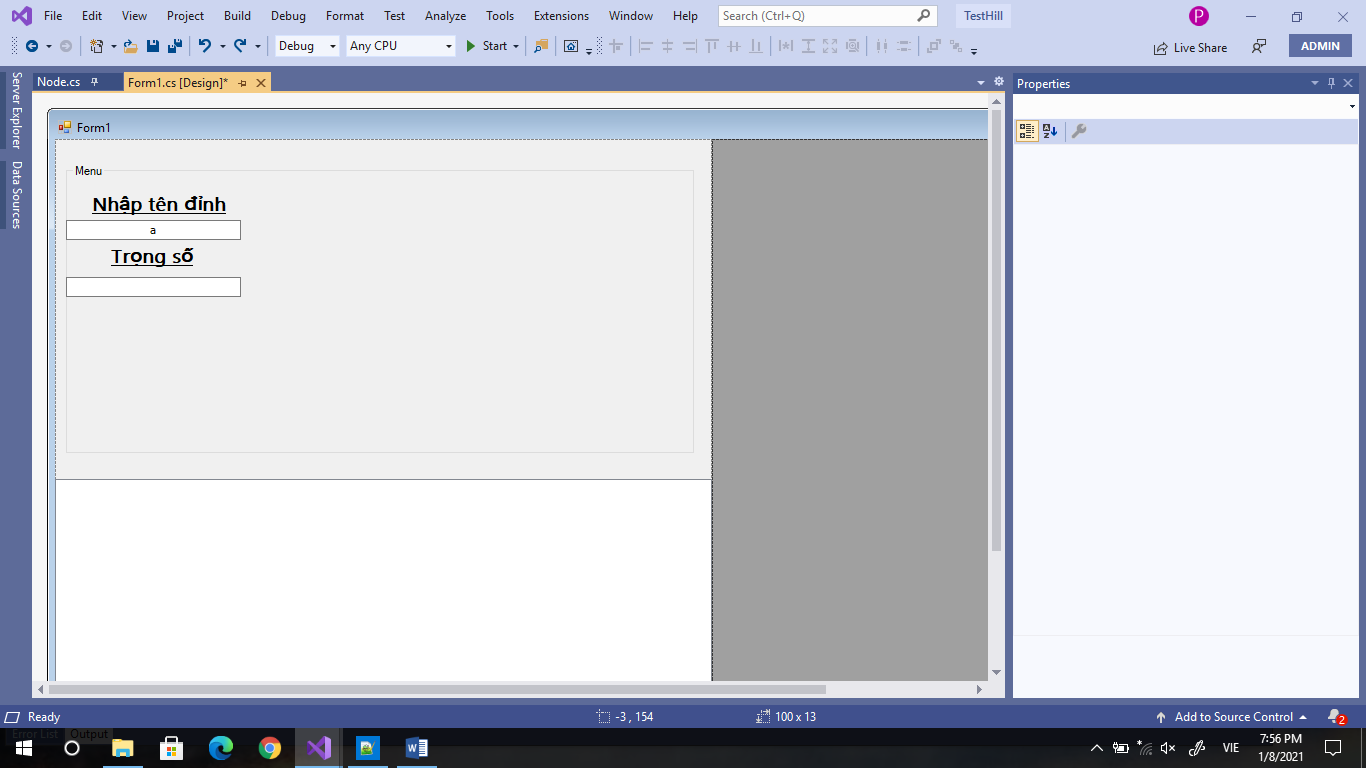


Thêm ô nhập liệu cho Label 1

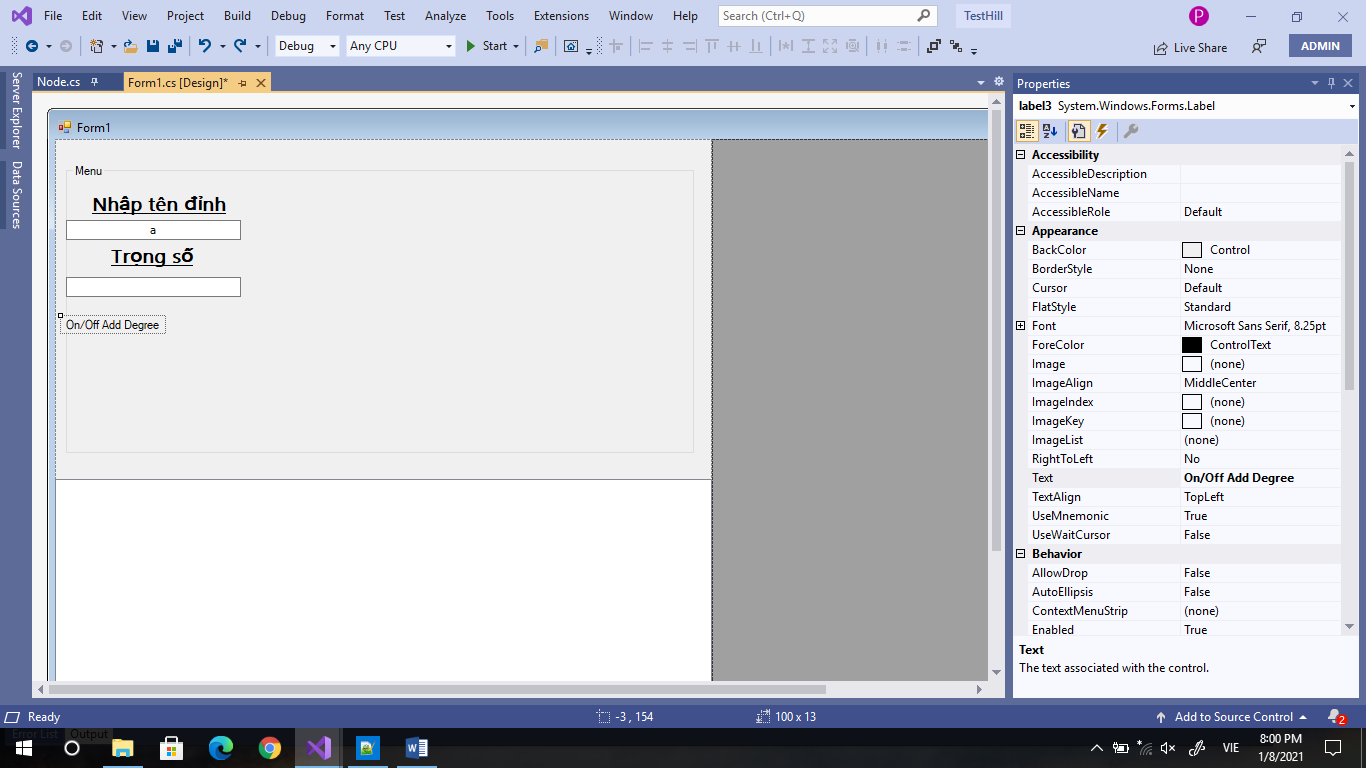
ở ô nhập liệu ,ta có thể nhập các kí tự chữ cái viết tường và viết hoa, ví dụ : a,A,b,B,c,C,..

