**Document Smart Contract Certificate**

1. **Yêu cầu đặt ra**

Tạo ra một **Smart Contract Certificate** để quản lý các chứng nhận của một người dùng nào đó. Yêu cầu cần phải có một chủ hợp đồng để quản lý và tạo ra danh sách các tài khoản admin, từ các tài khoản admin có thể quản lý thông tin người dùng cũng như chứng nhận của họ.

Chức năng của chủ hợp đồng:

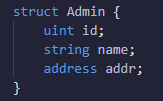
* Thêm, cập nhật, xóa thông tin admin.
* Thêm, cập nhật, xóa thông tin người dùng.
* Tạo ra các chứng nhận có thể chỉnh sửa, xóa, và có thể chuyển các chứng nhận cho một người dùng bất kỳ.
* Có thể cập nhật lại chủ sở hữu.

Chức năng của admin:

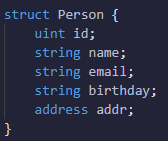
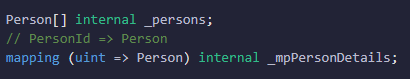
* Thêm, cập nhật, xóa thông tin người dùng.
* Tạo ra các chứng nhận có thể chỉnh sửa, xóa, và có thể chuyển các chứng nhận cho một người dùng bất kỳ.

1. **Phân tích yêu cầu**

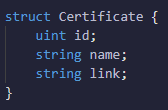
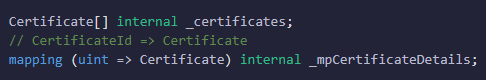
Tạo ra một cấu trúc để quản lý thông tin admin (id, name, addr), đồng thời tạo ra biến **\_admin** có kiểu dữ liệu là **Admin[]** để lưu danh sách thông tin admin, và tạo ra một mapping **\_mpAdminDetails** để ánh xạ từ một **adminId** ra thông tin **admin** một cách nhanh nhất. Phạm vi truy cập của 2 biến **\_admin, \_mpAdminDetails** đều là **internal.**

Tạo ra một cấu trúc để quản lý thông tin người dùng (id, name, email, birthday, addr), đồng thời tạo ra biến **\_persons** có kiểu dữ liệu là **Person[]** để lưu danh sách thông tin người dùng, và tạo ra một mapping **\_mpPersonDetails** để ánh xạ từ một personId ra thông người dùng. Phạm vi truy cập của 2 biến **\_persons, \_mpPersonDetails** đều là **internal.**

**** ****

Tạo một cấu trúc để quản thông tin chứng nhận (id, name, link), đồng thời tạo ra 2 biến **\_certificates** để lưu danh sách thông tin các chứng nhận, và tạo ra một mapping **\_mpCertificateDetails** để ánh xạ từ một **certificateId** ra thông chứng nhận. Phạm vi truy cập của 2 biến **\_certificates, \_mpCertificateDetails** đều là **internal.**

**** ****

Để biết được một chứng nhận nào đó đã thuộc về người dùng nào hay chưa ta cần tạo ra một mapping **\_mpCertificateToPersonId** để ánh xạ từ một **certificateId** ra thông tin của người dùng đang sở hữu chứng nhận này.



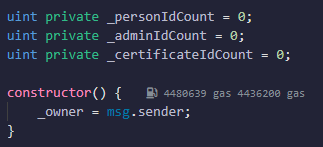
Và để có thể biết được người dùng này có những chứng nhận nào ta cũng tạo ra một mapping **\_mpCertificateOfPerson** dể ánh xạ từ một **personId** ra danh sách các chứng nhận mà người dùng đó đang sở hữu.



