1. LinkedList: Danh sách liên kết đơn
2. Khởi tạo danh sách: Phương thức InitList() được sử dụng để khởi tạo danh sách. Ban đầu, cả hai con trỏ pHead và pTail được đặt thành null, chỉ ra rằng danh sách không có phần tử nào.
3. Kiểm tra danh sách rỗng: Phương thức IsEmptyList() kiểm tra xem danh sách có rỗng không bằng cách kiểm tra xem con trỏ pHead có trỏ đến null không.
4. Thêm phần tử vào đầu danh sách: Phương thức AddFirst(Node p) thêm một nút mới vào đầu danh sách. Nếu danh sách đang rỗng, nút mới được thêm trở thành cả pHead và pTail. Nếu không, nút mới được thêm vào trước pHead và pHead được cập nhật.
5. Thêm phần tử vào cuối danh sách: Phương thức AddLast(Node p) thêm một nút mới vào cuối danh sách. Nếu danh sách đang rỗng, nút mới được thêm trở thành cả pHead và pTail. Nếu không, nút mới được thêm vào sau pTail và pTail được cập nhật.
6. Xóa phần tử đầu danh sách: Phương thức RemoveFirst() xóa nút đầu tiên khỏi danh sách bằng cách di chuyển con trỏ pHead đến nút tiếp theo.
7. Xóa phần tử cuối danh sách: Phương thức RemoveLast() xóa nút cuối cùng khỏi danh sách bằng cách di chuyển con trỏ pTail đến nút trước nút cuối và cập nhật con trỏ pTail.
8. Xóa phần tử sau một nút cho trước: Phương thức RemoveAfter(Node q) xóa nút ngay sau nút được chỉ định bởi con trỏ q bằng cách cập nhật con trỏ pNext của nút trước nút cần xóa.
9. Xóa một nút cụ thể: Phương thức RemoveNode(Node q) xóa một nút cụ thể bằng cách tìm nút cần xóa và sau đó sử dụng các phương thức xóa phù hợp.
10. Tìm kiếm nút: Phương thức SearchNode(string value) tìm kiếm một nút trong danh sách với giá trị khóa tương ứng và trả về nút đầu tiên mà có giá trị khóa giống với giá trị được chỉ định.
11. So sánh chuỗi: Phương thức Compare(string x, string y) so sánh hai chuỗi, loại bỏ khoảng trắng và không phân biệt chữ hoa chữ thường.
12. MyDictionary:
13. LoadData(string path):

* Mở một luồng đầu vào từ tệp văn bản tại đường dẫn được chỉ định.
* Đọc từng dòng từ tệp. Chuyển đổi dòng đọc được thành chữ thường và cắt thành một mảng từ với từ khóa và giá trị, sử dụng dấu tab (\t) làm dấu phân cách.
* Thêm từ khóa và giá trị vào bảng băm (MyHashTable) thông qua phương thức
* AddKey(string key, string value).
* Đóng luồng đầu vào sau khi đọc xong.

1. AddDictionary(string key, string value):

* Kiểm tra tính hợp lệ của từ khóa bằng cách gọi phương thức IsValidString(string word).
* Nếu từ khóa hợp lệ, thêm cặp từ điển mới vào bảng băm thông qua phương thức AddKey(string key, string value) của đối tượng MyHashTable.

1. RemoveDictionary(string key): Gọi phương thức RemoveKey(string key) của đối tượng MyHashTable để xóa cặp từ điển với từ khóa được chỉ định.
2. ChangeDictionary(string key, string value): Tìm kiếm nút trong bảng băm với từ khóa được chỉ định thông qua phương thức SearchKey(string key). Thay đổi giá trị của nút được tìm thấy thành giá trị mới.
3. SearchDictionary(string key): Gọi phương thức SearchKey(string key) của đối tượng MyHashTable để tìm kiếm nút trong bảng băm với từ khóa được chỉ định. Trả về nút đầu tiên được tìm thấy.
4. IsValidString(string word): Kiểm tra tính hợp lệ của chuỗi từ bằng cách xác định xem chuỗi có null không và có chỉ chứa các ký tự chữ cái tiếng Anh không.
5. BackUp(string path):

* Mở một luồng đầu ra đến tệp văn bản tại đường dẫn được chỉ định.
* Duyệt qua tất cả các phần tử trong bảng băm và mỗi danh sách liên kết trong bảng băm.
* Ghi từng cặp từ điển (từ khóa và giá trị) vào tệp văn bản, mỗi cặp trên một dòng, sử dụng dấu tab (\t) để phân cách.
* Đóng luồng đầu ra sau khi ghi xong.

1. MyHashTable:
2. InitHashTable():

* Duyệt qua tất cả các phần tử trong mảng HT.
* Đối với mỗi phần tử, tạo một danh sách liên kết mới thông qua new LINKEDLIST() và khởi tạo danh sách đó bằng cách gọi phương thức InitList().

1. CreateNode(string key, string value):

* Tạo một đối tượng Node mới.
* Gán giá trị của từ khóa (key) và giá trị (value) cho nút mới.
* Thiết lập con trỏ pNext của nút mới thành null.
* Trả về nút mới đã được tạo.

1. HashFunction(string key):

* Khởi tạo một biến k để tính tổng các mã Unicode của các ký tự trong từ khóa.
* Duyệt qua từng ký tự trong từ khóa và cộng vào k giá trị Unicode của từng ký tự trừ đi 48.
* Trả về phần dư của k khi chia cho số lượng phần tử trong mảng băm (N).

1. AddKey(string key, string value):

* Sử dụng hàm băm (HashFunction) để tính toán vị trí trong mảng băm cho từ khóa được cung cấp.
* Tạo một nút mới bằng cách gọi phương thức CreateNode(string key, string value).
* Thêm nút mới này vào cuối danh sách liên kết tại vị trí tính toán trong mảng băm thông qua phương thức AddLast(Node p) của lớp LINKEDLIST.

1. SearchKey(string key):

* Sử dụng hàm băm (HashFunction) để tính toán vị trí trong mảng băm cho từ khóa được cung cấp.
* Tìm kiếm nút trong danh sách liên kết tại vị trí tính toán trong mảng băm thông qua phương thức SearchNode(string value) của lớp LINKEDLIST.
* Trả về nút đầu tiên tìm thấy (nếu có).

1. RemoveKey(string key):

* Sử dụng hàm băm (HashFunction) để tính toán vị trí trong mảng băm cho từ khóa được cung cấp.
* Tìm kiếm nút chứa từ khóa trong danh sách liên kết tại vị trí tính toán trong mảng băm thông qua phương thức SearchNode(string value) của lớp LINKEDLIST.
* Nếu nút được tìm thấy, loại bỏ nút đó khỏi danh sách liên kết thông qua phương thức RemoveNode(Node q) của lớp LINKEDLIST.