3.Set tương tự như Label 1, ta tiếp tục cài đặt Label 2 đặt tên là Trọng số

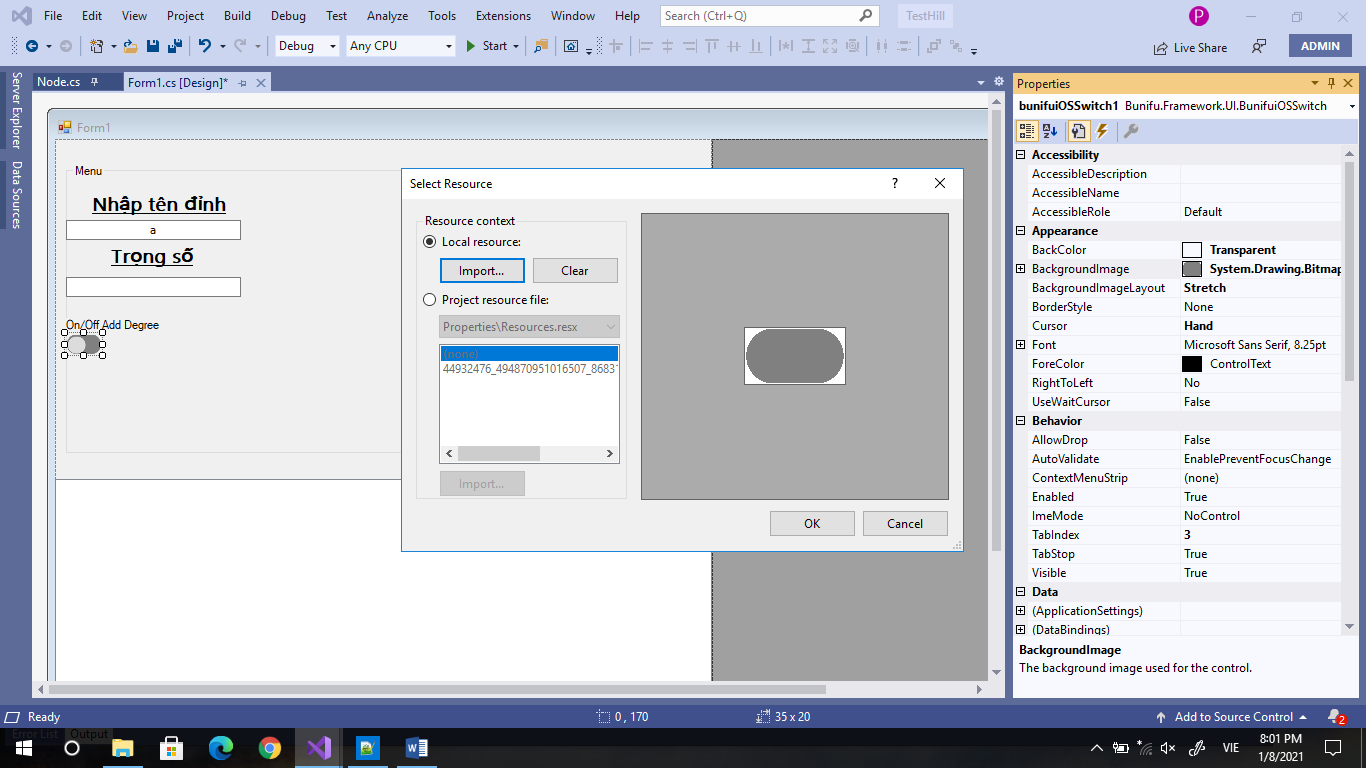




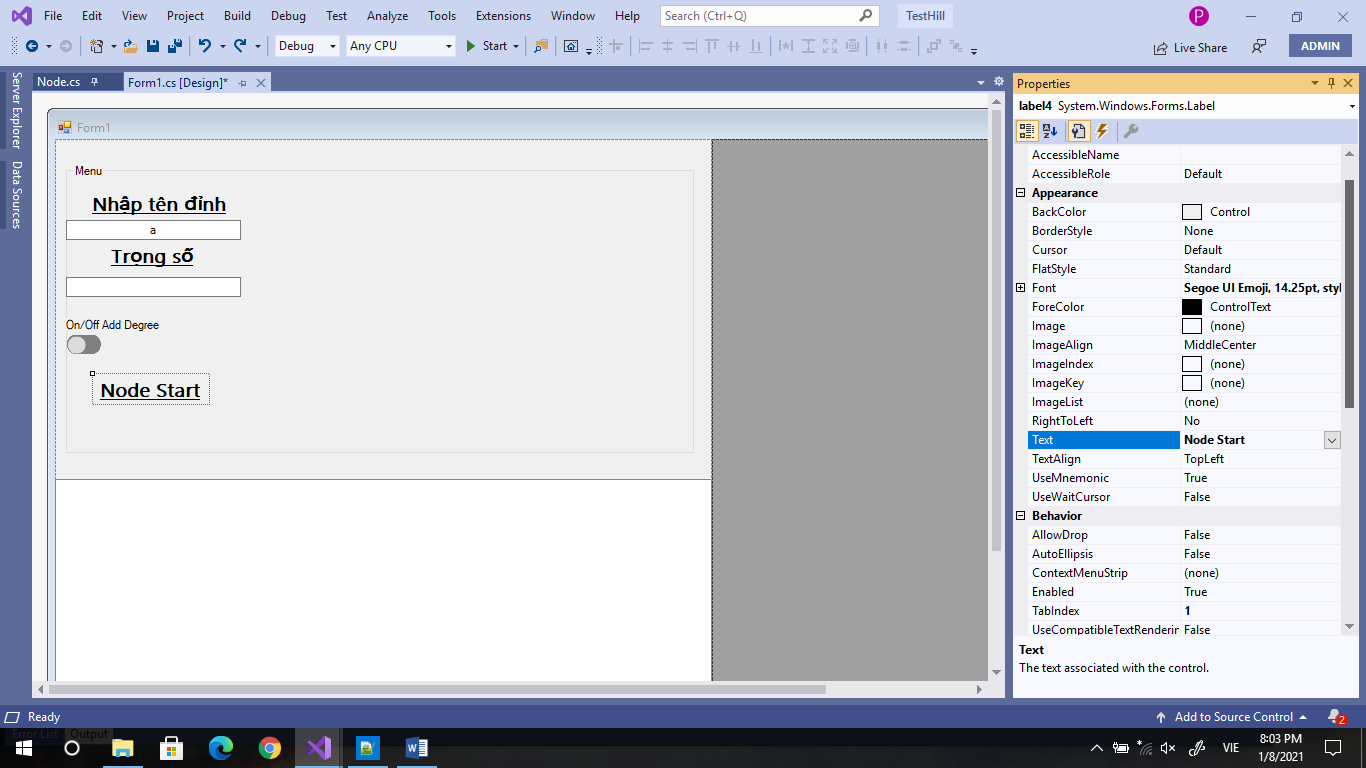
4.Set tương tự như Label 1, ta tiếp tục cài đặt Label 3 đặt tên là On/off Add Degree

Đây là công tắc cho phép sau khi nhập tên đỉnh và trọng số ta có thể cho phép hoặc tắt đi việc vẽ đỉnh đó kèm trọng số lên màn hình