Để biết địa chỉ chủ hợp đồng đang là ai ta tạo ra một biến **\_owner** có kiểu dữ liệu là **address** và phạm vi truy cập là **internal.**

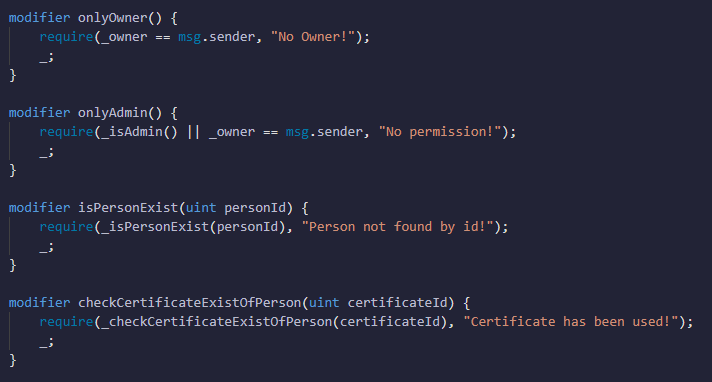
Ta tạo ra 3 biến **\_personIdCount, \_adminIdCount, \_certificateCount** để tạo ra **id** tự động cho 3 đối tượng **Admin, Person, Certificate** và phạm vi truy cập của 3 biến này là **private. (3 biến này luôn tăng)**

Với hàm **constructor** ta tiến hành cập nhật giá trị cho biến **\_owner** là địa chỉ của người deploy Smart Contract này.



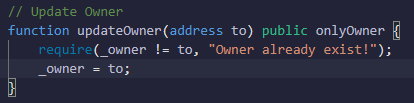
Theo yêu cầu của đặt ra ta tạo ra các modifier cần thiết như:

* **onlyOwner:** kiểm tra địa chỉ đang thực hiện chức năng có phải là chủ hợp đồng hay không nếu không phải thì đưa ra thông báo.
* **onlyAdmin:** kiểm tra địa chỉ đang thực hiện có phải là admin không hoặc có phải là chủ hợp đồng không nếu không thì đưa ra thông báo.
* **isPersonExist:** kiểm tra xem người dùng có **id** được truyền vào có tồn tại hay nếu không tồn tại người dùng thì đưa ra thông báo.
* **checkCertificateExistOfPerson:** kiểm tra chứng nhận có **id** được truyền vào đã được sử dụng hay chưa nếu đã được sử dụng thì đưa ra thông báo.



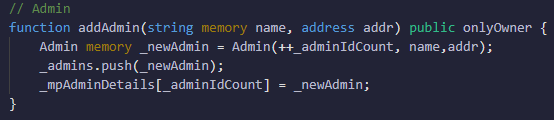
* 1. **Chức năng cập nhật chủ hợp đồng**

Để thực hiện được chức năng này ta cần phải modifier địa chỉ này có phải là chủ hợp đồng hay không trước khi vào hàm và khi vào bên trong hàm ta cần phải kiểm tra địa chỉ chủ hợp đồng mới có phải là địa chỉ của chủ hợp đồng hiện tại hay không. Nếu giống nhau thì đưa ra thông báo và kết thúc hàm ngược lại thì cập nhật lại chủ hợp đồng.



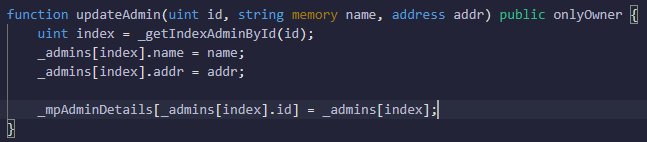
1. **Chức năng quản lý admin**
   1. **Chức năng thêm admin**

Để thực hiện hàm này ta cần truyền 2 tham số là **name, addr** và trước khi thực hiện hàm này ta cũng kiểm tra địa chỉ đang thực hiện có phải là chủ hợp đồng hay không. Nếu thỏa là chủ hợp đồng thì ta tiến hành tạo mới admin và lưu thông tin này vào 2 biến đã đề cập ở trên là **\_admins** và **\_mpAdminDetails.** Khi tạo mới admin thì biến **\_adminIdCount** cũng tự động cộng thêm 1.

