**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**SVTH : NGUYỄN LƯƠNG MỸ HÀ**

**MSSV : 16110060**

**SVTH : LÊ DUY QUÍ**

**MSSV : 16110190**

**TP. Hồ Chí Minh, tháng 12 năm 2017**

# MỤC LỤC

[MỤC LỤC 3](#_Toc500714979)

[DANH MỤC CÁC HÌNH 4](#_Toc500714980)

[NỘI DUNG 5](#_Toc500714981)

[1. MÔ TẢ PROJECT 5](#_Toc500714982)

[Sản phẩm đồ án: 5](#_Toc500714983)

[Phần mềm dùng để: 5](#_Toc500714984)

[2. MÔ TẢ QUÁ TRÌNH LÀM 5](#_Toc500714985)

[Cách thức xây dựng phần mềm: 5](#_Toc500714986)

[Thiết kế giao diện: 5](#_Toc500714987)

[Thiết kế lớp: 6](#_Toc500714988)

[Mô tả phương thức của mỗi lớp: 6](#_Toc500714989)

[Lớp Node.cs: 6](#_Toc500714990)

[Lớp BinarySearchTree.cs: 8](#_Toc500714991)

[TÀI LIỆU THAM KHẢO 9](#_Toc500714992)

# NỘI DUNG

## MÔ TẢ PROJECT

### Sản phẩm đồ án:

Làm phần mềm “Từ điển ứng dụng thuật toán cây nhị phân tìm kiếm”.

### Phần mềm dùng để:

Giúp người dùng có thể tra cứu từ điển Anh-Việt.

## MÔ TẢ QUÁ TRÌNH LÀM

### Cách thức xây dựng phần mềm:

Ứng dụng thuật toán cây nhị phân tìm kiếm lấy dữ liệu mỗi từ trong file “Dictionary.txt” cho vào phần mềm từ điển. Từ điển cho phép người dùng nhập thêm từ mới hoặc thêm từ đó vào danh sách những từ yêu thích, ngoài ra còn có thể tra cứu từ điển trên web <https://dictionary.cambridge.org/>.

### Thiết kế giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| TT | (Tên màn hình trên 1 dòng; dưới dòng này là hinh chụp màn hình) | Người thiết kế & giải thích ngắn gọn các quyết định thiết kế khi thiết kế màn hình (Người thiết kế ở trên 1 dòng. Sau dòng đó là phần giải thích. | Mục đích chính của màn hình |
| 1 | Form TuDienAV.cs | Nguyễn Lương Mỹ Hà (60%)  Lê Duy Quí (40%)  Giải thích thiết kế:  Thiết kế đơn giản, dễ sử dụng.  Hướng đối tượng người sử dụng phần mềm tra cứu từ điển.  Form sáng, nút bấm to rõ dễ thao tác. | Hiển thị từ điển và tra cứu.  Cho phép người dùng xóa từ cần xóa trong từ điển  Ngoài ra có thể gọi lệnh thêm từ, tra cứu được những từ trong danh sách yêu thích.. |
| 2 | Form ThemTu.cs | Lê Duy Quí  Form thiết kế đơn giản dễ hiểu gồm 2 textbox và một button “Thêm” | Cho phép người dùng nhập thông tin từ cần thêm vào từ điển |

### Thiết kế lớp:

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên lớp | Tên các SV phụ trách viết  (Nếu là code lấy từ ngồn khác thì phải chỉ rõ lấy từ ngồn nào). | Mục đích chính của lớp trong chương trình |
| 1 | Word.cs | Lê Duy Quí | Dùng để lưu trữ đối tượng từ |
| 2 | Node.cs | Nguyễn Lương Mỹ Hà (80%)  Lê Duy Quí(20%) | Khởi tạo, khai báo cấu trúc 1 node của cây nhị phân tìm kiếm và các phương thức hàm của node. |
| 3 | BinarySearchTree.cs | Nguyễn Lương Mỹ Hà | Khởi tạo, khai báo cấu trúc 1 cây nhị phân tìm kiếm và các phương thức hàm của cây. |

### Mô tả phương thức của mỗi lớp:

a.Class Word.cs

public string Eng { get; set; }

public string Viet { get; set; }

public Word(string name, string meaning)

{

this.Eng = name;

this.Viet = meaning;

}

b.Class Node.cs

public string English;

public string Vietnamese;

public Node Left;

public Node Right;

public Node(string TiengAnh, string TiengViet)

{

English = TiengAnh;

Vietnamese = TiengViet;

Left = null;

Right = null;

}

public bool isLeaf(ref Node node)

{

return (node.Right == null && node.Left == null);

}

public void InsertNode(ref Node node, string TiengAnh, string TiengViet)

{

if (node == null)

{

node = new Node(TiengAnh, TiengViet);

}

else

{

if (node.English.CompareTo(TiengAnh) > 0)

{

InsertNode(ref node.Left, TiengAnh, TiengViet);

}

else

{

if (node.English.CompareTo(TiengAnh) < 0)

{

InsertNode(ref node.Right, TiengAnh, TiengViet);

}

}

}

}

public void DelNode(ref Node node, string chuoi)

{

if (node.English.CompareTo(chuoi)>0)

{

DelNode(ref node.Left, chuoi);

}

else if(node.English.CompareTo(chuoi)<0)

{

DelNode(ref node.Right, chuoi);

}

else

{

Node p = node;

if (node.Left == null)

{

node = node.Right;

}

else

{

if (node.Right == null)

{

node = node.Left;

}

else

{

thayThe(p, node.Left);

}

}

}

}

public void thayThe(Node p, Node node)

{

if (node.Right != null)

{

thayThe(p, node.Right);

}

else

{

node.English = p.English;

node.Vietnamese = p.Vietnamese;

p = node;

node = node.Left;

}

}

public string search(Node node, string TiengAnh)

{

if (node == null)

return null;

if (node.English.CompareTo(TiengAnh) == 0)

{

return node.Vietnamese;

}

else if (node.English.CompareTo(TiengAnh) < 0)

{

return search(node.Right, TiengAnh);

}

else if (node.English.CompareTo(TiengAnh) > 0)

{

return search(node.Left, TiengAnh);

}

return null;

}

public void display(Node n)

{

if (n == null)

return;

display(n.Left);

//Console.WriteLine(n.English.ToString() + " : " +n.Vietnamese.ToString());

display(n.Right);

}

}

c.Class BinarySearchTree.cs

public Node Root;

public int count;

public BinarySearchTree()

{

count = 0;

Root = null;

}

public void del(string chuoi)

{

{

Root.DelNode(ref Root, chuoi);

}

}

public void Insert(string TiengAnh, string TiengViet)

{

if (isEmpty())

{

Root = new Node(TiengAnh, TiengViet);

}

else

{

Root.InsertNode(ref Root, TiengAnh, TiengViet);

}

count++;

}

public string search(string sTiengAnh)

{

return Root.search(Root, sTiengAnh);

}

public bool isLeaf()

{

if (!isEmpty())

return Root.isLeaf(ref Root);

return true;

}

public void display()

{

if (!isEmpty())

{

Root.display(Root);

}

}

public bool isEmpty()

{

return Root == null;

}

public int Count()

{

return count;

}

TÀI LIỆU THAM KHẢO:

<https://msdn.microsoft.com/en-us/>

<https://stackoverflow.com/>

<https://github.com/>