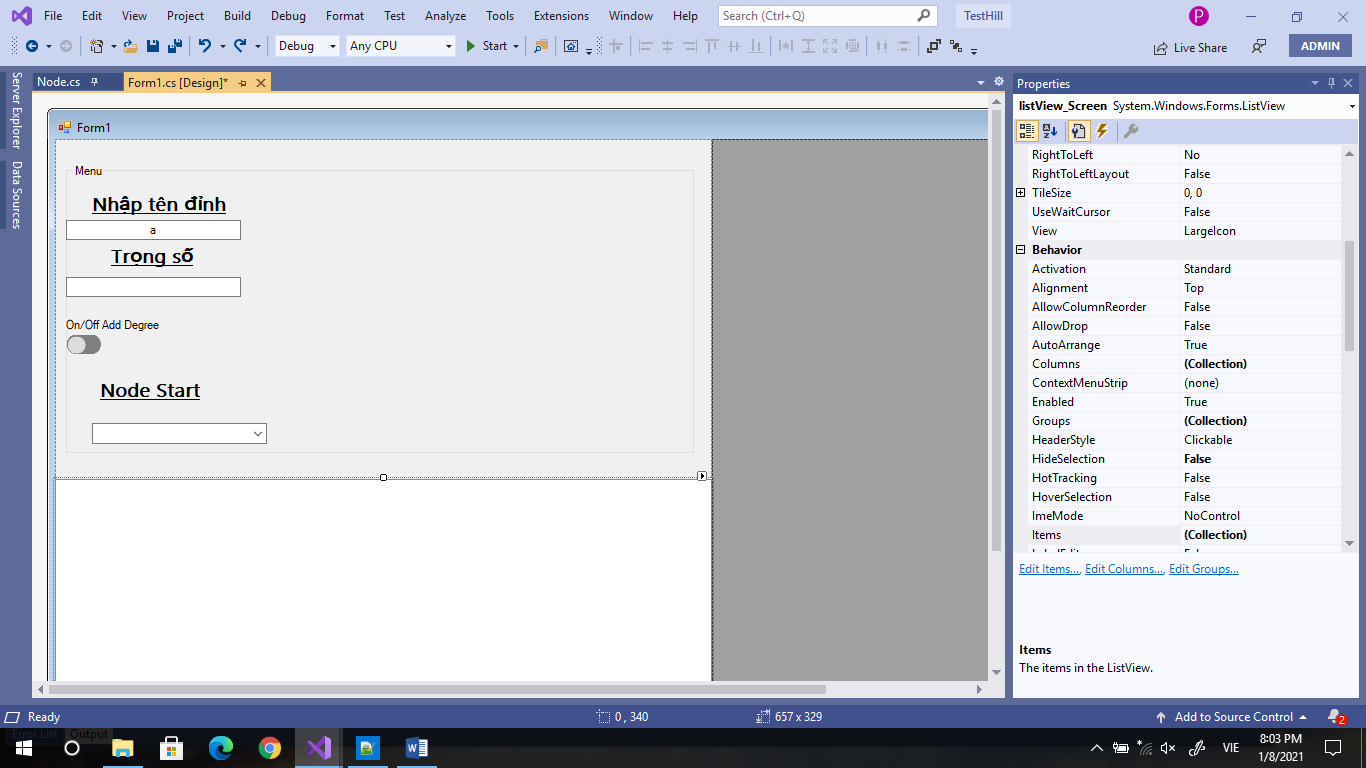
Thêm và set công tắc



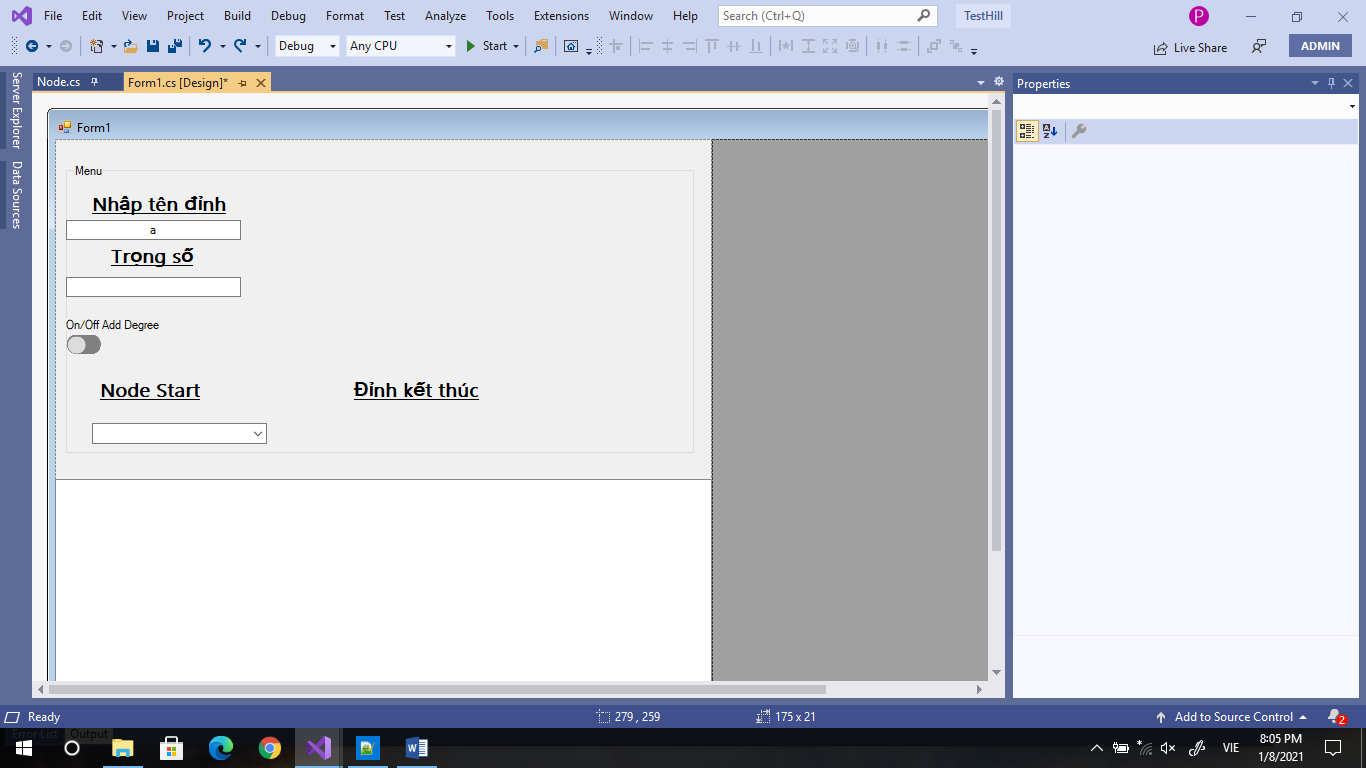
5. Tạo Label 3: Node Start( nút bắt đầu) để nhập vào núp bắt đầu.