****

* 1. **Chức năng cập nhật admin**

Để biết cập nhật thông tin cho admin nào thì ta cần truyền vào tham số **id** và các thông tin cần cập nhật cho admin là **name, addr** cũng tương tự như hàm thêm admin ta cũng kiểm tra địa chỉ đang thực hiện có phải là chủ hợp đồng trước khi vào hàm. Khi vào hàm ta kiểm tra id truyền vào có tồn tại hay không nếu không tồn tại thì kết thúc hàm. Ngược tại thì lấy ra thông tin admin theo **id** và cập nhật lại **name, addr**, đồng thời ta cũng cập nhật lại thông tin vào biến **\_mpAdminDetails.**

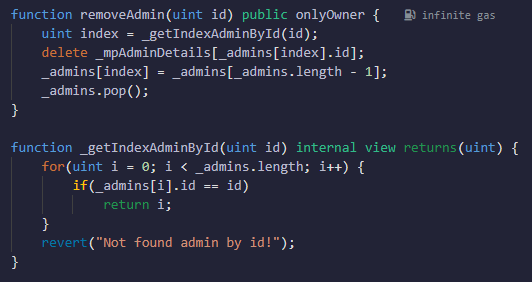


* 1. **Chức năng xóa admin**

Để biết xóa admin nào ta cần truyền vào tham số **id** và trước khi vào hàm ta kiểm tra xem địa chỉ đang thực hiện có phải là chủ hợp đồng hay không. Nếu thỏa ta lấy thông admin ra theo id và tiến hành xóa admin này trong 2 biến lưu trữ trước đó **\_mpAdminDetails, \_admins.**

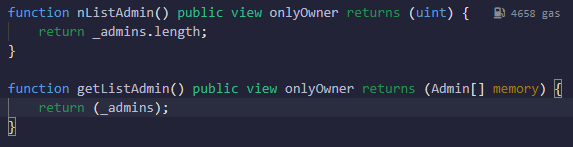
Để xóa admin trong mảng **Admin[]** ta không thể dùng **delete** như thường bởi vì khi xóa như vậy mảng vẫn không thay đổi kích thước và index của mảng không được liền mạch với nhau. Để giải quyết vấn đề này ta tiến hành làm như sau:

* Đầu tiên ta tìm index theo id admin cần xóa.
* Tiếp đến ta cập nhật lại thông tin admin này giống với thông tin admin ở cuối mảng.
* Cuối cùng dùng hàm **pop()** để xóa vị trí cuối của mảng.



* 1. **Chức năng xem số lượng và danh sách admin hiện tại**

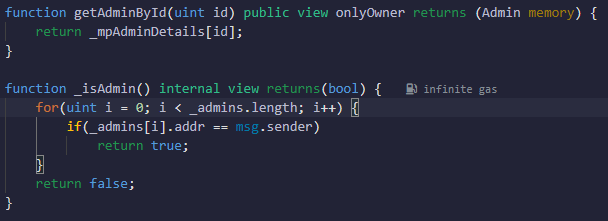
Ở trên ta sử dụng biến \_**admins** để lưu danh sách admin nên ta có thể biết được danh sách admin hiện tại và dể xem số lượng admin hiện có ta sử dụng **\_admins.length.**



* 1. **Chức năng xem chi tiết thông tin admin theo id**

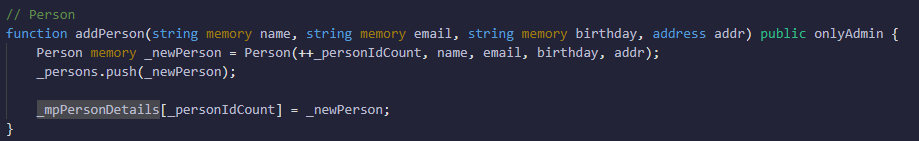
Để xem thông tin admin dựa vào id ta sử dụng biến **\_mpAdminDetails, \_mpAdminDetails** sẽ ánh xạ từ id ra thông tin của admin một cách nhanh chóng.

