PHẦN BÀI LÀM CỦA SINH VIÊN NGUYỄN ĐÔNG NHẬT:

**PHẦN II: CHI TIẾT CÀI ĐẶT CHƯƠNG TRÌNH**

***\** MÔ TẢ CHI TIẾT:**

**1. Node**

- Gồm thuộc tính Key (lưu data)

- Gồm 2 con trỏ: Left, Right để chỉ hướng trỏ đến các Node khác.

private class Node

{

public string Key;

public Node Left;

public Node Right;

public Node(string key)

{

Key = key;

Left = null;

Right = null;

}

}

**2. BinarySearchTree**

- Là cấu trúc cây nhị phân, bao gồm các Node trong phần nêu trên.

class BinarySearchTree

{

private Node root;

}

- Có chức năng Insert:

+ Gặp node có Key lớn hơn thì tiếp tục duyệt sang Left, và ngược lại cho đến khi trỏ đến null thì thôi.

+ Tạo ra 1 node mới.

private Node Insert(Node node, string key)

{

if (node == null)

{

node = new Node(key);

}

else

{

if (string.Compare(key, node.Key) < 0)

{

node.Left = Insert(node.Left, key);

}

else

{

node.Right = Insert(node.Right, key);

}

}

return node;

}

public void Insert(string key)

{

root = Insert(root, key);

}

- Có chức năng Delete:

+ Gặp node có Key lớn hơn thì tiếp tục duyệt sang Left, và ngược lại cho đến khi gặp node có giá trị trùng với giá trị cần tìm.

+ Xoá node tương ứng khi tìm thấy, không thì thôi.

private Node Delete(Node node, string key)

{

if (node == null)

{

return node;

}

if (string.Compare(key, node.Key) < 0)

{

node.Left = Delete(node.Left, key);

}

else if (string.Compare(key, node.Key) > 0)

{

node.Right = Delete(node.Right, key);

}

else

{

if (node.Left == null)

{

return node.Right;

}

else if (node.Right == null)

{

return node.Left;

}

node.Key = MinValue(node.Right);

node.Right = Delete(node.Right, node.Key);

}

return node;

}

public void Delete(string key)

{

root = Delete(root, key);

}

- Chức năng MinValue: tìm ra node có giá trị nhỏ nhất.

private string MinValue(Node node)

{

string minValue = node.Key;

while (node.Left != null)

{

minValue = node.Left.Key;

node = node.Left;

}

return minValue;

}

**3. EntryDefintion**

- Gồm các thuộc tính: PartOfSpeech(loại từ), Definition(định nghĩa), Example(ví dụ)

- Mục đích: lưu lại cấu trúc của 1 mục từ trong hệ thống từ điển.

class EntryDefinition

{

public string PartOfSpeech { get; set; }

public string Definition { get; set; }

public string Example { get; set; }

}

**4. DictionaryEntry**

- Gồm các thuộc tính: Word, Definitions.

- Lưu lại 1 tập hợp gồm 1 từ và các định nghĩa của nó.

class DictionaryEntry

{

public string Word { get; set; }

public List<EntryDefinition> Definitions { get; set; }

public DictionaryEntry(string word)

{

Word = word;

Definitions = new List<EntryDefinition>();

}

}

**5. Dictionary**

**-** Gồm thuộc tính: hashTable (Hashtable), bst (BinarySearchTree)

- Mục đích: thao tác với các tập hợp mục từ trong hệ thống từ điển

class Dictionary

{

private Hashtable hashTable;

private BinarySearchTree bst;

}

**5.1. AddEntry(string word, string partOfSpeech, string definition, string example)**

- Tạo ra 1 DictionaryEntry entry với việc ép kiểu từ hashTable.

- Check xem trong bảng băm hashTable liệu có từ này không:

+ Nếu có thì thêm tập hợp đó vào trong entry.

+ Ngược lại thì thêm tập hợp đó vào trong entry, thêm vào trong hashTable và cuối cùng là thêm từ đó vào trong cây bst.

public void AddEntry(string word, string partOfSpeech, string definition, string example)

{

if (hashTable.ContainsKey(word))

{

DictionaryEntry entry = (DictionaryEntry)hashTable[word];

EntryDefinition entryDefinition = new EntryDefinition

{

PartOfSpeech = partOfSpeech,

Definition = definition,

Example = example

};

entry.Definitions.Add(entryDefinition);

}

else

{

DictionaryEntry entry = new DictionaryEntry(word);

EntryDefinition entryDefinition = new EntryDefinition

{

PartOfSpeech = partOfSpeech,

Definition = definition,

Example = example

};

entry.Definitions.Add(entryDefinition);

hashTable.Add(word, entry);

bst.Insert(word);

}

Console.WriteLine("Đã thêm mục từ vào từ điển.");

}

**5.2. RemoveEntry(string word)**

- Check xem trong bảng băm hashTable liệu có từ này không:

+ Nếu có thì xoá từ đó ở trong hashTable, xoá trong bst.

+ Ngược lại thì không làm gì.

public void RemoveEntry(string word)

{

if (hashTable.ContainsKey(word))

{

hashTable.Remove(word);

bst.Delete(word);

Console.WriteLine("Đã loại bỏ mục từ khỏi từ điển.");

}

else

{

Console.WriteLine("Không tìm thấy từ trong từ điển.");

}

}

**5.3. ConvertToDataTable(Dictionary evd):**

- Tạo ra 1 dataTable mới, thêm các tiêu đề: Word, PartOfSpeech, Definition, Example

**-** Duyệt qua từng tập hợp có trong từ điển và lưu dữ liệu vào trong dataTable.

public static DataTable ConvertToDataTable(Dictionary evd)

{

DataTable dataTable = new DataTable();

dataTable.Columns.Add("Word");

dataTable.Columns.Add("PartOfSpeech");

dataTable.Columns.Add("Meaning");

dataTable.Columns.Add("Example");

foreach (DictionaryEntry entry in evd.GetEntries())

{

foreach (EntryDefinition definition in entry.Definitions)

{

dataTable.Rows.Add(entry.Word, definition.PartOfSpeech, definition.Definition, definition.Example);

}

}

return dataTable;

}

**5.4. ConvertToDataTable():**

- Tạo ra 1 dataTable mới, thêm các tiêu đề: Word, PartOfSpeech, Definition, Example.

**-** Duyệt qua từng tập hợp (entry) có trong hashTable và lưu dữ liệu vào trong dataTable.

public DataTable ConvertToDataTable()

{

DataTable dataTable = new DataTable();

dataTable.Columns.Add("Word");

dataTable.Columns.Add("PartOfSpeech");

dataTable.Columns.Add("Meaning");

dataTable.Columns.Add("Example");

foreach (DictionaryEntry entry in hashTable.Values)

{

foreach (EntryDefinition definition in entry.Definitions)

{

dataTable.Rows.Add(entry.Word, definition.PartOfSpeech, definition.Definition, definition.Example);

}

}

return dataTable;

}