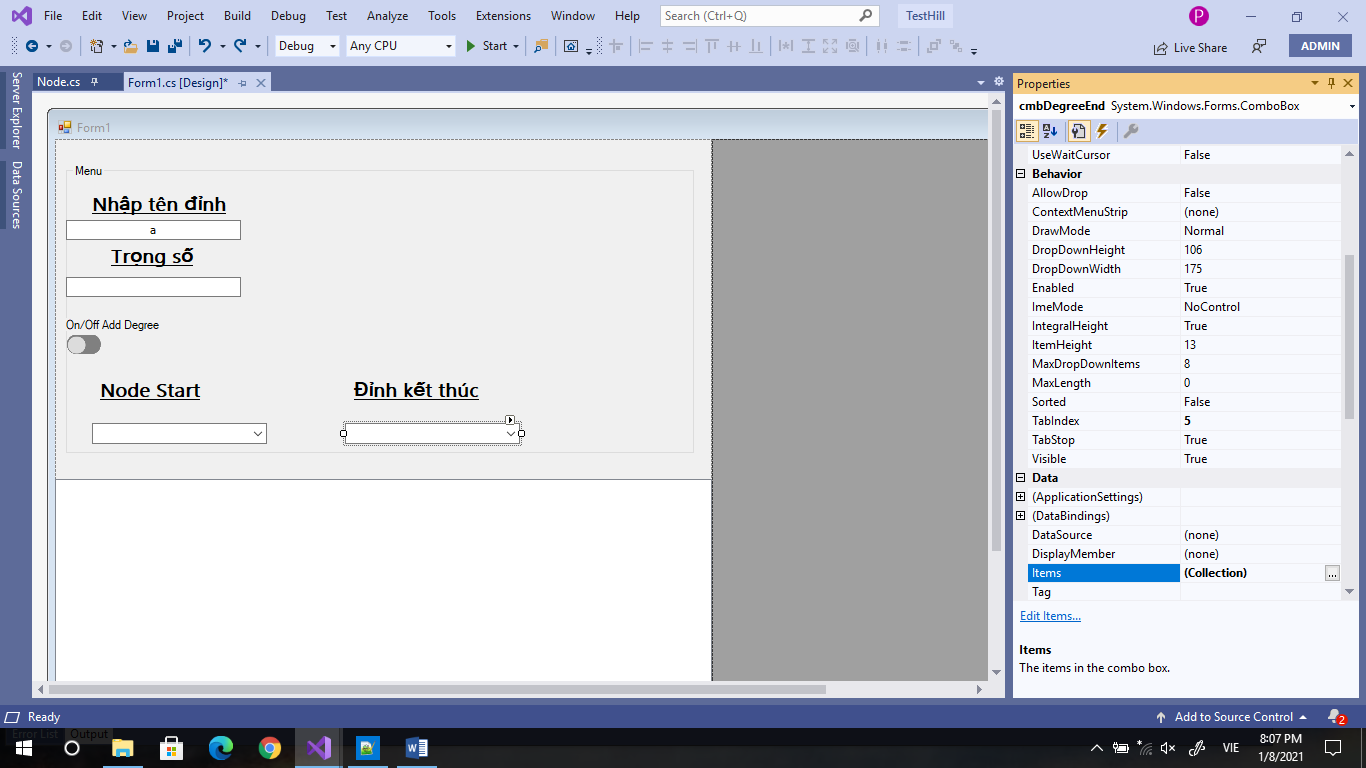
Thêm ô nhập liệu



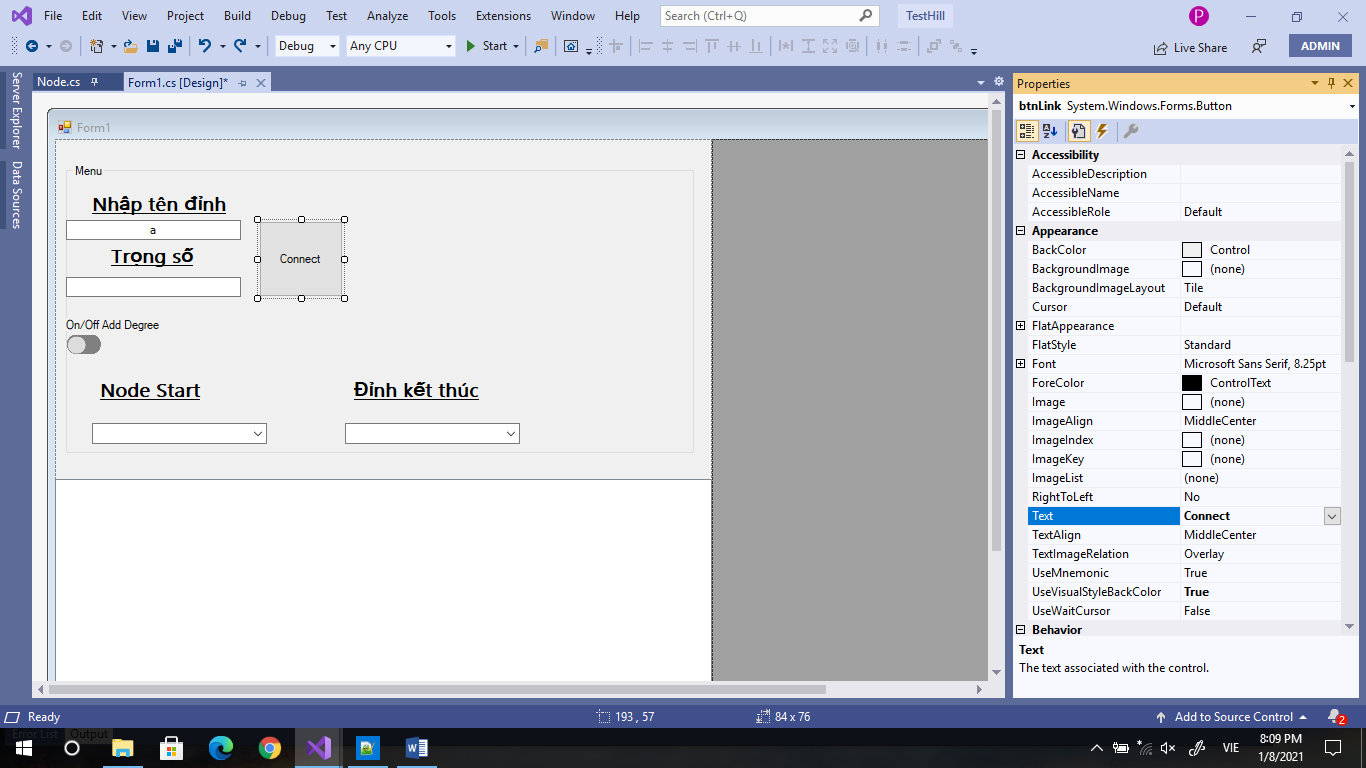
6. Tạo Label 4: Đỉnh kết thúc( nút mục tiêu) để nhập vào nút mục tiêu.