Hàm **\_isAdmin** sẽ kiểm tra xem địa chỉ đang thực hiện có tồn tại trong danh sách \_**admins** không nếu tồn tại thì trả về **true** ngược lại thì trả về **false.** Hàm này được sử dụng trong modifier **onlyAdmin**



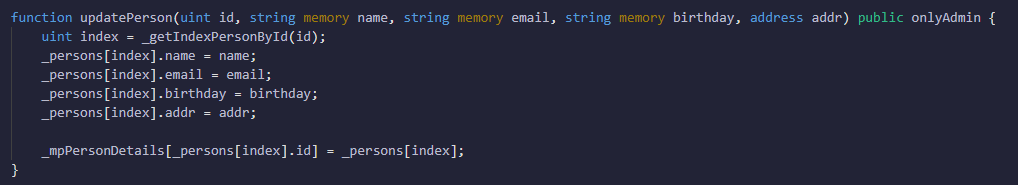
1. **Chức năng quản lý người dùng**
   1. **Chức năng thêm người dùng**

Để thực hiện hàm này ta cần truyền 4 tham số là **name, email, birthday, addr** và trước khi thực hiện hàm này ta sử dụng modifier **onlyAdmin**. Nếu thỏa điều kiện của modifier thì ta tiến hành tạo mới người dùng và lưu thông tin này vào 2 biến đã đề cập ở trên là **\_\_persons** và **\_mpPersonDetails.** Khi tạo mới người dùng thì biến **\_personIdCount** cũng tự động cộng thêm 1.



* 1. **Chức năng cập nhật người dùng**

Để biết cập nhật thông tin cho người dùng nào thì ta cần truyền vào tham số **id** và các thông tin cần cập nhật cho người dùng là **name, email, birthday, addr** cũng tương tự như hàm thêm người dùng ta cũng sử dụng modifier **onlyAdmin** trước khi vào hàm. Nếu thỏa điều kiện của modifier ta kiểm tra id truyền vào có tồn tại hay không nếu không tồn tại thì kết thúc hàm. Ngược tại thì lấy ra thông tin người dùng theo **id** và cập nhật lại **name, email, birthday, addr,** đồng thời ta cũng cập nhật lại thông tin vào biến **\_mpPersonDetails**.

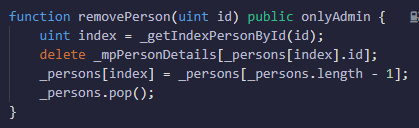


* 1. **Chức năng xóa người dùng.**

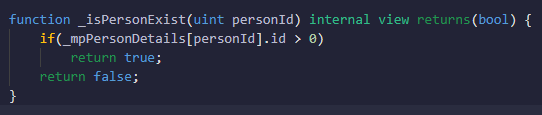
Để biết xóa người dùng nào ta cần truyền vào tham số **id** và trước khi vào hàm ta cũng sử dụng modifier **onlyAdmin** trước khi vào hàm. Nếu thỏa điều kiện modifier ta lấy thông người dùng ra theo id và tiến hành xóa người dùng này trong 2 biến lưu trữ trước đó **\_mpPersonDetails, \_persons.**

Để xóa người dùng trong mảng **Person[]** ta không thể dùng **delete** như thường bởi vì khi xóa như vậy mảng vẫn không thay đổi kích thước và index của mảng không được liền mạch với nhau. Để giải quyết vấn đề này ta tiến hành làm như sau:

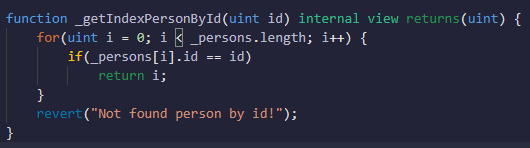
* Đầu tiên ta tìm index theo id người dùng cần xóa.
* Tiếp đến ta cập nhật lại thông tin người dùng này giống với thông tin người dùng ở cuối mảng.
* Cuối cùng dùng hàm **pop()** để xóa vị trí cuối của mảng.



Hàm **\_isPersonExist** dùng để kiểm tra người dùng có tồn tại hay không dựa vào tham số id được truyền vào. Ta sẽ dùng **\_mpPersonDetails** để ánh xạ từ **personId** và lấy ra thông tin người dùng. Khi dữ liệu được trả về ta chỉ cần kiểm tra **id** **> 0** hay không bới vì **id** luôn **> 0** từ đó ta sẽ biết được người dùng đó có tồn tại hay không.

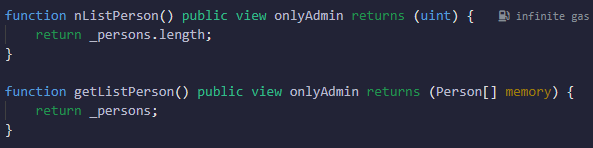
****

Hàm **getIndexPersonById** dùng để trả về vị trí của người dùng dựa vào id được truyền vào. Nếu không tìm thấy thì sẽ đưa ra thông báo.

****

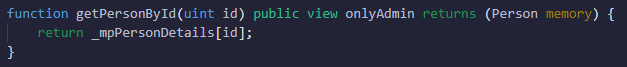
* 1. **Chức năng xem số lượng và danh sách người dùng hiện tại**

Ở trên ta sử dụng biến \_**persons** để lưu danh sách người dùng nên ta có thể biết được danh sách người dùng hiện tại và dể xem số lượng người dùng hiện có ta sử dụng **\_person.length.**



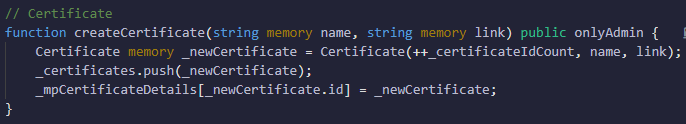
* 1. **Chức năng xem chi tiết người dùng theo id**

Để xem thông tin người dùng dựa vào id ta sử dụng biến **\_mpPersonDetails, \_mpPersonDetails** sẽ ánh xạ từ id ra thông tin của người dùng một cách nhanh chóng.

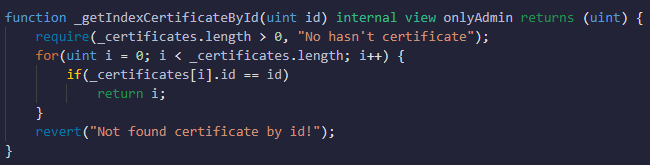


1. **Chức năng quản lý chứng nhận**
   1. **Chức năng tạo mới chứng nhận**

Để thực hiện hàm này ta cần 2 tham số là **name, link** và trước khi thực hiện hàm này ta cần phải kiểm tra địa chỉ người thực hiện có phải là admin không bằng cách sử dụng modifier **onlyAdmin** chúng ta đã định nghĩa ở trên**.** Nếu thỏa điều kiện của modifier thì tạo mới một đối tượng Certificate và thêm vào mảng chứng nhận hiện tại (**\_certificates)** đồng thời thêm vào biến **\_mpCertificateDetails** để các hàm sau có thể truy vấn chứng nhận theo id một cách nhanh chóng. Khi tạo mới đối tượng Certificate thì đồng thời biến \_certificateIdCount cũng được tăng lên 1.

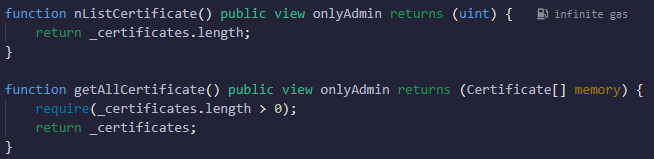


Hàm **getIndexCertitifcate** dùng để lấy vị trí của chứng nhận theo **id** nếu không tìm thấy chứng nhận nào thì hàm sẽ đưa ra thông báo.

****

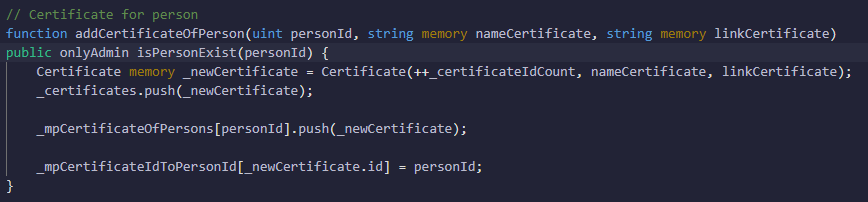
* 1. **Chức năng xem số lượng và danh sách chứng nhận hiện tại**

Ở trên ta đã sử dụng biến \_**certificates** có kiểu dữ liệu là **Certificate[]** nên ta dễ dàng lấy được số lượng và danh sách chứng nhận hiện tại.