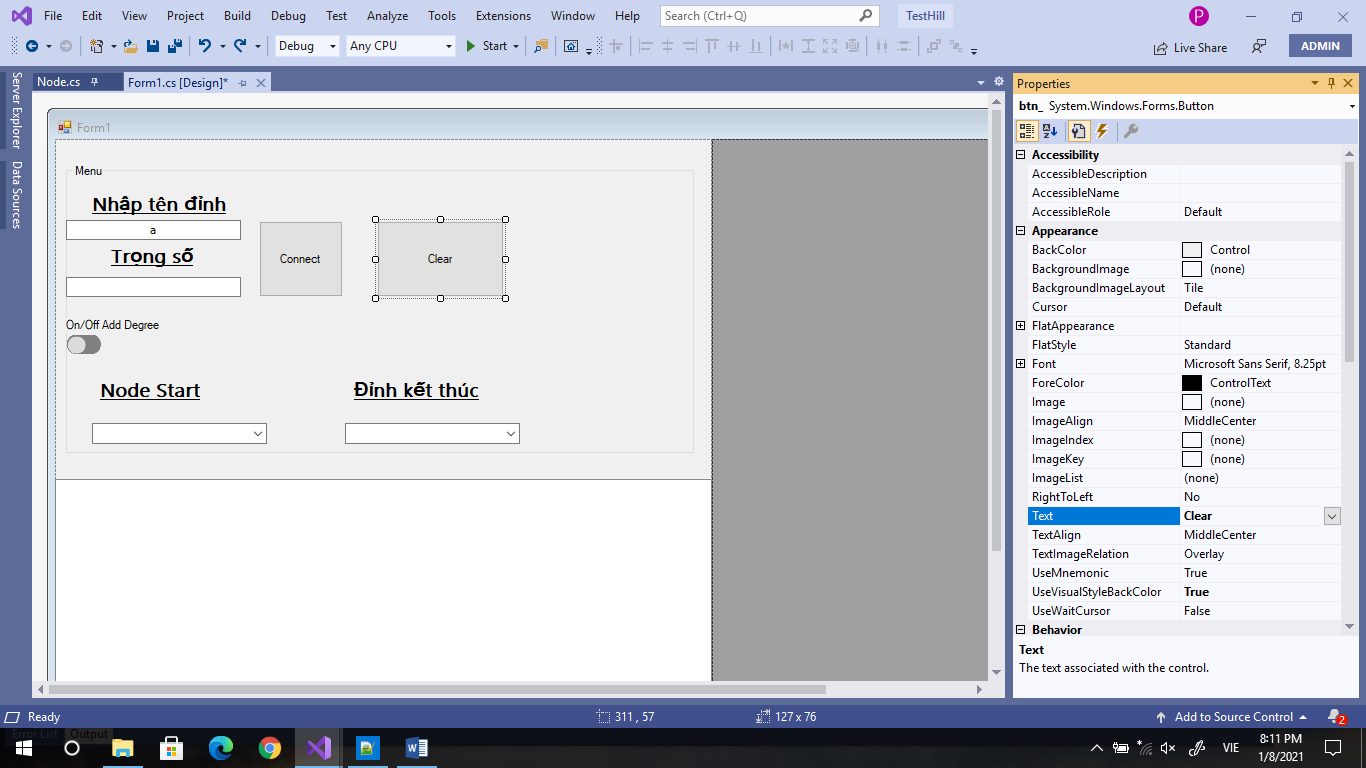
Thêm ô nhập liệu



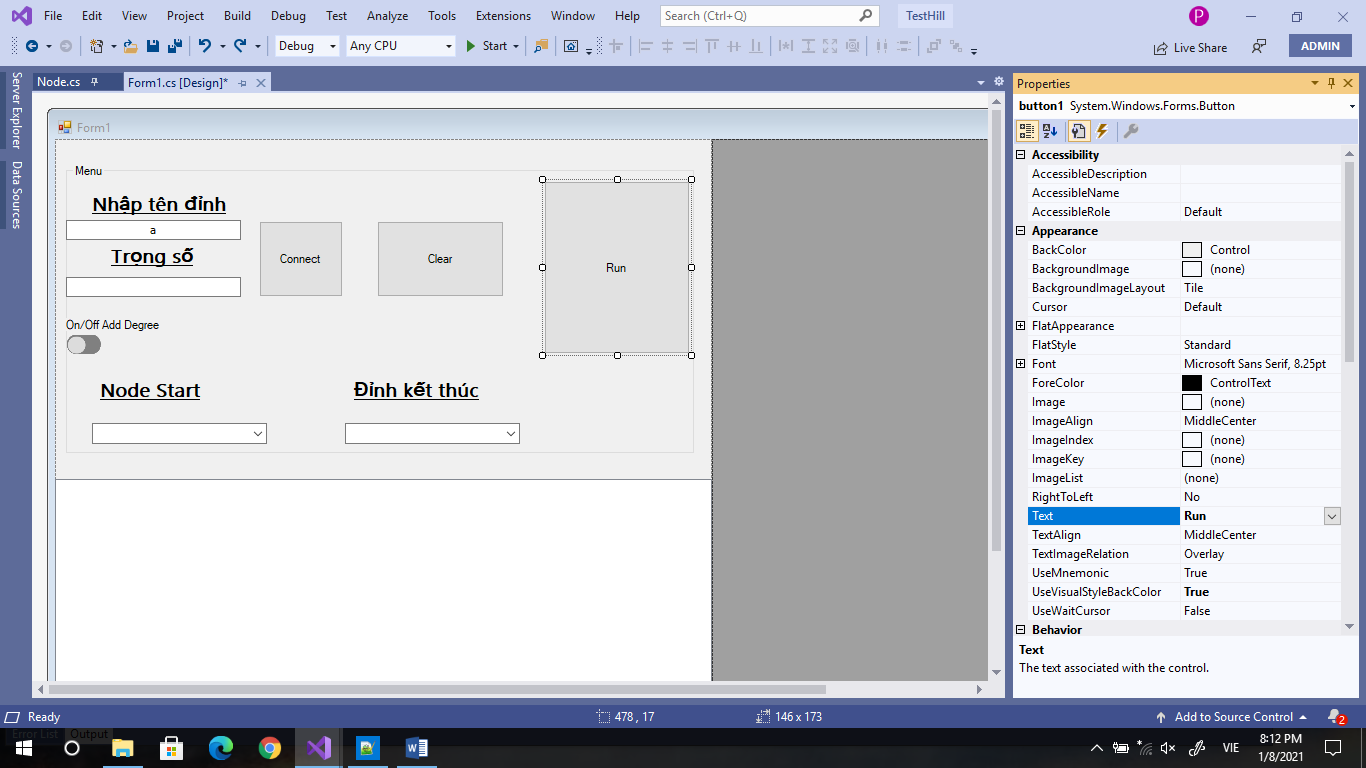
7.Thêm button đặt tên là connect, button có nhiệm vụ kết nối các đỉnh với nhau