* 1. **Chức năng tạo chứng nhận cho người dùng**

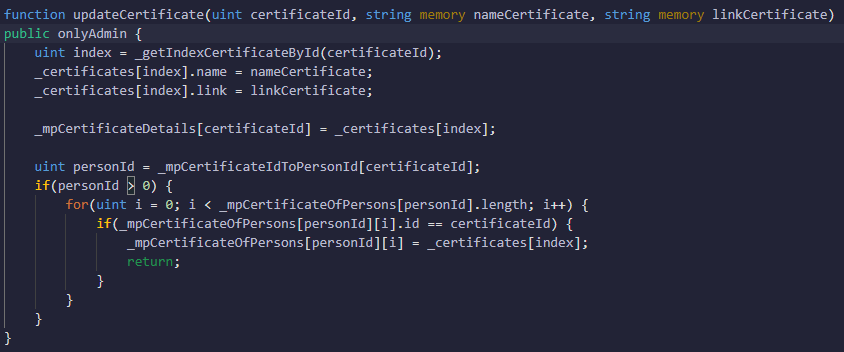
Để thực hiện được hàm này ta cần truyền vào 3 tham số là **personId, name, link** trong đó để biết được chứng nhận này được tạo cho người dùng nào thì ta cần phải truyền vào id của người dùng cần thêm chứng nhận. Và trước khi thực thi hàm ta cần phải kiểm tra địa chỉ của người đang thực hiện có phải là admin không bằng modifier **onlyAdmin** và kiểm tra id người dùng truyền vào có đúng không bằng modifier **isPersonExist** đã được định nghĩa ở trên. Nếu thỏa điều kiện của tất cả các modifier trên ta tiến hành khởi tạo một đối tượng **Certificate** mới với các tham số đã truyền vào và thêm đối tượng này vào mảng chứng nhận hiện tại, sau đó ta sử dụng biến \_**mpCertificateOfPerson** để ánh xạ từ id người dùng đã truyền vào và thêm chứng nhận mới vào danh chứng nhận hiện có của người dùng đó. Đồng thời sử dụng biến \_**mpCertificateIdToPerson** theo id của chứng nhận vừa tạo để lưu lại id của người dùng đã sở hữu chứng nhận này.



* 1. **Chức năng cập nhật chứng nhận**

Để sử dụng hàm cập nhật chứng nhận ta cần truyền vào 3 tham số là certificateId**, nameCertificate, linkCertificate**. Và cũng hàm tạo chứng nhận ta cũng dùng modifier **onlyAdmin** để kiểm tra xem địa chỉ đang thực hiện có phải là admin hay không. Nếu thỏa điều kiện của modifier ta tiến hành lấy vị trí theo id chứng nhận. Nếu tồn tại chứng nhận thì cập nhật lại theo các tham số truyền vào.

Tiếp theo ta kiểm tra chứng nhận này đã được sở hữu bởi người dùng nào chưa bằng biến \_**mpCertificateIdToPerson,** biến này sẽ ánh xạ từ id của chứng nhận và lấy ra id của người dùng và ta gán giá trị này vào **personId**. Ta kiểm tra nếu **personId > 0** nghĩa là chứng nhận đã được sở hữu lúc này ta sẽ lấy ra chứng nhận của người dùng này từ biến \_**mpCertificateOfPerson** và cập nhật lại thông tin.

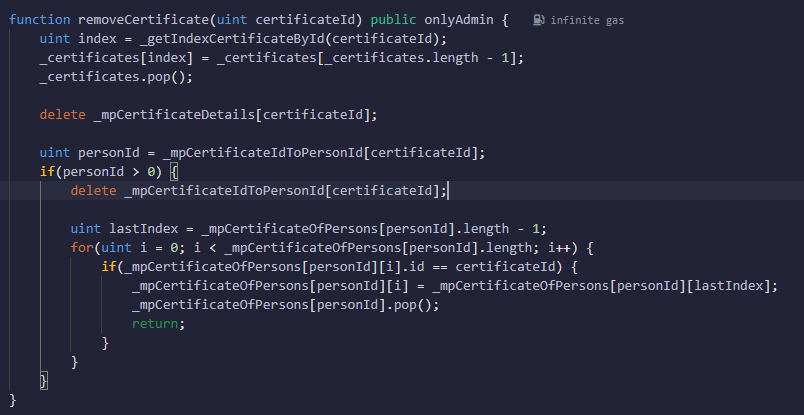


* 1. **Chức năng xóa chứng nhận**

Để sử dụng hàm xóa chứng nhận ta cần truyền vào 1 tham số là certificateId để biết được cần xóa chứng nhận nào. Và cũng hàm tạo chứng nhận ta cũng dùng modifier **onlyAdmin** để kiểm tra xem địa chỉ đang thực hiện có phải là admin hay không. Nếu thỏa điều kiện của modifier ta tiến hành lấy vị trí theo id chứng nhận. Nếu tồn tại chứng nhận ta tiến hành xóa chứng nhận tại vị trí đã tìm thấy. Trường hợp xóa dữ liệu trong mảng ta lưu ý là nếu dùng **delete** như ở mapping thì sẽ không được vì nó vẫn còn giữ nguyên **length** của mảng nên lúc này ta sẽ sử dụng cách khác.

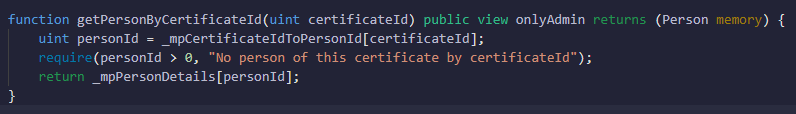
Đầu tiên ta cập nhật thông tin chứng nhận tại vị trí cần xóa theo thông tin của chứng nhận cuối cùng. Sau đó ta sử dụng **pop()** để xóa vị trí cuối mảng. Tiếp theo ta cũng xóa chứng nhận ở biến \_**mpCertificateDetails.**

Tiếp theo ta cũng tiến hành kiểm tra chứng nhận này có thuộc về người dùng nào hay không. Nếu chứng nhận đã được sở hữu ta tiến hành xóa chứng nhận đã được lưu trữ ở các biến \_**mpCertificateIdToPerson** theo **certificateId,** và xóa chứng nhận trên biến **\_mpCertificateOfPersons** theo **personId** đã tìm được dựa vào \_**mpCertificateIdToPerson.**

****

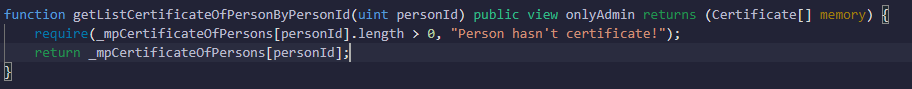
* 1. **Chức năng xem thông tin người dùng theo id chứng nhận**

Để biết được chứng nhận này đang được sở hữu bởi người dùng nào ta sử dụng hàm **getPersonByCertificateId** với tham số truyền vào là id của chứng nhận. Và trước khi thực thi hàm ta cũng dùng modifier **onlyAdmin** để kiểm tra xem địa chỉ đang thực hiện có phải là admin hay không. Nếu thỏa điều kiện modifier ta tiến hành tìm id người dùng bằng cách sử dụng biến **\_mpCertificateIdToPerson** để ánh xạ từ id chứng nhận ra id người dùng và gán giá trị này vào biến personId. Ta kiểm tra nếu **personId > 0** nghĩa là chứng nhận này đã được sở hữu và bây giờ ta chi cần sử dụng biến **\_mpPersonDetails** để ánh xạ từ **personId** vừa tìm được để lấy ra thông tin người dùng.