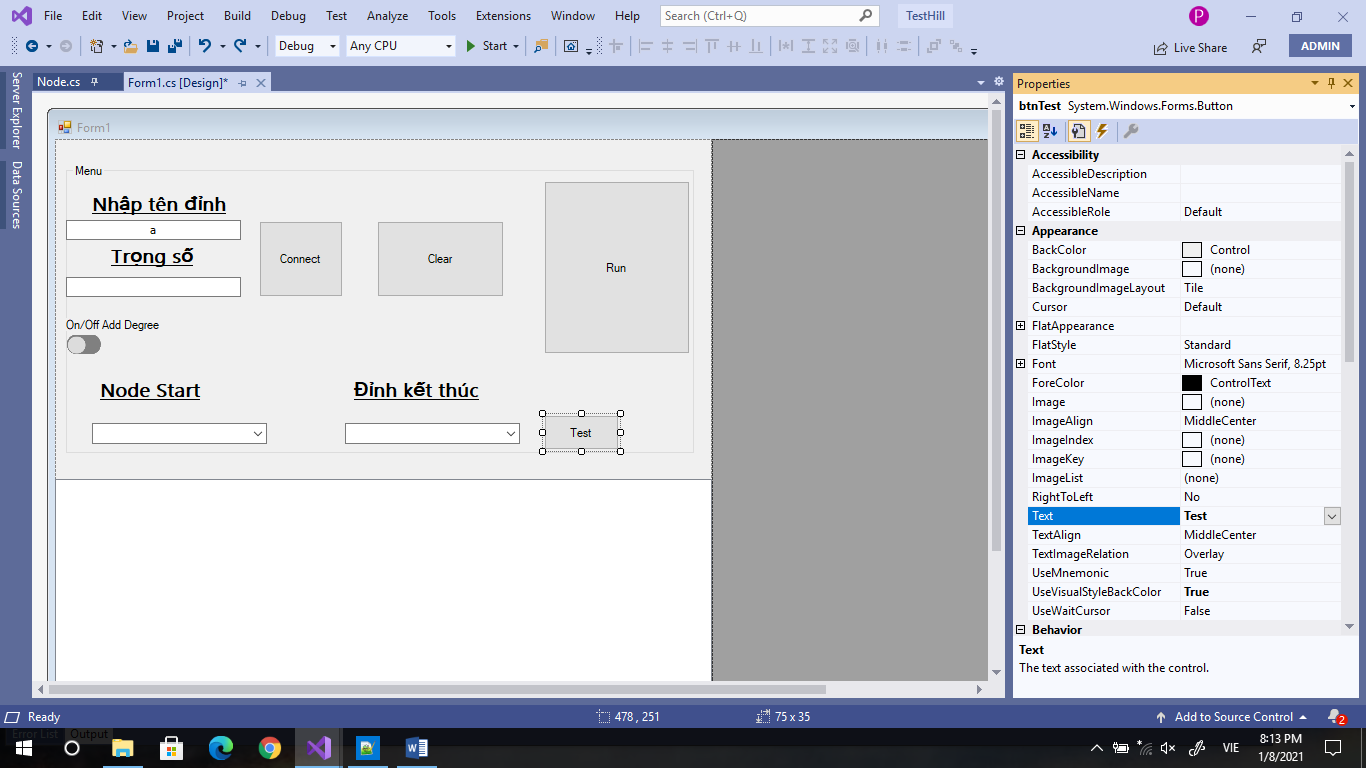
8.Thêm button Clear .button có nhiệm vụ làm sạch màn hình( tức khởi động lại và thực thi một bài toàn mới )



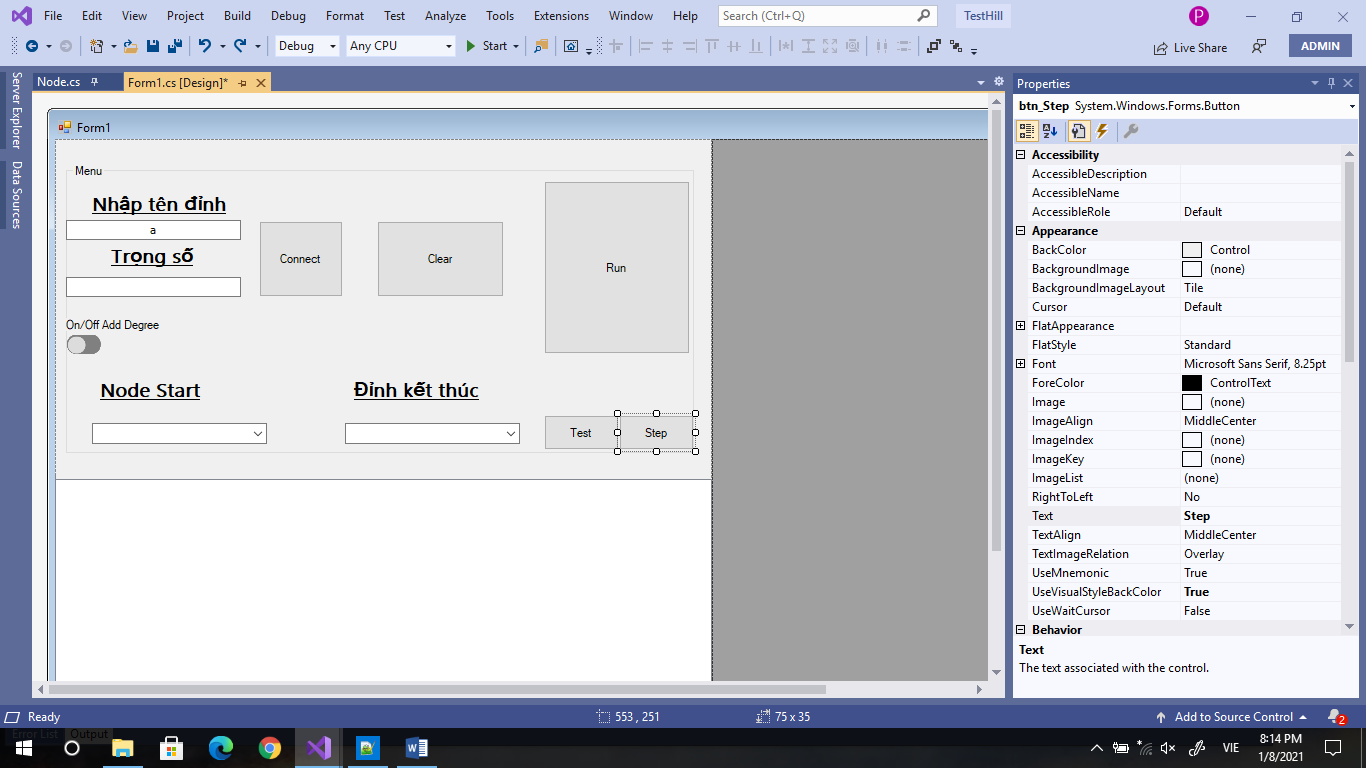
9. Thêm button Run, button có nhiệm vụ chạy ( thực thi ) chương trình