* 1. **Chức năng xem danh sách chứng nhận của người dùng**

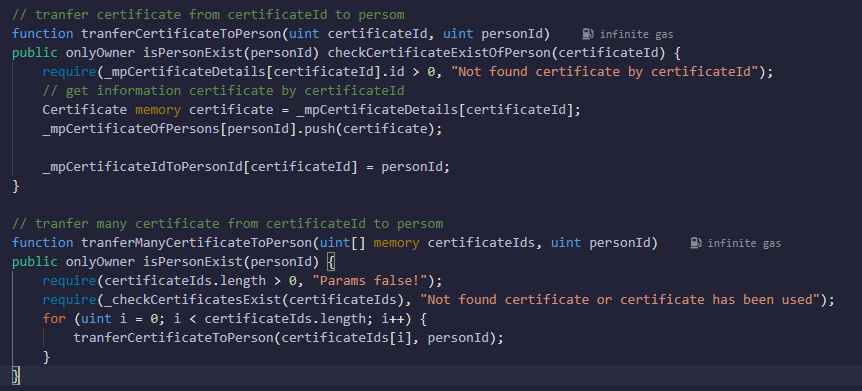
Để xem danh sách chứng nhận của người dùng ta sử dụng hàm **getListCertificateOfPersonByPersonId** với tham số truyền vào là id người dùng cần tra cứu. Và trước khi thực thi hàm ta cũng dùng modifier **onlyAdmin** để kiểm tra xem địa chỉ đang thực hiện có phải là admin hay không. Nếu thỏa điều kiện modifier ta tiến hành lấy dữ liệu từ biến **\_mpCertificateOfPerson** dựa vào id người dùng để ánh xạ ra danh sách chứng nhận của người dùng đó.



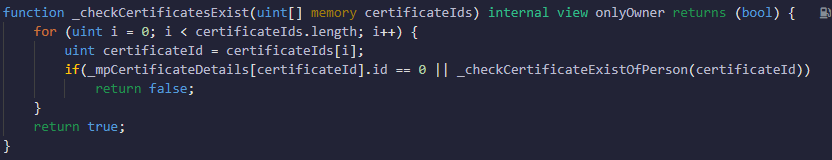
* 1. **Chức năng tranfer chứng nhận cho người dùng**

Để chuyển một chứng nhận cho một người dùng nào đó ta sử dụng hàm **tranferCertificateIdToPerson** với tham số là **certificateId** để biết chứng nhận nào cần chuyển**,** và **personId** để biết chứng nhận sẽ được chuyển cho người dùng nào. Và trước khi thực thi hàm ta cũng dùng modifier **onlyAdmin** để kiểm tra xem địa chỉ đang thực hiện có phải là admin hay không, nếu thỏa modifier này ta tiếp tục sử dụng modifier **isPersonExist** để kiểm id người dùng có tồn tại hay không nếu không tồn tại thì kết thúc hàm và ngược lại ta tiếp tục sử dụng modifier **checkCertificateExistOfPerson** để kiểm tra chứng nhận đã thuộc sở hữu của người dùng nào chưa nếu đã thuộc sở hữu thì kết thúc hàm ngược lại ta thực thi vào hàm và kiểm tra **id** chứng nhận có tồn tại hay không nếu không tồn tại thì kết thúc hàm ngược lại ta tiến hành lấy thông tin chứng nhận dựa vào id chứng nhận đã truyền vào, sau đó ta sử dụng biến **\_mpCertificateOfPerson** để thêm chứng nhận cho người dùng và cập nhật id người dùng vào biến \_**mpCertificateIdToPerson.**

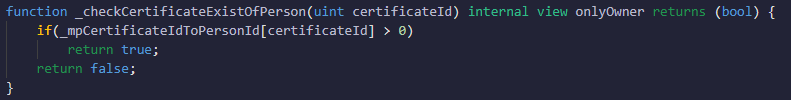
Để chuyển nhiều chứng nhận cho một người dùng ta cũng thực hiện các bước giống như trên.



Hàm \_**checkCertificateExist** dùng kiểm tra chứng nhận có tồn tại hay không với tham số là một mảng id chứng nhận. Với tham số này ta chỉ cần for và check với 1 điều kiện không thỏa thì return về **false** và ngược lại thì return về **true**.

****

Hàm \_**checkCertificateExistOfPerson** dùng để kiểm tra chứng nhận đã có người sở hữu chưa nếu có 1 người sở hữu thì return về **false** ngược lại thì return về **true.**

****