10. Thêm button Test. Button có nhiệm vụ kiểm tra

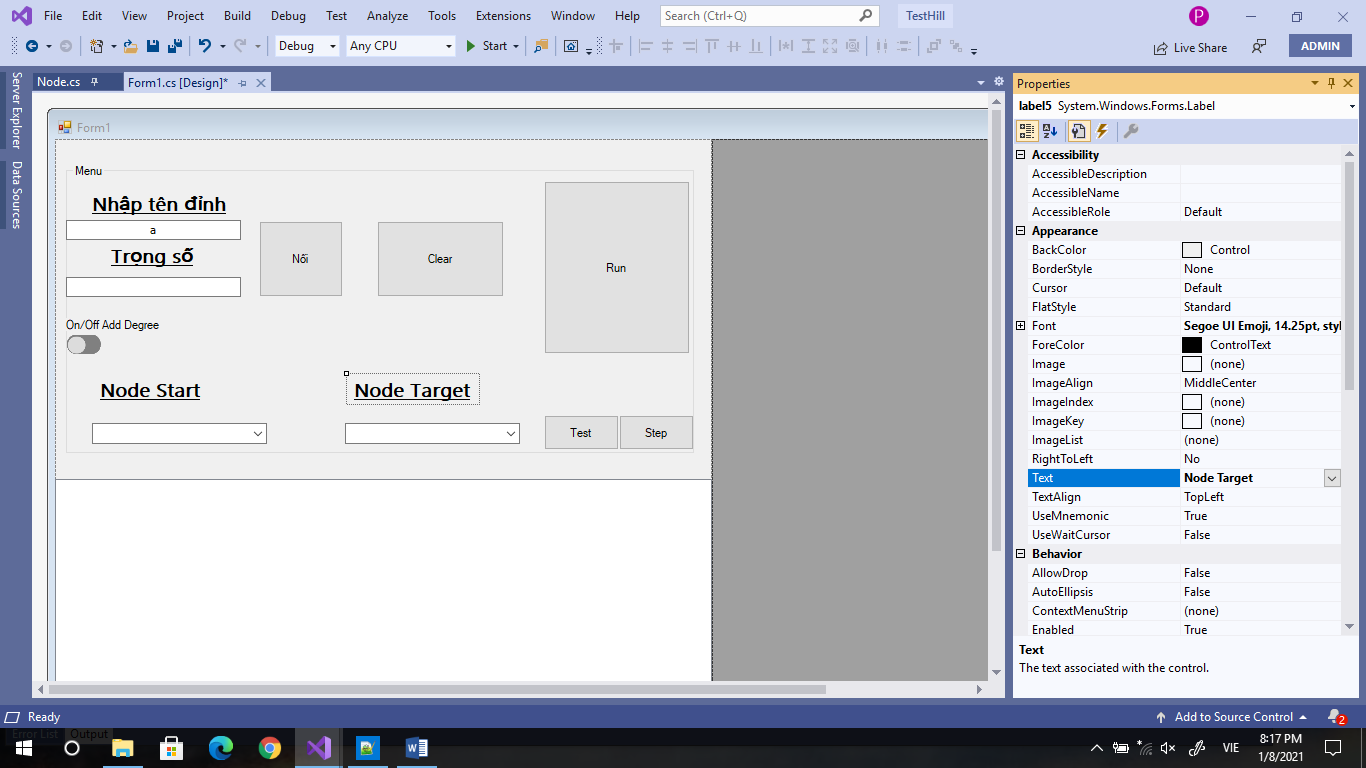


11.Thêm button Step, thực thi các lệnh nhập vào và thực thi từng bước của chương trình



12. Sử chữa và hoàn thiện Form:

Kiểm tra lại Form và chỉnh sửa cho cân đối. Ta thực hiện đổi tên Đỉnh kết thúc thành Node Target tại Label 5 : Text



13.Tiến hành cài đặt thuộc tính cho Form1

public partial class Form1 : Form

13.1 Cài đặt và khai báo các thuộc tính cho Form 1

#region Properties

string visited = "";

List<Node> L = new List<Node>();

List<Node> saveNode = new List<Node>();

List<Node> Visited = new List<Node>();

Node Target = new Node();

Node Start = new Node();

List<Point> points = new List<Point>();

Graphics gr;

Bitmap bm = new Bitmap(600, 600);

// Tạo bit map có tọa độ 600-600

Pen pDen = new Pen(Color.Black, 3); // Chọn màu đen để vẽ cạnh

Node temp = new Node();

bool onAddDegree = false;

int i = 0;

List<Button> buttons = new List<Button>();

List<Node> nodes = new List<Node>();

Queue<Node> queueNode = new Queue<Node>();

int k = 2;

#endregion

#region Initialize

public Form1()

{

InitializeComponent();

}

#endregion

13.2 Cài đặt phương thức

/\* Ý tưởng và Các công việc chưa hoàn thiện \*/

#region Methods

13.3 Khai báo tọa độ để vẽ các dòng

13.4 . Sự kiện Click tại button 1

13.5 Kiểm tra trung bình

13.6 Cắt chuỗi thua

13.7 . Cài đặt cho ô Reset

13.8 Chuyển màu nút

13.9 . Cài đặt bộ test và thuật toán test Hill

13.10. Thêm button

13.11 Khai báo nhập liệu

13.12 button test

13.13 Cài đặt vòng lặp do và thuật toán GBFS

13.14 button Clear

13.15 chọn